Г. И. СВЕТОЗАРОВА
А. А. МЕЛЬНИКОВ
А. В. КОЗЛОВСКИЙ

# ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ БЕЙСИК



Г. И. СВЕТОЗАРОВА А. А. МЕЛЬНИКОВ А. В. КОЗЛОВСКИЙ

Г. И. СВЕТОЗАРОВА,
А. А. МЕЛЬНИКОВ,
А. В. КОЗЛОВСКИЙ

# ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ БЕЙСИК

Под редакцией академика
С. В. ЕМЕЛЬЯНОВА

*Допущено Государственным комитетом СССР по народному образованию в качестве учебного пособия для студентов вузов*



МОСКВА «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
1988



Scan AAW

ББК 22.18
С24
УДК 519.682(075.8)

Светозарова Г.И., Мельников А.А., Козловский А.В. **Практикум по программированию на языке бейсик:** Учеб. пособие для вузов.— М.: Наука. Гл. ред. физ.-мат. лит., 1988.—368 с.— ISBN 5-02-013798-7.

Книга предназначена для приобретения практических навыков алгоритмизации задач и программирования на языке бейсик. В ней приводятся основные приемы и методы программирования, большое число типовых алгоритмов для решения задач различных классов (в том числе «невычислительного» характера) и примеры их использования. Дается описание языка бейсик.

Содержит более 500 задач из различных областей для самостоятельного решения, систематизированных в соответствии с используемыми методами и объединенных в 14 работ.

Для студентов, преподавателей, инженеров, а также для начинающих изучать программирование.

Табл. 9. Ил. 278. Библиогр. 24 назв.

Рецензенты:

кандидат физико-математических наук *С. А. Смагин*
кафедра прикладной математики МЭИ

*СВЕТОЗАРОВА Галина Ивановна, МЕЛЬНИКОВ Александр Алексеевич, КОЗЛОВСКИЙ Алексей Вадимович*

ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ БЕЙСИК

Редактор *О. Ю. Меркадер*
Художественный редактор *Т. Н. Кольченко*
Технический редактор *С. Я. Шкляр*
Корректоры *Т. С. Вайсберг, М. Л. Медведская*

ИБ № 32557

Сдано в набор 03.03.88. Подписано к печати 11.10.88. Т-17885. Формат 84×108/32. Бумага тип. № 2. Гарнитура литературная. Печать высокая. Усл. печ. л. 19,32. Усл. кр.-отт. 19,32. Уч.-изд. л. 23,89. Тираж 235 000 экз. (1-й завод 1—100 000 экз.). Заказ № 8-354. Цена 1 р. 20 к.

Ордена Трудового Красного Знамени издательство «Наука»
Главная редакция физико-математической литературы
117071 Москва В-71, Ленинский проспект, 15

Ордена Октябрьской Революции и ордена Трудового Красного Знамени
МПО «Первая Образцовая типография»
Союзполиграфпрома при Государственном комитете СССР
по делам издательств, полиграфии и книжной торговли
113054 Москва, Валовая, 28

Отпечатано с матриц на Киевской книжной фабрике,
252054 Киев-54, Воровского, 24

С $\frac{1702070000-187}{053(02)-88}$ 77-88

ISBN 5-02-013798-7

© Издательство «Наука». Главная редакция физико-математической литературы, 1988

# СОДЕРЖАНИЕ

Предисловие редактора . . . . . 5
Предисловие авторов . . . . . 7
Введение. Что такое ЭВМ? . . . . . 9

**Глава 1. Введение в программирование** . . . . . 13
1. Основные понятия программирования . . . . . 13
2. Понятие алгоритма . . . . . 18
3. Основные структуры алгоритмов. Понятие о структурном подходе к разработке алгоритмов . . . . . 25
Упражнения . . . . . 28

**Глава 2. Основные приемы программирования** . . . . . 37
1. Организация циклов . . . . . 37
2. Типы ошибок в программе. Исправление ошибок 42
3. Составление программ с использованием ввода данных 46
4. Порядок решения задач с использованием ЭВМ . . . 49
5. Составление программ разветвляющейся структуры. Контроль ввода данных . . . . . 53
6. Составление программ для обработки потока данных 58
7. Этапы решения задачи на ЭВМ. Некоторые практические рекомендации по разработке и отладке программ 60

**Глава 3. Описание языка бейсик** . . . . . 65
1. Общая характеристика алгоритмического языка бейсик . . . . . 65
2. Символы языка бейсик . . . . . 66
3. Простейшие конструкции языка . . . . . 67
3.1. Числа (67). 3.2. Переменные (68). 3.3. Стандартные функции (68). 3.4. Арифметические выражения (69).
4. Основные операторы бейсика . . . . . 70
4.1. Оператор присваивания (70). 4.2. Оператор безусловного перехода (70). 4.3. Условные операторы (70). 4.4. Операторы цикла (72). 4.5. Оператор-комментарий (73). 4.6. Операторы STOP и END (74).
5. Операторы ввода/вывода . . . . . 75
5.1. Операторы READ, DATA (74). 5.2. Оператор RESTORE (75). 5.3. Оператор INPUT (76). 5.4. Оператор PRINT (77). 5.5. Оператор PRINT USING (79). 5.6. Использование функции TAB в операторе PRINT (81).
6. Символьные переменные . . . . . 82
7. Массивы . . . . . 86
8. Матричные операторы . . . . . 92
9. Подпрограммы . . . . . 95

10. Определение нестандартных функций оператором DEF FN . . . . . .
11. Вычисляемые переходы. Операторы ON . . . . .
12. Ввод, исправление, выполнение программы. Работа за дисплеем . . . . . .

12.1. Ввод и выполнение программы (102). 12.2. Исправление программы (103). 12.3. Просмотр программы на экране. Очистка памяти. Задание имени программы (104). 12.4. Клавишные команды (105). 12.5. Режим непосредственного исполнения и его использование при отладке программ (106).

13. Работа с внешними устройствами . . . . . .

13.1. Использование печатающего устройства (107). 13.2. Использование внешних запоминающих устройств (109).

14. Специальные приемы разработки и организации программ сложной структуры . . . . . .

14.1. Методы разработки программ (115). 14.2. Команда APPEND (116). 14.3. Оператор COMMON (117). 14.4. Оператор CHAIN (117). 14.5. Оператор OVERLAY (117).

Глава 4. **Основы программирования на бейсике** . . . . .
*Работа 1.* Простейшие программы линейной структуры. Режим непосредственного исполнения. Программный режим . . . . . .
*Работа 2.* Простейшие программы циклической структуры
*Работа 3.* Ввод данных. Разветвления. Циклы и разветвления . . . . . .
*Работа 4.* Простейшие алгоритмы обработки массивов . . .
*Работа 5.* Использование подпрограмм и функций . . . .
*Работа 6.* Использование вычислительных методов . . . .
*Работа 7.* Вывод результатов в виде графиков, таблиц, гистограмм. Построение геометрических фигур
*Работа 8.* Решение задач с использованием массивов . . .
*Работа 9.* Обработка текстовых данных . . . . . .
*Работа 10.* Простейшие приемы работы с внешними устройствами . . . . . .

Глава 5. **Практика программирования на бейсике** . . .
*Работа 11.* Специальные приемы работы с целыми числами
*Работа 12.* Программирование игр . . . . . .
*Работа 13.* Разработка и выполнение программ сложной структуры с использованием внешних устройств

Дополнительные задачи . . . . . .

Список алгоритмов . . . . . .

Приложение 1. Символы бейсика в порядке старшинства и их семиразрядные восьмеричные коды . . . . . .
Приложение 2. Служебные слова и сокращения . . .
Приложение 3. Работа на бейсике на ЭВМ серии СМ в системе ОС РВ . . . . . .
Приложение 4. Работа на бейсике на ЭВМ серии «Электроника» в системе ФОБОС (ФОДОС)
Приложение 5. Сообщения об ошибках . . . . . .
Приложение 6. Команды бейсика . . . . . .

Список литературы . . . . . .

## ПРЕДИСЛОВИЕ РЕДАКТОРА

Постоянно возрастающая роль вычислительной техники в различных областях деятельности человека вызывает все большую необходимость широкой подготовки всех оканчивающих вуз специалистов в области программирования и использования ЭВМ как средства для решения возникающих на практике задач.

Современное применение ЭВМ в значительной степени связано с мини-, микро- и персональными ЭВМ и ориентировано на решение задач невычислительного характера. Однако учебные пособия, обеспечивающие подготовку студентов в этом направлении, практически отсутствуют.

Предлагаемый практикум восполняет имеющийся пробел. Практикум предназначен для начального изучения программирования студентами вузов различного профиля. В нем рассматриваются все аспекты подготовки и решения задачи на ЭВМ, включая методы формализации задач, приемы алгоритмизации, методы разработки программ, рекомендации по отладке и тестированию.

На основании анализа большого числа практических задач авторами были выделены и систематизированы типовые алгоритмы и приемы решения различных классов задач. Для многих классов задач это сделано впервые.

Последовательность и форма изложения методически глубоко продуманы и обеспечивают постепенное и последовательное усвоение материала. Большое число наглядных, хорошо подобранных примеров облегчает восприятие.

Для самостоятельного выполнения предлагаются интересные задачи (всего более 500) из различных областей,

в частности, игры, обработка текстов, физика, вычислительные задачи, комбинаторика, информационные системы, социологические исследования, контроль знаний и т. д. Задачи даются, как правило, в реальной постановке и требуют обязательного выполнения всех этапов работы по подготовке и решению задачи на ЭВМ, что будет способствовать приобретению практических навыков по программированию и использованию ЭВМ в различных областях.

Практикум обеспечивает возможность самостоятельной работы над курсом, что полностью согласуется с современным подходом к обучению в высшей школе.

В книге нашла отражение методика преподавания программирования, успешно используемая в МИСиС при обучении студентов, а также преподавателей и сотрудников в рамках факультетов повышения квалификации.

Можно надеяться, что настоящий практикум будет полезен при преподавании курсов, связанных с применением вычислительной техники в других вузах, и при самостоятельном изучении программирования.

Академик *С. В. Емельянов*

# ПРЕДИСЛОВИЕ АВТОРОВ

Предлагаемый читателю практикум предназначен для изучения программирования на младших курсах вузов любого профиля. Он ориентирован на выполнение программ в режиме диалога на мини-, микро- и персональных ЭВМ.

В качестве языка программирования используется алгоритмический язык бейсик, получивший в последнее время широкое распространение. Этот язык достаточно прост, но в то же время включает широкий набор средств, позволяющих решать на ЭВМ разнообразные задачи.

Практикум включает введение и пять глав. Рисунки и программы в книге имеют двойную нумерацию (первый — номер главы, второй — номер внутри данной главы).

Во введении дается краткое представление об ЭВМ.

В главе 1 вводятся основные понятия программирования, дается представление об алгоритмическом методе, используемом при решении задач на ЭВМ, и современных принципах разработки алгоритмов. Для облегчения понимания основных идей программирования используется так называемый «естественный» язык, позволяющий без излишней формализации записывать алгоритм в виде последовательного текста, что позволяет начать практическую работу по курсу с самых первых занятий и облегчает последующий переход к алгоритмическим языкам.

В главе 2 на доступном уровне рассматриваются основные аспекты решения задач на ЭВМ, приводятся широко применяемые в настоящее время типовые приемы и методы разработки программ и даются рекомендации по проверке их правильности.

В главе 3 излагаются средства языка бейсик. Конструкции языка иллюстрируются примерами законченных программ.

Глава 4 включает задания для самостоятельного выполнения, объединенные в 10 работ. Работы различаются типами используемых алгоритмов, приемами и методами программирования, и классами задач. Работы, составляющие содержание главы 4, подобраны так, чтобы показать возможности применения ЭВМ для решения широкого круга задач вычислительного и особенно невычислительного характера, и иллюстрируют диалоговые возможности языка бейсик. Основная часть задач приводится в реальной постановке, что позволяет приобрести навыки формализации задач для их решения на ЭВМ.

Глава 5 включает работы, предназначенные для углубленного изучения специальных приемов и методов программирования. Их выполнение рекомендуется при достаточном объеме курса после полного усвоения материала главы 4.

Каждая работа включает теоретическое введение, в котором приводятся типовые алгоритмы и приемы решения соответствующего класса задач, задачи для самостоятельного решения и указания к их решению, а также контрольные вопросы, которые могут использоваться как для самопроверки, так и для контроля преподавателем. Каждая работа содержит задания трех уровней:

— задание I уровня требует решения простых задач, которые сводятся к типовым алгоритмам, изложенным во введении к работе, и по существу представляют собой упражнения, направленные на приобретение навыков использования этих средств;

— задание II уровня содержит более сложные задачи, требующие сочетания типовых алгоритмов и в некоторой степени творческого подхода;

— задание III уровня содержит задачи, близкие к реальным, и рекомендуется для выполнения более подготовленным студентам.

Сказанное относится в основном к работам 5 — 13. Работы 1 — 4 для единообразия также имеют задания трех уровней. Однако в рамках темы работы каждый уровень направлен на приобретение навыков использования какого-либо одного приема или алгоритма.

Наличие заданий трех уровней позволяет в значительной степени индивидуализировать выполнение практикума.

Освоение материала книги рекомендуется в следующей последовательности:

1. Прочитав введение, получить общее представление об ЭВМ.
2. Познакомиться с основными понятиями программирования (глава 1), выполнить приведенные в этой главе упражнения.
3. Выполнить работы 1 — 5.
4. Далее, в зависимости от профиля специальности и объема курса выполнить на выбор все или часть работ 6 — 9.
5. Выполнить работу 10.
6. При достаточном объеме курса выполнить работы 11 — 13.

Лица, освоившие в полном объеме предложенный практикум, будут в достаточной степени подготовлены для самостоятельного решения на ЭВМ задач, возникающих в их практической деятельности.

Практикум может быть использован в качестве учебного пособия при изучении курсов по программированию, для самостоятельного овладения основами алгоритмизации и программирования на бейсике и работой на ЭВМ. Он будет полезен также преподавателям и учащимся средних учебных заведений.

Практикум составлен на основании методики преподавания программирования, успешно используемой в МИСиС при обучении студентов, а также преподавателей и сотрудников в рамках факультета повышения квалификации.

## ВВЕДЕНИЕ. ЧТО ТАКОЕ ЭВМ?

В настоящее время существует огромное разнообразие электронных вычислительных машин (ЭВМ) различных типов и марок. Во введении, не вдаваясь в подробности и не акцентируя внимание на особенностях отдельных ЭВМ, мы попытались сформировать общее представление о том, что такое вычислительная машина.

Такое общее представление будет полезным для тех, кто впервые сталкивается с программированием и вычислительной техникой.

Электронная вычислительная машина представляет собой устройство (точнее, совокупность взаимосвязанных устройств), которое способно выполнять определенный набор элементарных арифметических и логических операций, таких как «сложить два числа», «перемножить два числа», «сравнить два числа» и т. п. Выполнив одну операцию, ЭВМ может автоматически перейти к выполнению следующей и, таким образом, может выполнять длинные цепочки операций без вмешательства человека. ЭВМ работает с высокой скоростью, составляющей (в зависимости от типа ЭВМ) тысячи и миллионы операций в секунду.

Последовательность операций, которую должна выполнить ЭВМ, записанная в доступной для ЭВМ форме, называется программой (программой на машинном языке). То обстоятельство, что вычислительная машина работает по программе, составляет сущность программного принципа управления работой ЭВМ. Процесс разработки программ для ЭВМ называется программированием. Программа на машинном языке является очень детальной, и форма ее представления неудобна для человека. Пользователи ЭВМ в настоящее время не составляют таких программ. Как правило, программы составляются на так называемых алгоритмических языках (см. главу 3), более естественных и удобных для человека.

«Мозгом» ЭВМ, выполняющим операции и управляющим работой всей ЭВМ, является *процессор*. Программа, по которой работает ЭВМ, и данные, которые обрабатываются этой программой, хранятся в памяти ЭВМ. Команды на выполнение каждой операции (из таких команд состоит программа) и данные, над которыми операция должна быть выполнена, по очереди передаются в процессор. Результаты выполнения операции передаются обратно в память. Таким образом, память и процессор при выполнении программы постоянно обмениваются информацией (рис. 1).

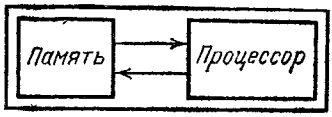


Рис. 1

*Память* построена из элементов, каждый из которых может находиться в одном из двух устойчивых состояний (двоичных запоминающих элементов). Такие элементы наиболее просты в конструктивном отношении, дешевы и надежны. (Простейшие примеры таких элементов: реле замкнуто, разомкнуто; электрическая лампочка горит, не горит.) Одному из двух возможных состояний элемента ставится в соответствие 0, другому — 1. Минимальная единица информации, представленная одним двоичным элементом, носит название *бит* (bynary digit). Чтобы хранить более емкую информацию, двоичные запоминающие элементы объединяются в группы. В современных ЭВМ принято использовать группы по 8 бит, называемые *байтом*. Все байты пронумерованы, номер байта называется его *адресом*. Байты могут объединяться в более крупные единицы памяти (ячейки памяти), используемые для размещения информации. Для каждой ЭВМ определена характерная для нее длина (число байт) ячейки памяти. Такая ячейка характерной для данной ЭВМ длины называется *словом*. Это не исключает возможности использования ячеек памяти другой длины (например, полуслово, двойное слово и т. п.).

Любая информация в ЭВМ, таким образом, должна представляться последовательностью нулей и единиц (в соответствии с состоянием отдельных двоичных элементов ячейки памяти). Если учесть, что 0 и 1 являются цифрами системы счисления по основанию 2 (двоичной системы счисления), то можно такую последовательность нулей и единиц считать двоичным числом. Тогда соответствие между привычным для человека представлением числа в десятичной системе (внешнее представление) и представлением этого числа в ЭВМ (внутреннее представление) может быть установлено правилами перевода чисел из десятичной системы

счисления в двоичную. Перевод чисел из внешнего представления во внутреннее и наоборот осуществляется автоматически самой машиной.

Набором нулей и единиц можно закодировать не только числовую информацию, но и нечисловую, например, буквы. Так, буква А (лат.) кодируется набором 01000001, буква В (лат.) — 01000010 и т. д. Таким образом, в памяти ЭВМ могут быть представлены данные любой природы, закодированные набором 0 и 1. Это позволяет решать на ЭВМ задачи не только вычислительного характера, но и информационного, задачи обработки текстов и т. п.

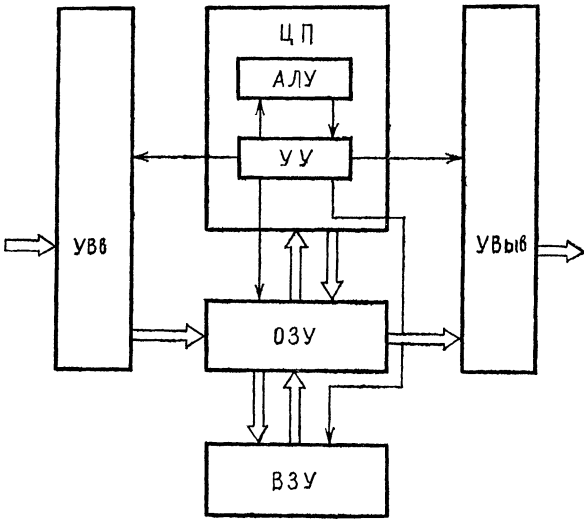



Рис. 2

Память ЭВМ состоит обычно из двух частей: оперативного запоминающего устройства (ОЗУ) и внешних запоминающих устройств (ВЗУ). В ОЗУ находятся выполняемая программа и используемые этой программой данные. Здесь же получаются результаты решения задачи.

Когда машина выключается, все, что находилось в ОЗУ (программа, данные), пропадает (стирается). ВЗУ могут хранить информацию достаточно долго и используются для долговременного хранения программ или частей программ, наборов данных, многократно используемых разными программами, и т. п.

Программа, которая должна быть выполнена, или данные, которые должны быть обработаны, предварительно передаются с ВЗУ в ОЗУ.

В состав каждой ЭВМ входят также устройства ввода (УВв), при помощи которых осуществляется ввод программы и данных в ОЗУ, и устройства вывода (УВыв), при помощи которых результаты выводятся в привычной для человека форме. Ввод информации в ЭВМ чаще всего осуществляется с клавиатуры дисплея. Для вывода используются печатающее устройство, цифровой и графический дисплеи, графопостроитель и т. д.

Устройства ввода и вывода, а также различные виды ВЗУ объединяются общим названием — *периферийные устройства*. Периферийные устройства при необходимости можно подключать к ЭВМ независимо друг от друга.

На рис. 2 приведена упрощенная типовая схема ЭВМ и показаны взаимодействия отдельных устройств (одинарная стрелка — передача управляющих сигналов, двойная стрелка — передача информации). На рис. 2 показаны также основные устройства, из которых состоит процессор (ЦП). Это арифметико-логическое устройство (АЛУ), в котором выполняются операции, и устройство управления (УУ), которое координирует работу всех узлов ЭВМ.

# Глава 1. ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ

## 1. Основные понятия программирования

Введем основные понятия программирования. Их рассмотрение позволит сосредоточить внимание на специфике вычислительной машины как средства выполнения разрабатываемых программ и подготовит читателя для восприятия дальнейшего материала.

*Исполнитель* — это человек или автомат (в частности, им может быть процессор ЭВМ), умеющий выполнять некоторый, вполне определенный конечный набор действий. Приказ на выполнение действия из указанного набора, выраженный каким-либо заранее оговоренным способом, называется *предписанием*, а вся совокупность допустимых приказов — *системой предписаний* исполнителя.

Давая задание исполнителю на выполнение некоторой работы, мы обычно выдаем ему не одно предписание, а некоторую конечную последовательность предписаний, задавая также порядок, в котором эти предписания должны быть выполнены. Такая последовательность предписаний с указанием порядка их выполнения называется *программой*.

Всякое действие производится над некоторыми объектами, и о его результатах можно полностью судить по изменению состояния этих объектов. Один из видов объектов — *переменная*. Понятие переменной является одним из центральных понятий программирования. (Понятие переменной имеет в программировании другой смысл, чем в математике.) Поясним это понятие на примере.

Пример. Требуется найти произведение любых двух натуральных чисел $n$, $m$. Результат обозначить через $k$.

Составим задание для исполнителя, который не умеет выполнять умножение, а умеет лишь складывать. В этом случае решение задачи не может быть выполнено в одно действие, а требует разложения на ряд последовательных действий, т. е. составления программы.

Так как программа должна «работать» для любой пары натуральных чисел, то сами конкретные числа в программе не фигурируют. Вместо чисел употребляются имена, обозначающие изменяемые объекты, которые называются *переменными*. Перед началом вычислений этим переменным должны быть присвоены значения. *Присвоение* — одно из важнейших действий, выполняемых вычислительной машиной (на использование которой в качестве исполнителя мы и ориентируемся в конечном счете).

Переменную можно представить себе как ящик, обозначенный именем, идентифицирующим эту переменную. Присвоение переменной с именем $n$ значения 5 можно представить себе так: положить в ящик, обозначенный $n$, 5 шаров. (Ящик является аналогом ячейки памяти ЭВМ.)

Значение одной переменной можно *переслать* в другую переменную. При этой операции значение пересылаемой переменной не изменяется. Например переслать значение $n$ в переменную $i$, или присвоить переменной $i$ значение $n$, означает задать $i$ такое же значение, которое имеет переменная $n$, т. е. скопировать значение $n$. Если было, допустим, $n=5$, то присвоение $i$ значения $n$ (записывается $i=n$ *) — $i$ присвоить значение $n$) означает, что нужно как бы посмотреть, какое число находится в ячейке памяти $n$, и такое же число «положить» в ячейку памяти $i$. Теперь будет $i=5$, но при этом $n$ сохранило значение ($n=5$).

Часто в программировании используется такая операция присваивания, когда слева и справа используется одна и та же переменная, например, $i=i+1$. Такая запись означает, что сначала должна быть выполнена операция сложения $(i+1)$, а затем полученная сумма присвоена переменной $i$ в качестве ее нового значения. При этом старое значение $i$ пропадает, «стирается». После выполнения этой операции $i$ будет иметь значение на 1 больше, чем перед ее выполнением.

Вернемся к нашей задаче. Чтобы получить произведение $n$ на $m$, используя операцию сложения, нужно просуммировать $m$ слагаемых, равных $n$, т. е. вычислить

$$k=\underbrace{n+n+\ldots+n}_{m \text{ раз}}.$$

Но эта формула — не программа, хотя бы потому, что здесь есть неопределенность (например, многоточие).

---

*) Знак «=» обозначает здесь не равенство, а операцию присвоения.

Решение этой задачи можно представить как последовательность выполнения следующих простых шагов:

Положить $k=0$.

Далее выполнить операцию

$k=k+n$

$m$ раз. При этом после каждого выполнения указанной операции значение $k$ увеличивается на $n$. В итоге в $k$ будет получен результат решения задачи.

Чтобы выполнить операцию требуемое число раз, нужно считать, сколько раз эта операция уже выполнена. Используем для этого вспомогательную переменную $i$. Назовем ее счетчиком. Перед первым прибавлением к $k$ значения $n$ положим $i=1$ и после очередного изменения $k$ значение счетчика $i$ будем менять на 1.

Тогда программа может быть записана так:

1 задать конкретные значения $n$, $m$
2 $k=0$
3 $i=1$
4 $k=k+n$
5 $i=i+1$
6 если $i\leqslant m$ идти к 4. (Повторить выполнение операций, начиная с п. 4.)
7 закончить вычисления

Пояснения к программе.

1. Программа написана на обычном языке человеческого общения с использованием общепринятой математической символики (это так называемый «естественный» язык).

2. Операторы *) 2, 3 задают начальные значения переменных $k$ и $i$. Оператор 4 при каждом своем выполнении увеличивает значение $k$ на $n$. Оператор 5 увеличивает значение счетчика на 1 после того, как выполнено очередное сложение. Оператор 6 проверяет условие $i\leqslant m$, и если оно выполняется, т. е. не все $m$ сложений еще выполнены, то происходит возврат к оператору 4 и повторное выполнение программы, начиная с оператора 4. Как только в процессе выполнения программы условие $i\leqslant m$ не будет выполнено, процесс вычислений заканчивается. Это произойдет, когда будет $i>m$, т. е. все нужные сложения выполнены.

Приведенная программа задает порядок действий, которые могут быть выполнены, когда $n$ и $m$ получат конкретные значения.

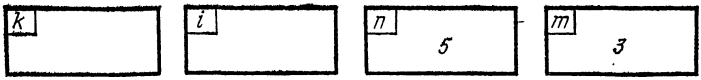
*) В языках программирования предписание о выполнении некоторой операции (например, операции присваивания) называется оператором. Этот термин и будем далее употреблять.

Совокупность значений переменных, которые должны быть заданы перед выполнением программы, называется *исходными данными*.

Упражнение 1. Выполнить программу при $n=5$, $m=3$. Проследить за изменением переменных.

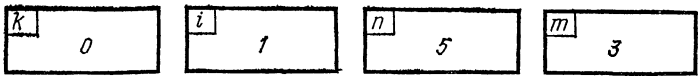
Указание. Для каждой переменной, используемой в программе ($n$, $m$, $k$, $i$), предусмотреть ячейку памяти. При задании значения переменной записать это значение в ячейке памяти с соответствующим именем. После выполнения программы сравнить содержимое ячейки $k$ с правильным ответом (результатом умножения 3 на 5).

Решение. Содержимое ячеек памяти (значения переменных) после выполнения оператора 1.



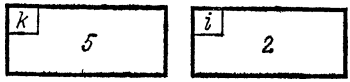

Отсутствие записи означает, что значение переменной не определено.

После выполнения операторов 2, 3.




Содержимое ячеек $n$, $m$ далее не изменяется.

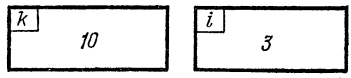
После первого выполнения операторов 4, 5.




Выполнение оператора 6 не изменяет значений переменных, а связано с проверкой условия. Условие здесь задано в виде отношения и вычисляется каждый раз при текущих значениях входящих в него переменных. Результатом вычисления условия является *да*, если условие удовлетворяется для входящих в него переменных, или *нет*, если условие не удовлетворяется. При значении *да* в данной программе будет осуществляться переход к оператору 4, при значении *нет* — переход к следующему по порядку оператору (в данном случае — окончание выполнения программы).

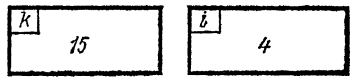
При первом выполнении оператора 6 проверяется условие $2\leqslant 3$. Оно имеет значение *да*, поэтому следующим выполняется оператор с номером 4.

После второго выполнения операторов 4, 5.




При втором выполнении оператора 6 условие $3 \leqslant 3$ имеет значение *да*, и снова выполняется оператор 4.

После третьего выполнения операторов 4, 5.




При третьем выполнении оператора 6 условие $4 \leqslant 3$ не удовлетворяется (имеет значение *нет*), и выполнение программы прекращается.

Результатом является последнее значение переменной $k$, равное 15.

Изменение значений переменных при выполнении программы можно представить в более компактном виде — в виде *трассировочной таблицы*, в которой записываются все значения, последовательно принимаемые изменяемыми переменными программы. Для рассмотренной выше программы изменяются только переменные $k$ и $i$, и трассировочная таблица имеет вид

| $k$ | 0 | 5 | 10 | 15 |
|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 |

Отметим, что в рассмотренной программе после выполнения оператора 6 (если условие выполняется) осуществляется возврат к оператору 4 и повторное выполнение операторов 4—6. Многократно повторяющаяся при выполнении часть программы носит название *цикл*. Цикл является типичной структурой, реализуемой в программах для ЭВМ. Его использование позволяет записать последовательность действий один раз, а выполнять их многократно, после каждого выполнения возвращаясь к началу этой последовательности (цикла) (см. также п. 3).

Упражнение 2. Составить программу на естественном языке для возведения числа $x$ в степень $k$ ($k$ — натуральное число), используя операцию умножения. Результат обозначить через $z$.

Выполнить программу для $x=4$, $k=3$, фиксируя изменение переменных с использованием: а) ячеек памяти; б) трассировочной таблицы.

Р е ш е н и е. Необходимо составить программу для вычисления

$$z=\underbrace{x\cdot x\cdot\ldots\cdot x}_{k\text{ раз}}.$$

Положим вначале

$z=1$,

а далее выполним операцию

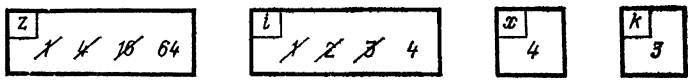
$z=z\cdot x$

$k$ раз. Для подсчета выполненных умножений используем счетчик $i$ (см. также предыдущий пример).

Программа на естественном языке будет иметь вид

1 задать конкретные значения $x$, $k$
2 $z=1$
3 $i=1$
4 $z=z\cdot x$
5 $i=i+1$
6 если $i\leqslant k$ идти к 4
7 закончить вычисления

Предусмотрим четыре ячейки памяти для переменных. При изменении значения переменной старое значение теперь будем зачеркивать и рядом записывать новое:




Последнее значение $z=64$ и является результатом выполнения программы.

Введенные в данном параграфе понятия — исполнитель, предписание, система предписаний, программа, переменная, присваивание, цикл — являются важными понятиями программирования, на которых базируется дальнейшее изложение.

## 2. Понятие алгоритма

*Алгоритмом* называется четкое описание последовательности действий, которые необходимо выполнить для решения задачи. Практически решение любой задачи требует

получения результата по заданным исходным данным. То есть можно сказать, что алгоритм описывает последовательный процесс преобразования исходных данных в результат.

Разработать алгоритм решения задачи означает разбить задачу на последовательно выполняемые *шаги* (этапы), причем результаты выполнения предыдущих этапов могут использоваться при выполнении последующих. При этом должны быть четко указаны как содержание каждого этапа, так и порядок выполнения этапов. Отдельный этап (шаг) алгоритма представляет собой либо другую, более простую задачу, алгоритм решения которой разработан ранее, либо должен быть достаточно простым и понятным без пояснений.

Четко сформулированная последовательность правил, описывающих этот процесс, и является алгоритмом.

Если алгоритм разработан, то его можно вручить для выполнения человеку (и вообще любому исполнителю, в том числе и ЭВМ), не знакомому с решаемой задачей, и, точно следуя правилам алгоритма, этот человек (или другой исполнитель) получит ее решение.

Например, вам предложено выполнить следующую последовательность действий при заданных значениях $a=1$, $b=3$, $c=2$:

1. Вычислить $D=b^2-4ac$.

2. Сравнить D с нулем. Если $D<0$, перейти к 3. В противном случае вычислить $x_1=(-b+\sqrt{D})/2a$, $x_2=(-b-\sqrt{D})/2a$.

3. Прекратить вычисления.

Оказывается, выполнив приведенную последовательность для указанных значений $a$, $b$ и $c$, вы решили квадратное уравнение $x^2+3x+2=0$.

Алгоритм обладает следующими основными свойствами, раскрывающими его определение:

1. Дискретность. Это свойство состоит в том, что алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов). При этом для выполнения каждого шага (этапа) алгоритма требуется некоторый конечный отрезок времени. То есть преобразование исходных данных в результат осуществляется во времени дискретно.

2. Определенность (или детерминированность). Это свойство состоит в том, что каждое правило алгоритма должно быть четким, однозначным и не оставлять

места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

3. Результативность (или конечность). Это свойство состоит в том, что алгоритм должен приводить к решению задачи за конечное число шагов.

4. Массовость. Это свойство состоит в том, что алгоритм решения задачи разрабатывается в общем виде, т. е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется *областью применимости* алгоритма. (В отдельных случаях исходные данные могут отсутствовать.)

Например, приведенный выше алгоритм решения квадратичного уравнения применим для различных наборов коэффициентов $a$, $b$, $c$ ($a \neq 0$).

Чтобы разработать алгоритм, нужно хорошо представить себе ход решения задачи. При этом полезно решить задачу самому (на бумаге) для каких-либо наборов данных, не требующих громоздких вычислений, запоминая выполняемые действия, так, чтобы далее эти действия формализовать, т. е. записать в виде последовательности четких правил.

Понятия алгоритма и программы разграничены не очень четко. Обычно программой называют окончательный вариант алгоритма решения задачи, ориентированный на конкретного исполнителя.

Этап, результатом которого является разработка алгоритма решения задачи, часто называют *алгоритмизацией*, понимая под этим сведение задачи к последовательности этапов, выполняемых последовательно друг за другом. В широком смысле алгоритмизация включает и выбор метода решения задачи, а также формы представления исходной информации с учетом специфики ЭВМ.

Разработанный алгоритм можно зафиксировать несколькими способами, например:

— на естественном языке (предыдущий пример);

— в виде схемы (см. ниже);

— на специальном языке для записи алгоритмов (алгоритмическом языке).

Запись алгоритма на естественном языке. Хотя естественный язык не требует детальных разъяснений и полной формализации, сформулируем не-

которые правила, которые облегчат в дальнейшем переход к алгоритмическому языку.

Введем следующие обозначения типичных этапов алгоритма:

1. Этап обработки (вычисления):

*v = выражение*

(*v* — переменная, *выражение* задает правило вычисления значения, которое далее будет присвоено переменной *v*. Это может быть, например, знакомое нам алгебраическое (арифметическое) выражение).

2. Проверка условия:

если *условие* идти к *N*.

Если условие удовлетворяется, выполняется переход к этапу *N*. Если условие не удовлетворяется, то условимся, что осуществляется переход к следующему по порядку этапу:

3. Конец вычислений:

закончить вычисления.

4. Переход к этапу с номером *N*:

идти к *N*.

Изображение алгоритма в виде схемы. *Схемой* называется наглядное графическое изображение алгоритма, когда отдельные действия (этапы) алгоритма изображаются при помощи различных геометрических фигур (*блоков*) *), а связи между этапами (последовательность выполнения этапов) указываются при помощи стрелок **), соединяющих эти фигуры.

Существует Государственный стандарт (ГОСТ), определяющий правила выполнения схем и обозначения для отдельных операций процесса обработки данных. Далее приводятся обозначения некоторых, наиболее часто употребляемых операций. Полный перечень установленных правил и условных обозначений см. ГОСТ 19.002-80 и ГОСТ 19.003-80.

---

*) Термин *блок* употребляется здесь и далее вместо термина *символ*, используемого ГОСТом, в связи с употреблением термина *символ* в программировании в другом смысле.

**) Направления сверху вниз и слева направо принимаются за основные и могут (если не имеют изломов) не обозначаться стрелками.

Типичные действия алгоритма изображаются следующими геометрическими фигурами.

Этап обработки (вычисления) изображается прямоугольником (рис. 1.1*а*). Внутри прямоугольника записывается содержание этого этапа.

Проверка условия изображается ромбом. Условие записывается внутри ромба. В результате проверки условия

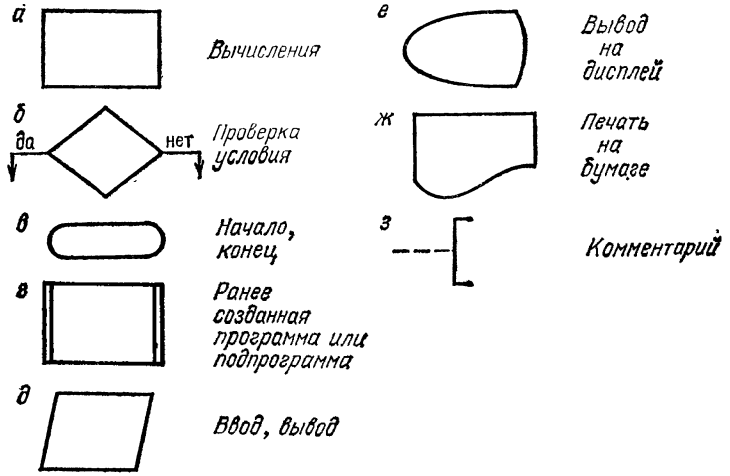



Рис. 1.1

осуществляется выбор одного из двух возможных путей вычислительного процесса (рис. 1.1*б*).

Если условие выполняется, т. е. имеет значение *да*, то следующим выполняется этап по стрелке *да*. Если условие не выполняется, то осуществляется переход по стрелке *нет*.

Начало и конец вычислительного процесса изображаются фигурой, показанной на рис. 1.1*в*. Внутри нее записывается начало или конец.

Ранее созданные и отдельно описанные алгоритмы и программы (подпрограммы; см. п.9 главы 3) изображаются фигурой на рис. 1.1*г*. Внутри нее указывается имя подпрограммы и параметры, при которых подпрограмма должна быть выполнена.

Ввод исходных данных и вывод результатов изображаются параллелограммом (рис. 1.1*д*). Внутри него пишется слово «ввод» или «печать» и перечисляются переменные, подлежащие вводу или выводу.

Параллелограммом обозначается ввод/вывод (в/в) вообще. Если ввод или вывод осуществляется с использованием конкретных устройств, то блоки в/в изображаются при помощи специальных фигур. Так, в/в с использованием дисплея, вывод на печатающее устройство изображаются соответственно фигурами, представленными на рис. 1.1*е*, 1.1*ж*. См. также ГОСТ 19.003-80.

Комментарий используется в тех случаях, когда пояснение не помещается внутри блока (рис. 1.1*з*).

В качестве примера изобразим в виде схемы алгоритм вычисления произведения двух натуральных чисел $n$ и $m$ с использованием операции сложения, записанный ранее на естественном языке, добавив этапы ввода/вывода (рис. 1.2).

Схема позволяет наглядно представить структуру алгоритма. В частности, на схеме хорошо видны циклы: это последовательности этапов, после последнего из которых осуществляется возврат к началу последовательности (на схеме это — замкнутые участки). Схемы обычно используются для изображения промежуточных вариантов алгоритма. Окончательный вариант, предназначенный для исполнителя-ЭВМ (программа), должен быть записан на алгоритмическом языке.

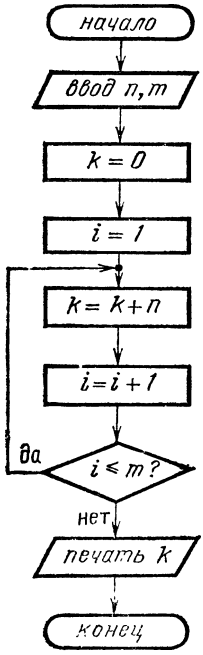


Рис. 1.2

В настоящее время существует технология разработки программ без использования схем. Однако независимо от этого на начальном этапе изучения программирования использование схем при разработке алгоритма целесообразно. Использование схем обеспечивает приобретение прочных навыков разработки алгоритмов с использованием типовых структур алгоритмов (см. п. 3), являющихся основой так называемого структурного подхода, особенно плодотворного при постановке и решении на ЭВМ сложных задач.

ГОСТом, помимо типов фигур, предписываются также определенные размеры их сторон, одинаковые размеры блоков. Этих требований необходимо придерживаться при оформлении окончательной документации. На промежуточных этапах разработки алгоритма придерживаться этих требований ГОСТа не обязательно.

Понятие об алгоритмических языках. Алгоритмические языки близки к естественному языку. Именно такая цель ставилась при их разработке. Однако правила построения конструкций в алгоритмическом языке более «жесткие». Это означает, что алгоритмические языки допускают меньшее разнообразие для описания действий алгоритма, чем естественный язык и привычная математическая символика, и машина однозначно понимает любую конструкцию языка. Например, для умножения двух переменных $a$ и $b$ общепринятая математическая символика допускает несколько возможных форм записи: 1) $ab$; 2) $a \times b$; 3) $a \cdot b$ и т. п. А на алгоритмическом языке, например на языке бейсик, эту операцию можно записать только единственным образом как A*B. Небольшие ошибки или описки, допускаемые в предложениях естественного языка и не искажающие, на наш взгляд, смысла, совершенно недопустимы в алгоритмическим языке, где каждый символ и его место в конструкции имеют строго фиксированное значение.

Так как программа на алгоритмическом языке предназначена для выполнения на ЭВМ, то в алгоритмическом языке, помимо средств, описывающих действия алгоритма, обязательно должны быть также определены средства, позволяющие вводить исходные данные в ЭВМ и выводить результаты выполнения программы.

Программа, составленная на алгоритмическом языке, не может быть непосредственно выполнена ЭВМ, так как ЭВМ умеет выполнять только последовательность элементарных операций, а в программе на алгоритмическом языке в одном выражении может, например, содержаться несколько операций, и форма записи такой программы понятна человеку, но недоступна ЭВМ. Поэтому необходимо какое-то промежуточное звено, которое выполняло бы работу по расчленению отдельных действий программы и записи их на машинном языке. Работа эта несложная, но требует скрупулезного внимания и педантичности. К такой работе больше всего и приспособлена ЭВМ. Перевод программы с алгоритмического языка на машинный осуществляется ЭВМ с помощью специальной программы, которая носит название *транслятор*. В программе-трансляторе «заложены» все правила алгоритмического языка и способы преобразования различных его конструкций на машинный язык. Именно поэтому при составлении программы на алгоритмическом языке нужно скрупулезно придерживаться правил этого языка, иначе ЭВМ вас не поймет или поймет

неправильно. Наиболее распространенными алгоритмическими языками в настоящее время являются бейсик, фортран, ПЛ/1, паскаль.

### 3. Основные структуры алгоритмов. Понятие о структурном подходе к разработке алгоритмов

Основные структуры алгоритмов — это ограниченный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий.

Приводимые ниже структуры рекомендуются при использовании так называемого структурного подхода к разработке алгоритмов и программ. Структурный подход предполагает использование только нескольких основных структур, комбинация которых дает все многообразие алгоритмов и программ.

К основным структурам относятся:

1. Следование. Последовательное размещение блоков и групп блоков. В программе реализуется последовательным размещением операторов.

2. Цикл *До* (рис. 1.3). Применяется при необходимости выполнить какие-либо вычисления несколько раз до выполнения некоторого условия. Особенность этого цикла в том, что он всегда выполняется хотя бы один раз,

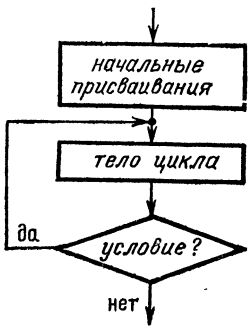


Рис. 1.3

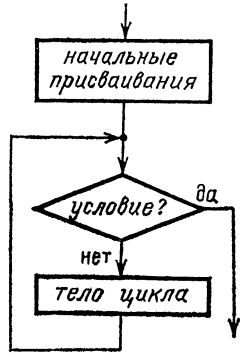


Рис. 1.4

так как первая проверка условия выхода из цикла происходит после того, как тело цикла выполнено. *Тело цикла* — та последовательность действий, которая выполняется многократно (в цикле). *Начальные присвоения* — задание начальных значений тем переменным, которые используются в теле цикла.

На естественном языке циклу *До* соответствует последовательность операторов:

1 Операторы начальных присвоений
2 Операторы тела цикла
3 Если *условие* идти к 2

Цикл, использованный в приведенном выше примере (стр. 15), это цикл *До*.

3. Цикл *Пока* (рис. 1.4). Цикл *Пока* отличается от цикла *До* тем, что проверка условия проводится до выполнения тела цикла, и если при первой проверке условие выхода из цикла выполняется, то тело цикла не выполняется ни разу.

На естественном языке циклу *Пока* соответствует последовательность операторов:

1 Операторы начальных присвоений
2 Если *условие* идти к 5
3 Операторы тела цикла
4 Идти к 2
5 . . .

4. Разветвление (рис. 1.5). Применяется, когда в зависимости от условия нужно выполнить либо одно,

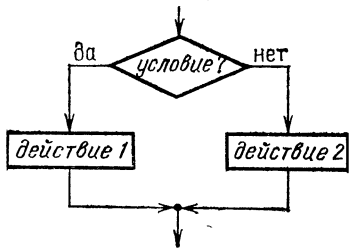



Рис. 1.5

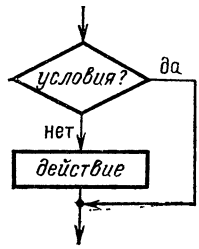



Рис. 1.6

либо другое действие. Действие 1 или действие 2 может в свою очередь содержать несколько этапов.

На естественном языке разветвлению соответствует последовательность операторов:

1 Если *условие* идти к 4
2 Операторы действия 2
3 Идти к 5
4 Операторы действия 1
5 . . .

5. Обход (рис. 1.6). Частный случай разветвления, когда одна ветвь не содержит никаких действий.

На естественном языке обходу соответствует последовательность операторов:

1 Если *условие* идти к 3
2 Операторы действия
3 . . .

6. Множественный выбор (рис. 1.7). Является обобщением разветвления, когда в зависимости от значения переменной ($I$) выполняется одно из нескольких действий. При $I=1$ выполняется действие $S1$, при $I=2$ — действие $S2$ и т. д.

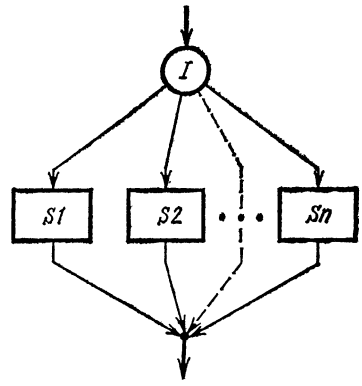


Рис. 1.7

Особенностью всех приведенных структур является то, что они имеют один вход и один выход, и их можно соединять друг с другом в любой последовательности. В частности, каждая структура может содержать любую другую структуру в качестве одного из блоков (см., например, упражнения 2, 4, 5).

Обычно при составлении схемы блоки размещаются друг под другом в порядке их выполнения. Возврат назад осуществляется только на циклах. Это дает простую и наглядную структуру алгоритма, по которой далее легко составить программу.

Одним из приемов разработки алгоритма решения более сложных задач является метод пошаговой детализации, когда первоначально продумывается и фиксируется общая структура алгоритма без детальной проработки отдельных его частей, но при этом также используются лишь основные структуры алгоритмов. Блоки, требующие дальнейшей детализации, обозначаются пунктирной линией. Далее прорабатываются (детализируются) отдельные блоки, не детализированные на предыдущем шаге. То есть на каждом шаге разработки уточняется реализация фрагмента алгоритма (или программы), и, таким образом, на каждом шаге мы имеем дело с более простой задачей. Полностью закончив детализацию всех блоков, мы получим решение всей задачи в целом. Описанный метод пошаговой детализации называется также программированием сверху вниз.

В некоторых случаях стремление во что бы то ни стало остаться в рамках структурного подхода приводит к необоснованному усложнению программы и потере ее наглядности и естественности. Если учесть, что структурное программирование имеет целью не подчинить программы каким-то правилам, а сделать их более удобными для восприятия, то в ряде случаев оказывается целесообразным отдать предпочтение ясности и естественности программы. Такого рода отступления содержит, например, программа 2.4.

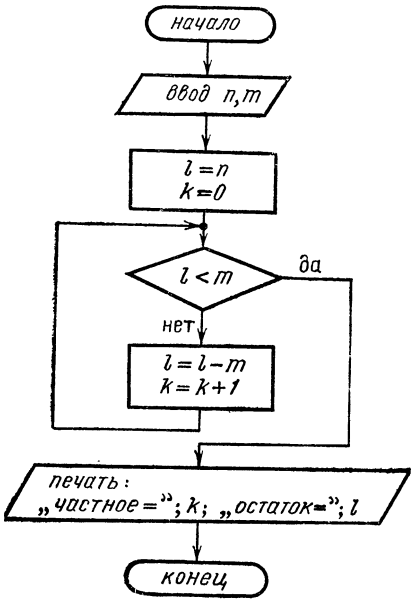


Рис. 1.8

Упражнения. Составить схему и программу на естественном языке для решения задач, используя типовые структуры алгоритмов и их простые сочетания.

Выполнить составленную программу при заданных исходных данных, фиксируя значения изменяемых переменных в ячейках памяти и с помощью трассировочной таблицы.

Для каждого из заданных наборов данных программу нужно выполнять заново, начиная с первого оператора.

**1.** Разделить натуральное число $n$ на натуральное число $m$, получить в качестве результата частное от деления $k$ и остаток $l$, т. е. представить число $n$ в виде

$$n=k\cdot m+l,$$

где $l<m$, $k$ — целое. Операцию деления не использовать.

Решение. Операцию деления можно представить как последовательность вычитаний делителя из делимого. Тогда количество вычитаний будет частным от деления $k$. При этом последовательные вычитания нужно проводить до тех пор, пока результат вычитания не станет меньше делителя. Тогда эта последняя разность и будет остатком от деления $l$.

Далее приводится схема алгоритма (рис. 1.8) и программа на естественном языке для решения этой задачи.

Программа «Деление двух натуральных чисел».
Исходные данные: $n$, $m$ — натуральные числа.
Результат: $k$, $l$.

1 задать значения $n$, $m$
2 $l=n$
3 $k=0$
4 если $l<m$ идти к 8
5 $l=l-m$
6 $k=k+1$
7 идти к 4
8 закончить вычисления

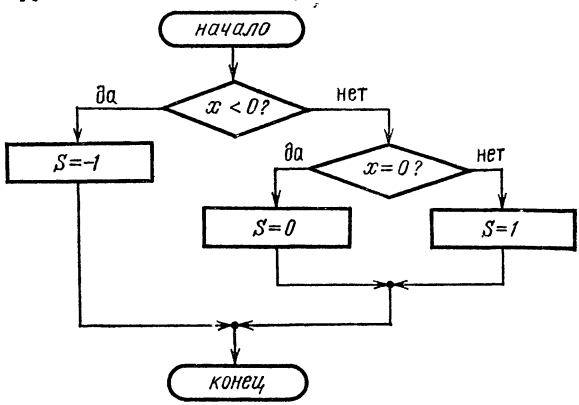
Обратите внимание на то, что проверка условия выхода из цикла проводится до входа в цикл (цикл *Пока*) и при $n<m$ остается $k=0$.

Выполнить программу при: 1) $n=15$, $m=5$; 2) $n=14$, $m=5$; 3) $n=3$, $m=5$.

**2.** Задано число $x$. Вычислить функцию

$$S=\begin{cases} -1, & \text{если } x<0, \\ 0, & \text{если } x=0, \\ 1, & \text{если } x>0. \end{cases}$$

(Это — функция знака числа.)




1.9

Решение. Схема алгоритма приводится на рис. **1.9.** Алгоритм представляет собой разветвление, содержащее в одной из ветвей разветвление.

Программа «Вычисление знака числа».
Исходные данные: $x$ — любое число.

Результат: $S$.

1 задать значение $x$
2 если $x<0$ идти к 8
3 если $x=0$ идти к 6
4 $S=1$
5 идти к 9
6 $S=0$
7 идти к 9
8 $S=-1$
9 закончить вычисления

Обратите внимание на использование оператора «идти к $N$» (операторы 5, 7) в этой программе. Он позволяет обойти одну из ветвей разветвления, если выполнена другая.

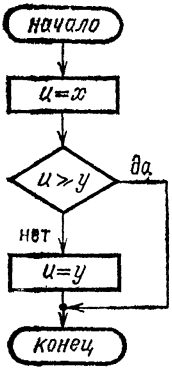


Рис. 1.10

Выполнить программу при: 1) $x=-6$; 2) $x=0$; 3) $x=12$.

3. Выбрать максимальное из двух чисел $x$, $y$ и присвоить его значение переменной $u$.

Р е ш е н и е. Для решения этой простой задачи будем просматривать числа по очереди. Пока мы «видим» только первое число $x$, будем считать его максимальным и присвоим $u$ значение $x$ ($u=x$). Затем $u$ сравним со вторым числом $y$, и если окажется, что значение $u$ меньше $y$, то $u$ присвоим новое значение, иначе $u$ оставим без изменения. Схема алгоритма приведена на рис. 1.10.

П р о г р а м м а «Нахождение максимального из двух чисел».

Исходные данные: $x$, $y$ — любые числа.

Результат: $u$ — максимальное из $x$, $y$.

1 задать значения $x$, $y$
2 $u=x$
3 если $u\geqslant y$ идти к 5
4 $u=y$
5 закончить вычисления

Следует обратить внимание на характерный прием, используемый в приведенной программе. Значение переменной $u$ в результате выполнения программы будет равно либо $x$, либо $y$. Одно из этих значений и присваивается переменной

$u$ в начале программы ($u=x$). Далее, переменная $u$ изменяет свое значение, только если $x<y$. Если же $x\geqslant y$, то значение $u$ сохраняется без изменений.

Идея, лежащая в основе этого алгоритма, позволяет единообразными действиями найти максимальное из трех и т. д. чисел.

Программа «Нахождение максимального из трех чисел».

Исходные данные: $x$, $y$, $z$ — любые числа.

Результат: $u$ — максимальное из $x$, $y$, $z$.

1 задать значения $x$, $y$, $z$
2 $u=x$
3 если $u\geqslant y$ идти к 5
4 $u=y$
5 если $u\geqslant z$ идти к 7
6 $u=z$
7 закончить вычисления

Выполнить программу при: 1) $x=8$, $y=12$, $z=6$; 2) $x=5$, $y=3$, $z=4$; 3) $x=6$, $y=2$, $z=13$.

**4.** В переменную $x$ по очереди помещаются (вводятся) 10 чисел. В переменной $P$ получить максимальное из этих чисел.

Решение. Если до входа в цикл переменной $P$ присвоить значение первого введенного числа $x$, то в цикле для очередного значения $x$ нужно проверять условие $P\geqslant x$, и если оно не выполняется, то заменить старое значение $P$ на новое, равное текущему значению $x$. После чего в $x$ вводится следующее значение. Алгоритм представляет собой цикл *До* с обходом внутри цикла. Схема алгоритма представлена на рис. 1.11 в двух вариантах.

В первом варианте (рис. 1.11*а*) имеются два блока *ввод x*. Первый из них выполняется только один раз для ввода первого числа, которое используется для задания первоначального значения $P$, и если остальные числа окажутся не больше первого, то $P$ сохранит это первое значение. Второй блок *ввод x* выполняется многократно (в цикле), и каждый раз осуществляется ввод в переменную $x$ (задание переменной $x$ значения) очередного из заданных десяти чисел.

Второй вариант (рис. 1.11*б*) составлен для случая поиска максимального из десяти положительных чисел. Если известно, что все вводимые числа будут больше какого-либо заданного значения (в данном случае больше нуля), то первоначально $P$ можно задать таким, чтобы первое введенное

значение $x$ его обязательно превзошло (например, вначале присвоить $P$ любое отрицательное число). Тогда ввод первого значения $x$ можно также осуществлять в цикле.

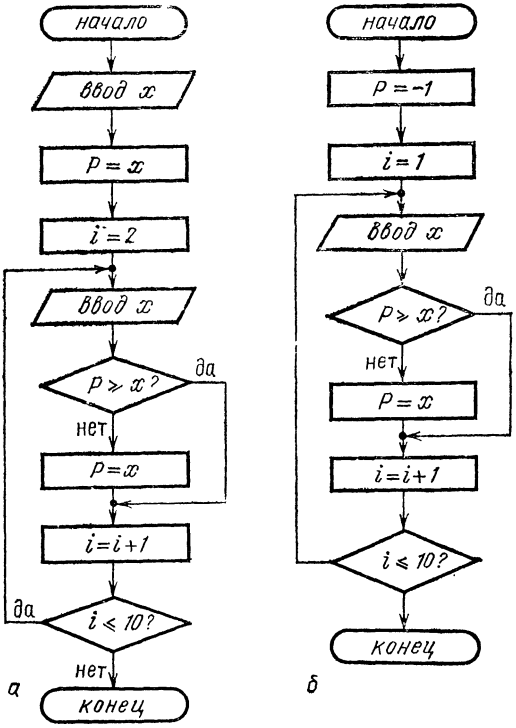



Рис. 1.11

Программа «Нахождение максимального из десяти положительных чисел».

1 $P=-1$
2 $i=1$
3 задать очередное значение $x$
4 если $P \geqslant x$ идти к 6
5 $P=x$
6 $i=i+1$
7 если $i \leqslant 10$ идти к 3
8 закончить вычисления

Выполнить программу для следующего набора чисел 3, 18, 24, 8, 11, 6, 12, 19, 44, 23.

**5.** Задано $n$ троек чисел $a$, $b$, $c$. Вводя их по очереди и интерпретируя как длины сторон треугольника, определить, сколько троек может быть использовано для построения треугольника (числа $a$, $b$, $c$ при вводе расположены в порядке возрастания $a \leqslant b \leqslant c$). Результат получить в переменной $k$.

У к а з а н и е. По трем сторонам с длинами $a$, $b$, $c$ ($a \leqslant b \leqslant c$) можно построить треугольник, если $c < a+b$.

Алгоритм решения задачи представлен на рис. 1.12.

Для каждой тройки чисел проверяется условие $c \geqslant a+b$, и если оно не выполняется (треугольник может быть построен), то значение $k$ увеличивается на 1. До входа в цикл, пока ни одна тройка чисел не введена и не проверена на возможность построения треугольника, $k$ полагается равным 0.

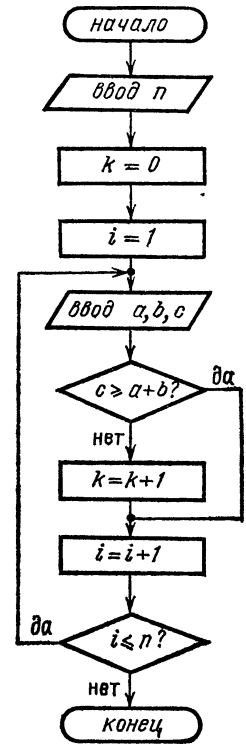


Рис. 1.12

П р о г р а м м а «Определение числа треугольников».

1 задать значение $n$
2 $k=0$
3 $i=1$
4 задать очередной набор $a$, $b$, $c$
5 если $c \geqslant a+b$ идти к 7
6 $k=k+1$
7 $i=i+1$
8. если $i \leqslant n$ идти к 4
9 закончить вычисления

Выполнить программу для пяти ($n=5$) троек чисел: 1) 6, 12, 3; 2) 5, 2, 2; 3) 11, 6, 2; 4) 1, 6, 3; 5) 4, 6, 2.

Задания 6 — 25 выполнить самостоятельно.

**6.** Вычислить $z=x^k$, используя операцию умножения, т. е. представляя $z$ как

$$z = \underbrace{x \cdot x \cdot \ldots \cdot x}_{k \text{ раз}}.$$

У к а з а н и е. В цикле выполнять $z=z \cdot x$, $i=i+1$ ($i$ — счетчик). До входа в цикл положить $z=1$, $i=1$.

Исходные данные: $x=3$, $k=6$.

7. Вычислить $S=\sum_{i=1}^{n} i$.

Указание. В цикле выполнять $S=S+i$, $i=i+1$. До входа в цикл положить $S=0$, $i=1$.

Исходные данные: $n=8$.

8. Вычислить $z=\sum_{i=m}^{n} i$.

См. указание к задаче 7.

Исходные данные: $m=3$, $n=6$.

9. Вычислить $P=\sum_{i=m}^{n} i^2$.

См. указание к задаче 7.

Исходные данные: $m=4$, $n=10$.

10. Вычислить $F=1\cdot 2\cdot \ldots \cdot m=m!$.

Указание. В цикле выполнять $F=F\cdot i$, $i=i+1$. До входа в цикл задать $F=1$, $i=1$.

Исходные данные: $m=6$.

11. Вычислить $F=m\cdot(m+1)\cdot \ldots \cdot n$, $m<n$.

См. указание к задаче 10.

Исходные данные: $m=5$, $n=9$.

12. Заданы два числа $x$, $y$. Переменной $m$ присвоить значение 1, если $x<y$, и $-1$, если $x\geqslant y$.

Исходные данные: 1) $x=2$, $y=3$; 2) $x=4$, $y=4$; 3) $x=3$, $y=2$.

13. Заданы два числа $k$, $m$. Переменной $n$ присвоить значение 1, если $k<m$, 0, если $k=m$, и $-1$, если $k>m$.

Исходные данные: 1) $k=3$, $m=3$; 2) $k=8$, $m=4$; 3) $k=3$, $m=6$.

14. Задано число $p$. Если $p<0$, то значение $y$ вычислить по формуле $y=p^2$; если $p\geqslant 0$ — по формуле $y=p^3$.

Исходные данные: 1) $p=2$; 2) $p=-3$.

15. Заданы числа $p$, $x$, $y$. Если $p<0$, то вычислить $t$ как сумму $x$ и $y$; если $p\geqslant 0$, то как разность $x$ и $y$.

Исходные данные: 1) $p=-6$, $x=4$, $y=2$; 2) $p=3$, $x=4$, $y=2$.

16. Найти сумму десяти произвольных чисел. Результат обозначить через $S$.

Указание. Каждое число вводить по очереди в переменную $x$. В цикле осуществлять задание (ввод) очередного значения $x$ и выполнять операции $S=S+x$, $i=i+1$ ($i$ — счетчик).

Исходные данные: набор из десяти чисел: 7, 6, 1, 12, 14, 11, 3, 1, 2, 5.

**17.** Найти сумму $n$ произвольных чисел. Результат обозначить через $S$.

Указание. Задать (ввести) значение $n$. Далее см. указание к задаче 16.

Исходные данные: $n=4$, набор из четырех чисел: 12, 2, 3, 7.

**18.** Найти произведение $n$ произвольных чисел.

См. указание к задачам 10, 16, 17.

Исходные данные: $n=4$, набор из четырех чисел: 12, 2, 3, 5.

**19.** Найти минимальное из $n$ отрицательных чисел. Результат получить в переменной $P$.

Указание. Ввести (задать) значение $n$. Далее см. задачу 4.

Исходные данные: $n=4$, набор из четырех чисел: —6, —2, —8, —12.

**20.** Найти максимальное из $n$ положительных чисел.

См. указание к задаче 19.

Исходные данные: $n=4$, набор из четырех чисел: 2, 6, 1, 3.

**21.** Заданы три числа $x$, $y$, $z$. Если $x<0$, то $P$ задать как максимальное из $y$, $z$. Если $x\geqslant 0$, то $P$ задать как минимальное из $y$, $z$.

Указание. См. задачу 3.

Исходные данные: 1) $x=-6$, $y=5$, $z=4$; 2) $x=-3$, $y=7$, $z=3$; 3) $x=12$, $y=3$, $z=8$; 4) $x=1$, $y=10$, $z=6$.

**22.** Заданы два числа $x$, $y$. Если их сумма положительна, то $P$ задать как $x^2+y^2$; если отрицательна или равна нулю, то $P$ задать как $(x+y)^2$.

Исходные данные: 1) $x=-4$, $y=7$; 2) $x=4$, $y=8$.

**23.** Заданы три числа $x$, $y$, $z$. Если $x+y>z$, положить $S=x+y+z$. Если $x+y\leqslant z$, то $S=x+y-z$.

Исходные данные: 1) $x=4$, $y=3$; $z=8$; 2) $x=7$, $y=1$, $z=3$.

**24.** Ввести $n$ чисел. Определить, сколько среди них положительных. Результат получить в переменной $k$.

Указание. Числа вводить (задавать значения) в переменную $x$ по очереди в цикле. Для каждого введенного числа проверять условие $x\leqslant 0$, и если оно не удовлетворяется, то выполнять операцию $k=k+1$. До входа в цикл положить $k=0$ (см. также задачу 4).

Исходные данные: $n=4$, набор из четырех чисел: 5, —7, 6, —2.

**25.** Ввести $n$ чисел. Определить, сколько среди них превосходит первое число.

У к а з а н и е. Первое число ввести до входа в цикл в переменную $P$. Остальные числа вводить в цикле в переменную $x$ и каждое введенное число сравнивать с $P$. См. также указание к задаче 24.

Исходные данные: 1) $n=5$, набор из пяти чисел: 7, 12, 6, 4, 8; 2) $n=4$, набор из четырех чисел: 6, 4, 5, 3.

# Глава 2. ОСНОВНЫЕ ПРИЕМЫ ПРОГРАММИРОВАНИЯ

В этой главе рассматриваются основные приемы разработки и проверки правильности программ, применение которых необходимо при решении на ЭВМ любых даже самых простых задач. Изучать эту главу целесообразно параллельно с самостоятельным выполнением работ, приведенных в главе 4.

В качестве примеров здесь приводятся программы на языке бейсик. Бейсик очень похож на «естественный» язык (см. п. 2 главы 1), но в качестве служебных слов в бейсике вместо русских используются английские слова: IF — *если*, GO TO — *идти к*, STOP — *закончить вычисления*, INPUT— *задать (ввести) значения переменных*. Будет использоваться также оператор PRINT — *печатать* — для вывода результатов вычислений на экран дисплея. Полное описание языка бейсик дается в главе 3.

## 1. Организация циклов

Изучение раздела рекомендуется после выполнения работы 1 (глава 4).

Как уже отмечалось (см. п. 3 главы 1), существуют два основных вида цикла — *До* и *Пока*. Рассмотрим их подробнее.

На бейсике циклу *До* (рис. 1.3) соответствует следующая структура программы:

```
1 Операторы начальных присваиваний
2 Тело цикла
3 IF условие GO TO 2
```

Циклу *Пока* (рис. 1.4) соответствует структура программы:

```
1 Операторы начальных присваиваний
2 IF условие GO TO 5
3 Тело цикла
4 GO TO 2
5 . . .
```

Рассмотрим программу *)

```
10 P=0
20 PRINT P,P*P
30 P=P+1
40 IF P<=10 GO TO 20
```

(* — знак умножения, <= употребляется вместо $\leqslant$). При ее выполнении на экран дисплея выводятся числа от 0 до 10 и их квадраты.

Это пример цикла *До*. Здесь операторы 20, 30 образуют тело цикла. Собственно телом цикла является оператор 20. Оператор 30 относится к операторам, организующим цикл (наряду с операторами 10 и 40). Оператор 40 — оператор условного перехода — осуществляет после каждого прохода цикла проверку условия выхода из цикла (вернее сказать — условия продолжения цикла). Оператор 10 осуществляет начальные присваивания и обеспечивает возможность выполнения цикла при первом его прохождении. В цикле — многократно — выполняются операторы 20—40.

Для организации цикла в этой программе используется переменная P, которая называется *управляющей переменной цикла*. Для организации цикла нужно знать начальное значение управляющей переменной $P_{нач}$, конечное значение управляющей переменной $P_{кон}$ и шаг ее изменения H.

Значения $P_{нач}$, $P_{кон}$, H назовем *параметрами цикла*. При этом количество повторений цикла определяется по формуле

$$\frac{P_{кон}-P_{нач}}{H}+1.$$

Это значение можно использовать для контроля правильности организации цикла.

Перед тем как писать программу, содержащую цикл, целесообразно заполнить таблицу такого вида:

Таблица 2.1. **Управляющая переменная и параметры цикла**

| Управляющая переменная цикла | Начальное значение | Конечное значение | Шаг изменения | Количество повторений цикла |
|---|---|---|---|---|
| P | 0 | 10 | 1 | 11 |

*) В программе на бейсике принято нумеровать операторы, начиная с 10, с шагом 10.

(Таблица заполнена для управляющей переменной P, использованной в предыдущей программе.)

Дополним рассмотренную программу.

1. Предусмотрим печать заголовка таблицы оператором

PRINT «ЧИСЛО», «КВАДРАТ ЧИСЛА»

Этот оператор должен быть расположен до входа в цикл.

2. Предусмотрим вывод пустой строки после заголовка оператором

PRINT

(Этот оператор называется оператором PRINT без списка.)

3. В конце программы разместим оператор STOP, который прекращает выполнение программы, когда будет P>=10, и перенумеруем строки программы стандартным образом, изменив также номер строки в операторе условного перехода.

Тогда программа будет иметь вид

```
10 PRINT "ЧИСЛО","КВАДРАТ ЧИСЛА"
20 PRINT
30 P=0
40 PRINT P,P*P
50 P=P+1
60 IF P<=10 GO TO 40
70 STOP
```

При выполнении этой программы на экране будут высвечиваться результаты:

```
ЧИСЛО          КВАДРАТ ЧИСЛА

 0              0
 1              1
 2              4
 3              9
 4              16
 5              25
 6              36
 7              49
 8              64
 9              81
 10             100

STOP AT LINE 70
```

Сообщение

STOP AT LINE 70 (останов в строке 70)

появляется на экране при выполнении оператора STOP.

Как самому составить программу решения задачи, требующей использования циклов?

Рассмотрим порядок разработки программы на примере.

Пример. Составить таблицу перевода миль в километры для расстояний от 5 до 75 миль с шагом 5. 1 миля составляет 1,609 км.

С чего начать разработку программы?

1. Сначала нужно решить, в каком виде мы хотим получить результат. Допустим, что это должна быть таблица с заголовком, например, такого вида:

| Мили | Км |
|---|---|
| 5 | 8.045 |
| 10 | 16.09 |
| . . . | |
| 75 | 120.675 |

2. Выделить повторяющиеся действия.

Для рассматриваемой задачи для 5 миль, для 10, для 15 и т. д. нужно печатать расстояние в милях и соответствующее ему расстояние в километрах.

3. Продумать организацию цикла для выполнения повторяющихся действий требуемое число раз.

Для организации цикла необходимо выбрать управляющую переменную. Для нашего примера это может быть количество миль. Обозначим управляющую переменную через M и заполним таблицу 2.2.

Таблица 2.2

| Управляющая переменная цикла | Начальное значение | Конечное значение | Шаг изменения | Количество повторений цикла |
|---|---|---|---|---|
| M | 5 | 75 | 5 | 15 |

Для вывода заголовка таблицы по аналогии с предыдущей программой используем два оператора, один из которых выведет на экран заголовок, а второй обеспечит вывод пустой строки перед печатью самих результатов.

Таким образом, программа может быть составлена в виде

```
10 PRINT "МИЛИ","КМ"
20 PRINT
30 M=5
40 PRINT M,1.609*M
50 M=M+5
60 IF M<=75 GO TO 40
70 STOP
```

Пояснения к программе. Оператор 10 печатает заголовок.

Оператор 20 выводит пустую строку.

Оператор 30 устанавливает начальное значение управляющей переменной цикла.

Оператор 40 составляет тело цикла и при каждом изменении переменной M выводит на экран ее значение (количество миль) и его эквивалент в километрах.

Оператор 50 увеличивает текущее значение управляющей переменной на шаг, равный 5.

Оператор 60 осуществляет проверку, пройдено ли последнее значение.

Оператор 70 прекращает выполнение программы.

В результате выполнения этой программы на экране появится:

```
 МИЛИ          КМ

 5             8.045
 10            16.09
 15            24.135
 20            32.18
 25            40.225
 30            48.27
 35            56.315
 40            64.36
 45            72.405
 50            80.45
 55            88.495
 60            96.54
 65            104.585
 70            112.63
 75            120.675

STOP AT LINE 70
```

В бейсике имеются операторы цикла FOR и NEXT (FOR — *для*, NEXT — *следующий*), которые обеспечивают многократное выполнение операторов, расположенных между ними и образующих тело цикла.

С использованием операторов FOR, NEXT предыдущая программа будет иметь вид

```
10 PRINT "МИЛИ","КМ"
20 PRINT
30 FOR M=5 TO 75 STEP 5
40 PRINT M,1.609*M
50 NEXT M
60 STOP
```

В операторе FOR используются еще два служебных слова: TO — *до*, STEP — *шаг*.

Типовые алгоритмы циклической структуры и задачи для самостоятельного выполнения, требующие использования циклов, содержатся в работе 2 (глава 4). Там же приводится детальная типовая структура цикла, организуемого при помощи управляющей переменной цикла (рис. 4.1).

## 2. Типы ошибок в программе. Исправление ошибок

Если программа только что составлена, то в очень редких случаях она не содержит ошибок. Нужно заранее настраиваться на то, что в программе есть ошибки, и их нужно найти и исправить.

Можно выделить три типа ошибок:

— синтаксические ошибки, т. е. несоответствие конструкций в программе правилам бейсика. Эти ошибки самые простые, их легко исправить. При вводе оператора, содержащего синтаксическую ошибку, на экране, как правило, появляется указание на то место в операторе, где эта ошибка сделана, и на то, какого рода эта ошибка. То есть машина сама контролирует соответствие вводимой программы правилам языка. Однако в обнаружении и исправлении синтаксических ошибок нельзя полностью положиться на ЭВМ. Предположим, что при умножении переменной A на 2 вы забыли написать знак умножения (т. е. вместо A*2 в программе написано A2). Однако машина не воспринимает эту запись как ошибочную, считая, что вы используете переменную с именем A2 (что допускается правилами бейсика). Программа может быть даже выполнена, но она не даст правильного результата.

— ошибки в организации программы. Например, в программе имеется оператор

```
GO TO 500
```

а оператора с номером 500 в программе нет.

При выполнении такого оператора машина выдает соответствующее сообщение об ошибке.

— ошибки в алгоритме. Эти ошибки труднее всего найти. Они не могут быть обнаружены машиной, так как она не «знает» решаемой задачи. Это, например, так называемые *зацикливания*, когда программа не выходит из цикла и ничего не печатает. Иногда просто высвечиваются неправиль-

ные результаты или результаты вычисляются правильно, но не выводятся из-за отсутствия оператора PRINT. В этом случае нужно остановить выполнение программы и просмотреть программу внимательно.

Однако, как правило, сразу за дисплеем найти ошибки не удается. В этом случае нужно сесть за стол (выключить машину) и применить очень действенный метод проверки работы вашей программы, который называется: *выполнение программы вручную*. Для этого нужно представить себе, что вы — машина, и, начав с первого оператора программы, выполнять оператор за оператором, пока не будет обнаружена причина неправильной работы программы. Нужно выполнять эту работу методично и терпеливо, отключив свой интеллект, не делая правдоподобных догадок и не перескакивая через несколько операторов.

Конечно, лица, имеющие опыт составления программ и работы на ЭВМ, обычно выполняют эту работу до первого выхода на машину. Тем, кто только приступает к изучению программирования, требуется некоторое время, чтобы уяснить, как машина воспринимает и выполняет операторы программы. И в первых своих программах новички делают ошибки, которые они сами обнаружить не в состоянии. Они могут поверить в ошибки, только если машина не выполняет составленную ими программу или выполняет ее неправильно.

Чтобы имитировать работу ЭВМ, нужно хорошо понимать, как она работает. Представим себе, что мы остановили выполнение программы и заглянули внутрь ЭВМ. Что бы мы там «увидели»?

1. В памяти записана программа в той форме, в которой мы ее ввели.

2. В другом участке памяти располагаются переменные, которые были созданы к настоящему моменту. Каждая переменная занимает свое место в памяти (ячейку памяти) и имеет значение.

3. В памяти хранится еще одна специальная переменная, в которой находится номер выполняемой в данный момент строки (оператора). То есть машина «знает», в каком месте программы она находится. После выполнения очередного оператора значение этой специальной переменной, которая имеет название *указатель программы*, изменяется. Если выполняемым оператором был оператор PRINT, по которому что-то появилось на экране, или оператор присваивания, по которому в ячейку памяти помещено значение, то после его выполнения значение *указателя про-*

*граммы* заменяется на следующий (в порядке возрастания) номер в последовательности операторов и машина начнет выполнять оператор с этим следующим номером.

При выполнении оператора GO TO ничего не высвечивается и не изменяются значения переменных, просто значение указателя программы заменяется на тот номер, который указан в операторе GO TO.

Оператор IF выполняется так же, как GO TO, если *условие* выполняется (имеет значение *да*). Если *условие* не выполняется (имеет значение *нет*), то в указателе программы появляется номер следующего оператора в программе.

По оператору STOP прекращается выполнение программы.

Имитируя работу машины, мы должны иметь:

— программу;

— переменные с их текущими значениями;

— значение указателя программы (номер выполняемого оператора);

— информацию, высвечиваемую на дисплее при выполнении программы.

Для нас памятью (подобной памяти ЭВМ) будет лист бумаги, на котором в отведенных местах записана эта необходимая информация.

При выполнении программы вручную удобно пользоваться следующей таблицей. Назовем ее таблицей выполнения программы.

Таблица 2.3. **Таблица выполнения программы**

| Номер выполняемого оператора | |
|---|---|
| Переменные и их текущие значения | |
| Дисплей | Программа |
| | |

В верхней строке этой таблицы записывается текущее значение указателя программы, т. е. номер выполняемого оператора. При изменении этого значения старый номер зачеркиваем и рядом пишем новый. В следующей строке будем записывать имя той переменной, которая впервые встретилась при выполнении программы, и ее значение.

Если значение переменной изменяется, то старое значение зачеркнем и рядом напишем новое. В основной части таблицы справа располагаем выполняемую программу, а слева выписываем то, что в соответствии с операторами PRINT должно появляться на экране.

Выполняя программу вручную, нужно на время забыть «цель» программы и выполнять оператор с номером 20 только потому, что такое значение имеет указатель программы.

Как же найти ошибку в программе, выполняя ее вручную? Для этого нужно переключаться из состояния «робота», в котором вы находитесь, выполняя очередной оператор, в состояние человека, обладающего интеллектом. Сначала вы — «робот» и выполняете оператор точно так, как это делает машина, потом опять становитесь человеком и спрашиваете: «Тот ли получен результат, которого я ожидал?» Если — да, то продолжаете выполнение программы. Если — нет, нужно думать, почему программа работает неправильно.

Рассмотрим пример. Предположим, что мы хотим напечатать таблицу умножения для числа 12, т. е. получить на экране:

```
ТАБЛИЦА УМНОЖЕНИЯ НА 12
 1 * 12 = 12
 2 * 12 = 24
   . . . . .
12 * 12 = 144
```

Для этого мы написали программу

```
10 PRINT "ТАБЛИЦА УМНОЖЕНИЯ НА 12"
20 P=1
30 P=P+1
40 IF P<=12 GO TO 30
50 PRINT P;"*12=";P*12
60 STOP
```

Ввели эту программу и после ее выполнения получили на экране:

```
ТАБЛИЦА УМНОЖЕНИЯ НА 12
13 * 12 = 156
STOP AT LINE 60
```

То есть получен не тот результат, который мы ожидали.

Для проверки работы программы выполним ее вручную. Составим таблицу выполнения.

Таблица 2.4

| Номер выполняемого оператора | 10 | 20 | 30 | 40 | 30 |
|---|---|---|---|---|---|
| Переменные и их текущие значения P : | 1 | 2 | 3 | | |
| Дисплей | Программа | | | | |
| | 10 PRINT "ТАБЛИЦА УМНОЖЕНИЯ НА 12"<br>20 P=1<br>30 P=P+1<br>40 IF P<=12 GO TO 30<br>50 PRINT P;"*12=";P*12<br>60 STOP | | | | |

Выполняя программу, в некоторый момент осознаем, что P изменяется, а на экране ничего не появляется (действительно, оператор печати расположен после цикла!). Правильная программа выглядит следующим образом:

```
10 PRINT "ТАБЛИЦА УМНОЖЕНИЯ НА 12"
20 PRINT
30 P=1
40 PRINT P;"*12=";P*12
50 P=P+1
60 IF P<=12 GO TO 40
70 STOP
```

Однако описанный метод проверки работы программы не дает результата, если вы не знаете бейсика. Допустим, вы считаете, что * означает сложение. Тогда, например, если P=4, и вы выполняете вручную оператор T=P*P, то T будет иметь у вас значение 8, тогда как при выполнении этого оператора машиной T будет присвоено значение 16. Таким образом, если результат работы программы не совпадает с ожидаемым, а выполнение программы вручную дает ожидаемый результат, то вам нужно повторить бейсик.

### 3. Составление программ с использованием ввода данных

Программы, которые мы рассматривали до сих пор, не требуют никакой дополнительной информации для того, чтобы машина могла их выполнить. Каждая из них решает одну конкретную задачу, и при ее выполнении всегда получается один и тот же результат.

Обычно программа составляется в общем виде так, чтобы без каких-либо изменений она могла использоваться для решения различных задач одного класса (вспомним свойство алгоритма — массовость). Так как задачи одного

класса различаются исходными данными, а решаются с использованием одной и той же программы, то, следовательно, эти исходные данные в самой программе фигурировать не должны. Но при выполнении программа должна обрабатывать конкретный набор исходных данных.

Пусть, например, мы хотим составить программу для получения таблицы умножения для любого натурального числа. Вместо конкретного числа в программе используем переменную N. Тогда в предыдущей программе оператор 40 нужно заменить на

```
40 PRINT P; "*"; N; "="; P*N
```

и оператор 60 на

```
60 IF P<=N GO TO 40
```

(Оператор 10 также требует изменения, но на выполнение программы он не влияет.)

Перед началом вычислений в переменные (точнее, в ячейки памяти), предусмотренные в программе для исходных данных, должны быть введены значения. Для выполнения этой операции предназначен оператор ввода INPUT (см. п. 5.3 главы 3). Так, оператор

```
INPUT N
```

обеспечит при выполнении программы ввод значения, набираемого на клавиатуре, а переменную N. Этот оператор должен быть выполнен до первого использования переменной N в программе.

При выполнении оператора INPUT на экране появляется знак ?, означающий, что машина ждет ввода данных. Желательно, чтобы при этом на экране появлялось также точное указание о том, какие данные должны быть введены в ЭВМ.

Вывести сообщение на экран можно при помощи оператора PRINT. Например, при выполнении операторов

```
10 PRINT "ВВЕДИТЕ ЧИСЛО , ДЛЯ КОТОРОГО"
11 PRINT "ХОТИТЕ ПОЛУЧИТЬ ТАБЛИЦУ УМНОЖЕНИЯ"
20 INPUT N
```

на экране появится:

```
ВВЕДИТЕ ЧИСЛО, ДЛЯ КОТОРОГО
ХОТИТЕ ПОЛУЧИТЬ ТАБЛИЦУ УМНОЖЕНИЯ
?
```

Далее нужно набрать это число на клавиатуре и нажать ВК.

Программа 2.1 составлена для получения таблицы умножения для любого числа.

**Программа 2.1**

```
10 PRINT "ВВЕДИТЕ ЧИСЛО , ДЛЯ КОТОРОГО"
11 PRINT "ХОТИТЕ ПОЛУЧИТЬ ТАБЛИЦУ УМНОЖЕНИЯ"
20 INPUT N
30 P=1
40 PRINT P;"*";N;"=";P*N
50 P=P+1
60 IF P<=N GO TO 40
70 STOP
```

В бейсике, помимо числовых переменных, с которыми мы имели дело до сих пор, могут использоваться и символьные переменные, значениями которых являются последовательности символов (см. п. 6 главы 3). Для обозначения символьной переменной к ее имени приписывается знак $. Значения символьных переменных можно задавать так же, как значения числовых, в частности, использовать оператор ввода INPUT.

Возможность ввода данных (строк символов) в символьные переменные в бейсике делает возможным разработку программ, обладающих диалоговыми свойствами, так что работа ЭВМ по программе воспринимается как реакция или даже инициатива вполне сознательного партнера.

Составим, например, программу, которая спрашивает имя и затем здоровается с его обладателем. При выполнении программы на экране должен появиться вопрос: «Как Вас зовут?». Имя, набираемое далее на клавиатуре, машина должна запомнить в какой-либо символьной переменной (например, N $) и далее значение этой переменной выводить на экран со словами приветствия. Далее приводится программа 2.2 и результат ее выполнения.

**Программа 2.2**

```
10 PRINT "КАК ВАС ЗОВУТ ?"
20 INPUT N$
30 PRINT "ПРИВЕТ , "N$"!"
40 STOP

RUNNH

КАК ВАС ЗОВУТ ?
? ВАСЯ
ПРИВЕТ , ВАСЯ!

STOP AT LINE 40
```

## 4. Порядок решения задач с использованием ЭВМ

Решение задач с использованием ЭВМ включает следующие этапы: формулировка задачи на профессиональном языке, математическая постановка задачи, выбор метода решения, выбор структуры данных и разработка алгоритма, программирование, отладка и тестирование программы, счет по готовой программе и анализ результатов. Подробно эти этапы рассмотрены в п. 7. Здесь в упрощенном виде рассмотрены те этапы, выполнение которых необходимо при решении простых задач в начальный период изучения программирования.

1. Точная формулировка цели решения задачи, т. е. определение, что именно нужно получить в результате решения задачи и в каком виде желательно получить результат.

2. Выбор обозначений для переменных программы. Выделение исходных данных и результатов.

Обозначения переменных должны быть согласованы с возможностями обозначения переменных в алгоритмическом языке (в частности, в бейсике). Информацию об используемых переменных удобно разместить в таблице 2.5.

Таблица 2.5. **Характеристика переменных программы**

| Имя переменной в программе | Физический смысл переменной | Назначение переменной | Ограничение на исходные данные |
|---|---|---|---|
| | | | |

Эту таблицу обычно не удается полностью заполнить перед составлением программы, так как в процессе разработки программы может появиться необходимость в дополнительных переменных для хранения промежуточных результатов, поэтому таблицу приходится дополнять при составлении программы (алгоритма).

3. Разработка алгоритма решения задачи, т. е. расчленение исходной задачи на отдельные простые этапы.

На этом этапе нужно выделить циклы и продумать их организацию. При разработке алгоритма необходимо использовать только типовые структуры алгоритмов и их сочетания. Алгоритм можно фиксировать в виде схемы, причем нужно иметь в виду, что схему приходится обычно переделывать много раз, до получения ее окончательного варианта.

4. Написание по схеме программы на бейсике.

При составлении программы нужно следить за тем, чтобы переменные, являющиеся исходными данными, вводились оператором INPUT, а результаты выводились оператором PRINT. В программе нужно предусмотреть печать необходимых заголовков и пояснений к выводимым результатам, а также печать указаний для выполнения операторов ввода.

Работа по составлению программы на бейсике не должна вызывать трудностей. Если вы испытываете трудности, значит, плохо был выполнен этап разработки алгоритма.

5. Ввод программы в машину и проверка ее работы на различных исходных данных.

Используя эти правила, составим программу для решения следующей задачи.

З а д а ч а. Представим себе пирамиду из шаров, основание которой представляет собой квадрат со стороной, состоящей из N шаров. Следующий слой состоит из шаров, лежащих в углублениях нижнего слоя, т. е. представляет собой квадрат со стороной, состоящей из N—1 шаров и т. д. Верхний слой содержит один шар. Требуется определить, сколько шаров потребуется для строительства пирамиды из N слоев.

Р е ш е н и е. Легко заметить, что один слой содержит число шаров, равное квадрату номера слоя (верхний слой, содержащий один шар, считаем первым). Поэтому математически задача сводится к вычислению суммы квадратов натуральных чисел от 1 до N, т. е. вычислению $1^2+2^2+\ldots+N^2$.

В результате решения задачи должно быть получено одно число — суммарное количество шаров, необходимое для строительства пирамиды. Это число должно быть выведено на экран с поясняющим текстом.

Исходными данными для задачи является количество слоев пирамиды, которое обозначим через N. Результат решения задачи — суммарное число шаров обозначим через R. Понадобится еще вспомогательная переменная: текущий номер слоя — обозначим его через K. Заполним таблицу 2.6 для переменных программы.

Для решения задачи будем постепенно накапливать в переменной R сумму («текущее» общее количество) шаров, составляющих один слой, два слоя, три слоя и т. д. После окончания суммирования в R будет получена сумма шаров, составляющих все N слоев, т. е. результат решения задачи.

Таблица 2.6. **Характеристика переменных программы**

| Имя переменной | Физический смысл переменной | Назначение переменной | Ограничения на исходные данные |
|---|---|---|---|
| R | «Текущее» общее число шаров | Результат | |
| N | Число слоев пирамиды | Исх. дан. | Любое натуральное |
| K | Число слоев, обработанных к данному моменту | Вспомог. перемен. | |

Таким образом, программа должна содержать цикл по номеру слоя. Данные, необходимые для организации цикла, внесем в таблицу 2.7.

Таблица 2.7

| Управляющая переменная цикла | Начальное значение | Конечное значение | Шаг изменения | Количество повторений цикла |
|---|---|---|---|---|
| K | 1 | N | 1 | N |

Составим промежуточный вариант схемы (рис. 2.1*а*), в котором показана организация цикла, но некоторые блоки (обозначены пунктирной линией) требуют конкретизации. Уточним их содержание. Начнем с тела цикла. Тело цикла должно содержать действия, выполняемые для одного слоя. Это — вычисление числа шаров в K-м слое, равное K*K, и прибавление его к «текущему» общему R. Это действие записывается как

$$R = R + K * K$$

Начальными операторами должны быть заданы значения тех переменных, которые используются в цикле, с тем чтобы при первом прохождении цикла переменные в правых частях операторов присваивания и в операциях сравнения имели значения. Для нашей задачи переменной R должно быть присвоено значение 0 и введено значение N.

Введем эти уточнения и составим схему в окончательном варианте (рис. 2.1*б*).

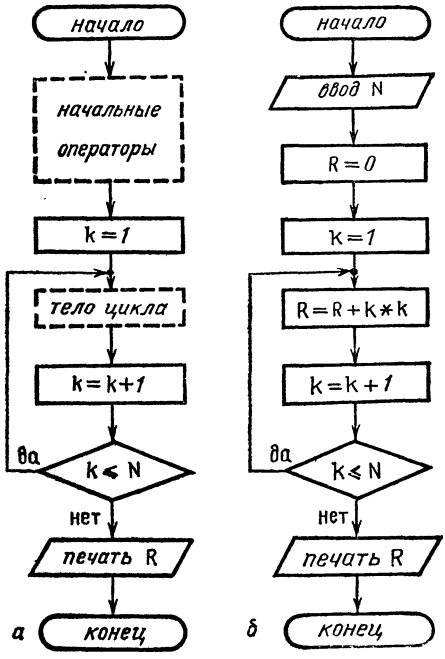



Рис. 2.1

Далее приводится программа 2.3, составленная в соответствии со схемой, и результат ее выполнения при N=4.

**Программа 2.3**

```
10 PRINT "СКОЛЬКО СЛОЕВ"
20 INPUT N
30 R=0
40 K=1
50 R=R+K*K
60 K=K+1
70 IF K<=N GO TO 50
80 PRINT R;" ШАРОВ ПОНАДОБИТСЯ"
90 STOP

RUNNH

СКОЛЬКО СЛОЕВ
? 4
 30 ШАРОВ ПОНАДОБИТСЯ

STOP AT LINE 90
```

## 5. Составление программ разветвляющейся структуры. Контроль ввода данных

Изучение раздела рекомендуется после выполнения работ 1, 2 (глава 4).

При программировании разветвления на языке бейсик следует иметь в виду, что на схеме (рис. 1.5) отдельные ветви разветвления расположены параллельно, а в программе они должны следовать друг за другом. Причем, если одна ветвь выполнена, то другую выполнять не нужно. Этот обход второй ветви (действие 2) осуществляется оператором GO TO N, который обеспечивает переход к общей части программы после выполнения первой ветви (действие 1).

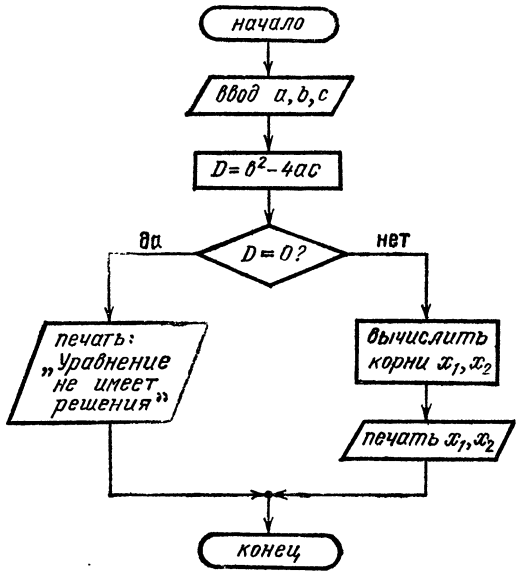



Рис. 2.2

Организация разветвления в программе на бейсике осуществляется оператором условного перехода IF. Сразу за оператором IF должны располагаться операторы действия 2. Эта часть программы начинает выполняться, если условие в операторе IF имеет значение *нет*, т. е. структура разветвление имеет следующий программный эквивалент:

1 IF *условие* GO TO 4
2 Операторы действия 2
3 GO TO 5

4 Операторы действия 1
5 Операторы общей части программы

В качестве примера составим программу для решения (нахождения вещественных корней при $a \neq 0$) квадратного уравнения

$$ax^2 + bx + c = 0.$$

Квадратное уравнение имеет решение, если дискриминант $D = b^2 - 4ac$ не является отрицательным, т. е. после ввода коэффициентов $a$, $b$, $c$ нужно вычислить $D$ и проверить условие $D < 0$. Если это условие выполняется (имеет значение *да*), то нужно вывести сообщение «Уравнение не имеет решения». Если условие $D < 0$ не выполняется, нужно напечатать корни, вычисляемые по формулам

$$x_1 = (-b + \sqrt{D})/2a, \quad x_2 = (-b - \sqrt{D})/2a.$$

Схема представлена на рис. 2.2.

Далее приводится программа 2.4, составленная в соответствии со схемой.

**Программа 2.4**

```
10 PRINT "ВВЕДИТЕ КОЭФФИЦИЕНТЫ УРАВНЕНИЯ"
20 INPUT A,B,C
30 D=B*B-4*A*C
40 IF D<0 GO TO 90
50 PRINT "КОРНИ УРАВНЕНИЯ"
60 X1=(-B+SQR(D))/(2*A) \ X2=(-B-SQR(D))/(2*A)
70 PRINT X1;X2
80 GO TO 100
90 PRINT "УРАВНЕНИЕ НЕ ИМЕЕТ РЕШЕНИЯ"
100 STOP
```

Используя составленную программу, решим следующую задачу. Определить, через сколько секунд упадет на землю камень, брошенный с высоты 2 м вверх с начальной скоростью 10 м/с.

Зависимость вертикальной координаты камня от времени определяется уравнением

$$h(t) = 2 + 10t - gt^2/2,$$

где $g = 9{,}8$ м/с$^2$ — ускорение свободного падения.

Для решения поставленной задачи нужно найти значение $t$, при котором $h(t) = 0$, т. е. необходимо решить квадратное уравнение

$$-gt^2/2 + 10t + 2 = 0.$$

Это можно сделать, выполнив составленную программу при A=—4.9, B=10, C=2.

Результат выполнения программы

```
ВВЕДИТЕ КОЭФФИЦИЕНТЫ УРАВНЕНИЯ
? -4.9,10,2
КОРНИ УРАВНЕНИЯ
-.1835  2.22432

STOP AT LINE 100
```

Физический смысл имеет только второй (положительный) корень — искомое время в секундах.

Частный случай разветвления — о б х о д (см. рис. 1.6) имеет следующий программный эквивалент:

1 IF *условие* GO TO 3
2 Операторы действия
3 Операторы общей части программы

При программировании обхода нужно следить за тем, чтобы обход осуществлялся по стрелке *да* во избежание лишних операторов GO TO.

В качестве примера составим программу для нахождения максимального из двух чисел $x$ и $y$, используя структуру о б х о д. По сравнению со схемой (см. рис. 1.10) программа 2.5 дополнена операторами ввода данных и вывода результата.

**Программа 2.5**

```
10 PRINT "ВВЕДИТЕ ДВА ЧИСЛА"
20 INPUT X,Y
30 U=X
40 IF U>=Y GO TO 60
50 U=Y
60 PRINT "БОЛЬШЕЕ ЧИСЛО";U
70 STOP
```

(>= употребляется вместо $\geqslant$).

При вводе данные, как правило, подвергаются различным видам контроля. В частности, если исходные данные не принадлежат диапазону, для которого программа обеспечивает правильную обработку, программа не должна выполняться. Организация контроля вводимых данных для этого случая демонстрируется в следующем примере.

П р и м е р. Программа — льстец.

На экране должен появляться вопрос: «Кто ты: мальчик или девочка? Введи M или D», и в зависимости от ответа на экране должен появляться текст «Мне нравятся девочки» (если было введено D) или «Мне нравятся мальчики» (если было введено M).

Исходные данные (M или D) будем вводить в символьную переменную, например S□. Текст, подлежащий выводу (в зависимости от ответа), присвоим другой символьной

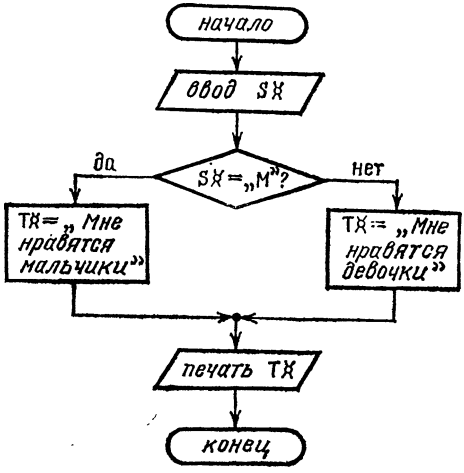


Рис. 2.3

переменной, например T□, и затем эту переменную выведем на печать (см. программу 2.6).

Схема алгоритма для этого примера приведена на рис. 2.3.

**Программа 2.6**

```
10 PRINT "КТО ТЫ: МАЛЬЧИК ИЛИ ДЕВОЧКА ? ";
20 PRINT "ВВЕДИ M ИЛИ D"
30 INPUT S$
40 IF S$="M" GO TO 70
50 T$="МНЕ НРАВЯТСЯ ДЕВОЧКИ"
60 GO TO 80
70 T$="МНЕ НРАВЯТСЯ МАЛЬЧИКИ"
80 PRINT T$
90 STOP
```

Недостатком этой программы является то, что она не защищена от неправильного ввода. Действительно, если вводится любой другой символ (или символы), кроме M, на экране появится текст «Мне нравятся девочки». Например, недостаточно внимательный мальчик может ввести что-нибудь вроде: «Я — мальчик», и ответ машины будет неверным. Для защиты программы от неправильного ввода необходимо, если условие S□="M" не выполняется, проверять также усдовие S□="D", и если оно также не выпол-

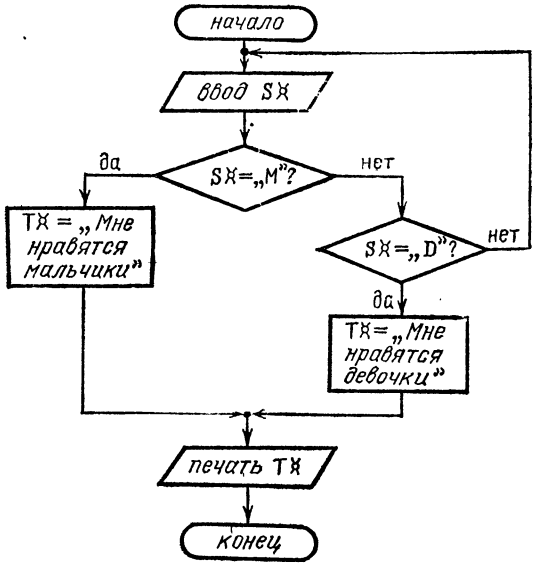



Рис. 2.4

няется, то нужно вернуться к оператору ввода. Этот прием иллюстрируется следующей схемой (рис. 2.4) и программой 2.7.

**Программа 2.7**

```
10 PRINT "КТО ТЫ: МАЛЬЧИК ИЛИ ДЕВОЧКА ? "
20 PRINT "ВВЕДИ M ИЛИ D"
30 INPUT S$
40 IF S$="M" GO TO 90
50 IF S$="D" GO TO 70
60 GO TO 20
70 T$="МНЕ НРАВЯТСЯ ДЕВОЧКИ"
80 GO TO 100
90 T$="МНЕ НРАВЯТСЯ МАЛЬЧИКИ"
100 PRINT T$
110 STOP
```

З а м е ч а н и е. В приведенной программе (а также на схеме) допущено отступление от структурного подхода. Строгое соблюдение правил, допускающих использование только типовых структур алгоритма, потребовало бы введения вспомогательной переменной и привело бы к необоснованному усложнению программы и потере ее наглядности.

## 6. Составление программ для обработки потока данных

Часто требуется обработать одинаковым образом данные, поступающие в ЭВМ последовательно друг за другом. Например, найти средний рост учеников класса, вводя по очереди рост каждого ученика, или найти суммарную стоимость всех товаров, купленных одним покупателем в универсальном магазине, и т. п. Рассмотрим два способа организации таких программ:

— число данных известно до начала работы программы. Тогда в начале программы количество данных вводится в переменную, которая используется далее везде, где требуется общее число данных (например, для проверки условия окончания цикла);

— число данных произвольно и заранее неизвестно. В этом случае для окончания ввода используется специальное значение того же типа, что и вводимые данные (признак конца данных), которое заведомо не может встретиться в потоке данных. Например, если все вводимые числа положительные, то в качестве признака конца может использоваться любое отрицательное число. Следующие два примера иллюстрируют рассмотренные способы.

Пример. В классе N учеников. Определить средний рост учеников класса, вводя в ЭВМ по очереди рост каждого ученика.

Решение. Рост учеников будем суммировать в переменной S, а затем разделим эту сумму на общее число учеников N.

Далее приводится программа 2.8 для решения этой задачи.

**Программа 2.8**

```
10 PRINT "СКОЛЬКО УЧЕНИКОВ В КЛАССЕ ?"
20 INPUT N
30 S=0
40 FOR I=1 TO N
50 PRINT "ВВЕДИТЕ РОСТ СЛЕДУЮЩЕГО УЧЕНИКА"
60 INPUT P
70 S=S+P
80 NEXT I
90 IF N=0 GO TO 130
100 S=S/N
110 PRINT "СРЕДНИЙ РОСТ УЧЕНИКОВ = ";S
120 STOP
130 PRINT "В КЛАССЕ НЕТ УЧЕНИКОВ"
140 STOP
```

*Список используемых переменных* *). Исходные данные: N — число учеников, P — рост одного ученика.

Результат: S — средний рост учеников класса.

Вспомогательные переменные: I — счетчик числа учеников (управляющая переменная цикла).

Чтобы получить результат работы этой программы, нужно ввести все N данных.

Пример. Определить средний рост учеников класса. Для окончания ввода ввести 0.

Далее приводится программа 2.9 для решения этой задачи.

**Программа 2.9**

```
10 S=0
20 I=0
30 PRINT "ВВЕДИТЕ РОСТ СЛЕДУЮЩЕГО УЧЕНИКА"
40 PRINT "ДЛЯ ОКОНЧАНИЯ ВВОДА ИСПОЛЬЗУЙТЕ НОЛЬ"
50 INPUT P
60 IF P=0 GO TO 100
70 S=S+P
80 I=I+1
90 GO TO 30
100 IF I=0 GO TO 140
110 S=S/I
120 PRINT "СРЕДНИЙ РОСТ УЧЕНИКОВ = ";S
130 STOP
140 PRINT "В КЛАССЕ НЕТ УЧЕНИКОВ"
150 STOP
```

*Список используемых переменных.* Исходные данные: P — рост одного ученика.

Результат: S — средний рост учеников.

Вспомогательные переменные: I — число учеников.

Если общее число данных произвольно (второй способ), но при каждом прохождении цикла вводится и обрабатывается не одно (как в последнем примере), а два или более данных, то для удобства работы с программой одно данное целесообразно вводить отдельно от других и использовать его для организации цикла. Этот прием иллюстрируется в следующем примере.

Пример. Определить средний рост и вес учеников класса.

Решение. Рост и вес каждого ученика будем вводить различными операторами INPUT. Для окончания ввода в переменную для роста введем 0. Далее приводится программа 2.10, в которой реализован описанный прием для решения поставленной задачи.

*) Используется далее вместо таблицы переменных.

*Список используемых переменных.* Исходные данные: P, V — рост и вес одного ученика.

Результат: S, W — средний рост и вес учеников класса.

Вспомогательные переменные: I — число учеников.

**Программа 2.10**

```
10 S=0 \ W=0
20 I=0
30 PRINT "ВВЕДИТЕ РОСТ УЧЕНИКА"
40 PRINT "ДЛЯ ОКОНЧАНИЯ ВВОДА ВВЕДИТЕ НОЛЬ"
50 INPUT P
60 IF P=0 GO TO 130
70 S=S+P
80 PRINT "ВВЕДИТЕ ВЕС УЧЕНИКА"
90 INPUT V
100 W=W+V
110 I=I+1
120 GO TO 30
130 IF I=0 GO TO 170
140 S=S/I \ W=W/I
150 PRINT "СРЕДНИЙ РОСТ=";S,"СРЕДНИЙ ВЕС=";W
160 STOP
170 PRINT "УЧЕНИКОВ НЕТ"
180 STOP
```

З а м е ч а н и я. 1. Возможно написание нескольких операторов в одной строке (см. п. 1 главы 3). В этом случае операторы отделяются друг от друга символом \ (строки 10 и 140).

2. Если все данные, которые необходимо обрабатывать при каждом прохождении цикла, вводить одним оператором INPUT, то для окончания ввода нужно вводить такое же количество специальных значений, как и число данных. Например, если для ввода переменных P и V в приведенной программе использовать в цикле один оператор INPUT P,V, то при вводе P=0 нужно было бы вводить какое-либо фиктивное значение и для V, хотя для окончания цикла используется далее только значение P.

## 7. Этапы решения задачи на ЭВМ. Некоторые практические рекомендации по разработке и отладке программ

Изучение раздела рекомендуется после выполнения работ 1—3 (глава 4).

Рассмотрим теперь подробнее все этапы решения задач на ЭВМ.

Следует отметить, что отдельные этапы взаимосвязаны: последующие этапы зависят от реализации предшествующих, а после выполнения очередного этапа может потребоваться возврат к предыдущим этапам и поиск их новых решений.

Первым этапом постановки и решения задачи на ЭВМ являются четкая формулировка задачи (обычно на профессиональном языке), выделение исходных данных для ее решения и точные указания относительно того, какие результаты и в каком виде должны быть получены.

Второй этап — формальная (математическая) постановка задачи, т. е. представление ее в виде уравнений, соотношений, ограничений и т. п. При этом некоторые задачи, решаемые в настоящее время на ЭВМ, либо не допускают, либо не требуют математической постановки (например, задачи обработки текстов).

Третий этап — выбор метода решения. Выбор метода определяется решаемой задачей, а также возможностями ЭВМ (ее быстродействием, объемом памяти, точностью представления чисел, наличием разработанных ранее готовых программ (см. п. 9 главы 3) и т. п.). Выполнение этого этапа требует некоторого кругозора как в области программирования, так и в области используемых методов.

Четвертый этап — разработка алгоритма на основе выбранного метода. При выборе алгоритма желательно рассмотреть и проанализировать несколько вариантов, прежде чем сделать окончательный выбор. Следует обратить внимание на тесную взаимосвязь третьего и четвертого этапов, так как алгоритм в большой степени определяется выбранным методом, хотя один и тот же метод в свою очередь может быть реализован при помощи различных алгоритмов.

При разработке алгоритма решения сложной задачи следует использовать метод пошаговой детализации (см. п. 3 главы 1), следя за тем, чтобы на каждом шаге структура алгоритма оставалась простой и ясной. Следует максимально использовать существующие типовые или разработанные ранее алгоритмы для отдельных фрагментов (блоков) алгоритма.

Пятый этап — выбор структуры данных. От выбора способа предоставления данных зависит и алгоритм их обработки. Поэтому четвертый и пятый этапы взаимосвязаны. Нужно выбирать структуру данных, наиболее естественную для решаемой задачи, использовать массивы для

представления данных, когда это наиболее очевидный способ их организации.

Шестой этап — собственно программирование, т. е. запись разработанного алгоритма на языке программирования. Вопросы выбора подходящего языка здесь не рассматриваются.

Если разработка алгоритма выполнена хорошо, то программирование принципиальных трудностей не вызывает.

Однако можно высказать некоторые рекомендации по составлению программы, которые облегчат ее отладку и дальнейшее использование.

1. Программа должна быть универсальной, т. е. не зависящей от конкретного набора данных. Если данные представлены в виде массивов, то в качестве исходных данных следует рассматривать не только сами данные, но и их количество, для того чтобы одна и та же программа могла быть использована для обработки массивов различных размеров. Подробнее эти вопросы рассмотрены во введении к работе 8 (глава 4).

Универсальная программа должна обрабатывать вырожденные случаи (например, число элементов вектора равно 0 или 1) и печатать сообщение об ошибке, если размер массива превысил допустимое значение (использованное в описании массива).

2. Вместо констант лучше использовать переменные. Если в программе используются константы, то при их изменении нужно изменять в исходной программе каждый оператор, содержащий прежнюю константу. Эта процедура отнимает много времени и часто вызывает ошибки.

В программе следует предусмотреть контроль вводимых данных (в частности, программа не должна выполняться, если данные выходят за пределы допустимого диапазона; см., например, программу 2.7).

3. Некоторые простые приемы позволяют повысить эффективность программы (т. е. уменьшить количество выполняемых операций и время работы программы). К таким приемам относятся:

— использование операции умножения вместо возведения в степень для низких степеней, например X*X*X вместо X ^ 3, выполняется быстрее;

— арифметическое выражение, которое несколько раз вычисляется в программе с одними и теми же данными, лучше вычислить один раз и присвоить его значение переменной, которую и использовать везде вместо арифметического выражения;

— при организации циклов в качестве границ индексов использовать переменные, а не выражения, которые вычислялись бы при каждом прохождении цикла;

— особое внимание обратить на организацию циклов, убрав из них все повторяющиеся с одинаковыми данными вычисления и выполняя их до входа в цикл.

4. Программа должна содержать комментарии, позволяющие легко проследить за логической взаимосвязью и функциями отдельных ее частей.

При написании программы следует заботиться о ее структуре так, чтобы программа была удобочитаемой. В частности, в программе должны быть хорошо видны циклы. Для этого операторы FOR и NEXT нужно размещать в различных строках, не содержащих других операторов (см., например, программу 2.8). Если цикл организуется без использования операторов FOR и NEXT, то в отдельных строках, не содержащих других операторов, должны быть размещены операторы, организующие цикл (задающие начальное значение управляющей переменной цикла, осуществляющие проверку условия выхода из цикла и т. п.). В отдельных случаях операторы FOR и NEXT могут быть размещены в одной строке. Тогда оператор FOR должен быть первым в строке, оператор NEXT — последним (см., например, программу 3.6).

Седьмой этап — тестирование и отладка программы — это проверка правильности работы программы и исправление обнаруженных ошибок. (Эти вопросы частично затрагивались в п. 2 главы 3.)

Для выполнения тестирования необходимо подготовить тесты. *Тест* — это специально подобранные исходные данные в совокупности с теми результатами, которые должна выдавать программа при обработке этих данных. Разработка тестов — трудоемкая работа, часто требующая выполнения ручных просчетов. При составлении тестов нужно стремиться обеспечить проверку всех ветвей программы.

Далее даются некоторые рекомендации по тестированию и отладке.

1. Проверьте работу программы (или отдельных ее частей) вручную до выхода на машину (см. п. 2 главы 2).

2. Проводите тестирование и отладку отдельно для логически самостоятельных частей программы.

3. Используйте отладочную печать в наиболее ответственных местах программы. Операторы печати для отладки располагайте в отдельных строках так, чтобы можно было легко убрать их из программы после окончания отладки.

4. При отладке программы в диалоговом режиме (например, программы на бейсике) предусмотрите в программе несколько операторов STOP (см. п. 4.6 главы 3).

5. При тестировании, если это возможно, используйте меньшие объемы данных, чем те, на которые рассчитана программа.

Восьмой этап — счет по готовой программе и анализ результатов. Этот этап является итогом выполнения всех предыдущих этапов и служит подтверждением (или опровержением) их правомерности. После этого этапа, возможно, потребуется пересмотр самого подхода к решению задачи и возврат к первому этапу для повторного выполнения всех этапов с учетом приобретенного опыта. Более подробно затронутые здесь вопросы обсуждаются в книгах [2, 7].

# Глава 3. ОПИСАНИЕ ЯЗЫКА БЕЙСИК

## 1. Общая характеристика алгоритмического языка бейсик

Алгоритмический язык бейсик используется преимущественно в режиме диалога человека и ЭВМ. Этот язык ориентирован на решение различных задач вычислительного и невычислительного характера с небольшим объемом исходной информации. Название языка BASIC возникло от сокращения английских слов *Beginner's All-purpose Symbolic Instruction Code* (многоцелевой язык символических инструкций для начинающих).

Следует отметить, что стандарта на язык бейсик не существует и различные его модификации могут существенно отличаться друг от друга. В настоящей книге дается описание языка бейсик применительно к ЭВМ «Электроника-60», СМ-4, ДВК-2.

В языке бейсик существуют как средства для описания действий алгоритма, которые используются при составлении программ,— операторы бейсика, так и средства, которые служат для общения с ЭВМ. Последние имеют форму приказов для немедленного выполнения и называются командами.

Основным режимом в бейсике является программный режим, когда заранее составленная программа полностью вводится в ЭВМ и затем выполняется. Другим, вспомогательным режимом является так называемый режим непосредственного исполнения, когда операторы бейсика, как команды, выполняются сразу после ввода в ЭВМ (см. п. 12.5).

Программа на бейсике состоит из пронумерованных строк. В одной строке может содержаться один или несколько операторов, разделенных символом \ (косая черта слева направо). Обычно строки нумеруются, начиная с 10, с шагом 10. Вводиться строки могут в любом порядке, а выполняются в порядке возрастания их номеров. Номера строк используются в операторах передачи управления.

При этом оператор, которому передается управление, должен быть первым в строке.

Пример программы на бейсике.

**Программа 3.1**

```
10 S=0
20 I=1
30 S=S+I
40 I=I+1
50 IF I<=10 THEN 30
60 PRINT S

RUNNH

 55
```

Приведенная программа вычисляет сумму первых 10 натуральных чисел. Смысл каждого оператора и порядок их выполнения в программе очевидны.

RUNNH — это команда (вводится без номера). Она является приказом на выполнение программы. После набора RUNNH на клавиатуре нужно нажать клавишу ВК (возврат каретки), и программа начнет выполняться. В результате выполнения программы будет вычислено значение S и оператором PRINT выведено на экран.

## 2. Символы языка бейсик

В бейсике используются следующие символы:

1) 26 заглавных латинских букв от A до Z;

2) 10 цифр от 0 до 9;

3) знаки: . (точка), ; (точка с запятой), , (запятая), "(кавычки), ' (апостроф);

4) знаки арифметических операций: + (плюс), — (минус), * (знак умножения), / (знак деления), ^ (знак возведения в степень);

5) ( ) круглые скобки;

6) ⎵ пробел;

7) знаки операций отношения $<$, $>$, $=$ (сочетание $<=$ используется вместо $\leqslant$, $>=$ вместо $\geqslant$, $<>$ вместо $\neq$);

8) знаки: ⃞ (знак денежной единицы), & (коммерческое «и» — амперсанд), @ (коммерческое «эт»), \ (косая черта слева направо), % (процент), # (номер), ?, !;

9) буквы русского алфавита от А до Я.

Используемые в бейсике служебные слова и сокращения и их перевод на русский язык приводятся в приложении.

## 3. Простейшие конструкции языка

**3.1. Числа.** Запись чисел на языке бейсик близка к естественной. Числа, не имеющие дробной части, записываются привычным образом как последовательность цифр со знаком + или — (знак + можно опустить), например, +10, 12, —136. В числах, имеющих дробную часть, для отделения целой части от дробной используется точка вместо запятой, например, 1.2, —0.6, или —.6 (0 целых можно опустить). Такая форма записи чисел называется основной.

Допускается также запись чисел в форме с порядком. Например, число 0,00012, или эквивалентное ему $1{,}2 \cdot 10^{-4}$, на языке бейсик может быть записано как 1.2E—4, где E—4 используется вместо $10^{-4}$. Число 100000, или эквивалентное ему $10^5$, может быть записано как 1E5. Буква E и следующее за ней целое число называются порядком. Целое число в записи порядка может содержать не более двух цифр. Порядку обязательно должно предшествовать число, записанное в основной форме. Сравните: $10^5$, но 1E5.

Диапазон чисел, с которыми можно оперировать в бейсике, составляет от $10^{-38}$ до $10^{38}$, количество значащих цифр числа не должно превышать шести, лишние цифры будут отброшены. Так, например, если вводится число 61232736, то оно будет представлено в ЭВМ только первыми шестью цифрами, т. е. как, например, $6.12327 \cdot 10^7$. При вводе числа 1000001 последняя 1 будет отброшена и число будет храниться в памяти как число $10^6$. При вводе числа 0.00001648 оно будет храниться как $1.648 \cdot 10^{-5}$, при вводе числа 0.0000001648974 оно будет храниться как число $1.64897 \cdot 10^{-6}$ и т. д.

Под число, записанное в одной из приведенных выше форм (т. е. в основной форме или в форме с порядком), выделяется ячейка памяти длиной в 4 байта, и число хранится в ней в так называемой форме с плавающей запятой. Это — вещественные числа.

В языке бейсик существует еще одна форма записи чисел, когда к числу приписывается знак %. Такая форма записи используется только для целых чисел (чисел, не имеющих дробной части). Например, 1%,— 34% и т. п. Целые числа могут принимать значения в диапазоне от —32768 до 32767. Под целое число, записанное в такой форме, выделяется ячейка памяти длиной 2 байта, и числа в ней представляются в так называемой форме с фиксированной запятой. Подробнее о формах представления чисел в памяти ЭВМ см., например, [8]. Целые числа в бейсике

используются в основном из соображений экономии памяти.

**3.2. Переменные.** Для обозначения вещественных переменных, т. е. переменных, значениями которых являются вещественные числа, в бейсике используются имена, состоящие либо из одной буквы, либо из буквы и цифры. Например, A, A2, C. В качестве букв используются прописные буквы латинского алфавита. Для целочисленных переменных, значениями которых являются целые числа, к имени добавляется знак %. Например, A%, A2%, C%.

Целочисленные и вещественные переменные должны принимать значения в пределах ограничений, указанных в 3.1. Выход за пределы этих диапазонов вызывает сообщение об ошибке.

**3.3. Стандартные функции.** При работе на ЭВМ имеется возможность использовать уже готовые (стандартные) программы, которые хранятся в памяти ЭВМ, для вычисления часто употребляемых функций. В таблице 3.1 приводятся стандартные функции, которые можно употреблять в бейсике. Аргумент стандартной функции заключается в круглые скобки. В качестве аргумента можно употреблять любое арифметическое выражение (см. п. 3.4).

Таблица 3.1. **Стандартные функции**

| Запись на бейсике | Математическое определение | Запись на бейсике | Математическое определение |
|---|---|---|---|
| SIN (X) | $\sin x$ | EXP (X) | $e^x$ |
| COS (X) | $\cos x$ | ABS (X) | $\lvert x \rvert$ |
| ATN (X) | $\operatorname{arctg} x$ | SGN (X) | $\begin{cases} 1, & x>0 \\ 0, & x=0 \\ -1, & x<0 \end{cases}$ |
| LOG (X) | $\ln x$ | | |
| LOG10 (X) | $\lg x$ | | |
| INT (X) | $[x]$ — наибольшее целое, не превосходящее $x$ | SQR (X) | $\sqrt{x}$ |
| | | RND (X) | Датчик случайных чисел |
| | | PI | число $\pi$ |

Замечание. Аргумент функции RND можно опустить. Аргумент тригонометрической функции задается в радианах. Для перевода значения, заданного в градусах, в радианы можно использовать формулу

*знач. в рад.=знач. в град*·$\pi$/180.

Для арктангенса значение угла находится в интервале ($-\pi$ /2; $\pi$/2).

Для получения других обратных тригонометрических функций можно использовать формулы

$\arcsin(x) = \operatorname{arctg}(x/\sqrt{1-x^2})$,
$\arccos(x) = \operatorname{arctg}(\sqrt{1-x^2}/x)$,
$\operatorname{arcctg}(x) = \operatorname{arctg}(1/x)$.

**3.4. Арифметические выражения.** Арифметические выражения соответствуют общепринятым алгебраическим выражениям. В арифметическое выражение могут входить числа, переменные, функции (стандартные или определяемые в программе (см. п. 10), соединенные знаками арифметических операций. Число или переменная также является арифметическим выражением. Для обозначения арифметических операций используются знаки +, —, * (умножение), / (деление), ^ (возведение в степень).

Примеры арифметических выражений.

| Обычная запись | Запись на бейсике |
|---|---|
| 4 | 4 |
| $a$ | A |
| $a+4$ | A+4 |
| $2a$ | 2*A |
| $\frac{ab}{c}$ | A*B/C |

Если в арифметическом выражении имеется несколько различных арифметических операций, то порядок их выполнения задается правилами приоритета.

Правила приоритета арифметических операций в бейсике следующие:

1. ^ (возведение в степень).
2. *, / (умножение, деление).
3. +, — (сложение, вычитание).

В арифметическом выражении могут употребляться круглые скобки. Если имеются скобки, то операции в скобках выполняются в первую очередь.

Если в выражении несколько операций подряд имеют одинаковый приоритет, то они выполняются по порядку слева направо. Например,

X/Y*Z соответствует $(x/y)z$,
X^Y^Z соответствует $(x^y)^z$,
X/(Y*Z) соответствует $x/(yz)$.

Замечание. В арифметическом выражении могут употребляться величины разных типов — целого и вещественного. Выполнение арифметических операций над величинами одного типа дает результат того же типа, сочетание целой величины и вещественной дает вещественный результат.

## 4. Основные операторы бейсика

**4.1. Оператор присваивания.** Оператор присваивания служит для присваивания переменной значения арифметического выражения и имеет вид

LET *v*=*ар. выр.*

(LET можно опустить), где *v* — переменная, которой присваивается значение.

При выполнении оператора присваивания вычисляется выражение в правой части и присваивается переменной в левой части *v*.

Примеры.

```
L=8
P=P+1
```

При употреблении оператора присваивания необходимо следить за тем, чтобы к моменту его выполнения переменные, входящие в выражение правой части, были определены (имели числовые значения).

**4.2. Оператор безусловного перехода.** Общий вид оператора

GO TO N

где N — номер строки.

Этот оператор передает управление первому оператору в строке с номером N.

Оператор GO TO используется, в частности, при организации разветвлений для того, чтобы обойти операторы второй ветви, если выполнена первая.

**4.3. Условные операторы.** Условные операторы служат для изменения порядка выполнения операторов в зависимости от какого-либо условия. Условные операторы могут использоваться для организации циклов и разветвлений.

Общий вид условных операторов

IF *условие* *) $\left\{\begin{matrix}\text{THEN}\\ \text{GO TO}\end{matrix}\right\}$ N

IF *условие* THEN *оператор*

где *условие* имеет вид

*(ар. выр.)*$_1$ θ *(ар. выр.)*$_2$

(θ — одна из операций отношения <, <= (≤) , > , >= (≥) , = , <> (≠));

N — номер строки;

*оператор* — любой оператор бейсика, в том числе это может быть другой условный оператор.

Действие условного оператора заключается в следующем: если условие удовлетворяется, то в первом случае осуществляется переход к строке с номером N, во втором случае выполняется оператор, следующий за THEN.

Если условие не удовлетворяется, то в первом случае осуществляется переход к оператору, следующему за условным. Во втором случае осуществляется переход к первому оператору следующей строки, т. е. все операторы в строке, следующие за условным оператором, при этом игнорируются.

Пример. Составить программу табулирования функции $y=x \ln x$ на отрезке [2; 3] с шагом 0, 1 (см. программу 3.2).

**Программа 3.2**

```
10 X=2
20 Y=X*LOG(X) \ PRINT X,Y
30 X=X+.1
40 IF X<=3 GO TO 20

RUNNH

 2            1.38629
 2.1          1.55807
 2.2          1.73461
 2.3          1.91569
 2.4          2.10112
 2.5          2.29073
 2.6          2.48433
 2.7          2.68178
 2.8          2.88293
 2.9          3.08766
 3            3.29583
```

*) В условии могут сравниваться также и символьные величины (см. п. 6).

В приведенной программе условный оператор используется для организации цикла. Если условие $X \leqslant 3$ выполняется, то осуществляется возврат в начало цикла к строке 20 для вычисления и печати значения функции в очередной точке.

Пример. Составить программу для вычисления площади круга или площади квадрата в зависимости от введенного признака (см. программу 3.3).

**Программа 3.3**

```
10 INPUT X,M
20 IF M<0 THEN S=X*X \ PRINT "ПЛОЩАДЬ КВАДРАТА";S \ GO TO 40
30 S=PI*X*X \ PRINT "ПЛОЩАДЬ КРУГА";S
40 STOP
```

Приведенная программа в зависимости от введенного значения M вычисляет площадь квадрата со стороной X или площадь круга с радиусом X. Если $M<0$, то выполняются оператор в строке 20, следующий за THEN, и далее следующие операторы в этой строке. Если $M \geqslant 0$, то оператор, следующий за THEN, не выполняется, а осуществляется переход к строке с номером 30.

**4.4. Операторы цикла.** Цикл является типичной структурой алгоритмов, реализуемых на ЭВМ. Для организации циклов в алгоритмических языках предусмотрены специальные операторы. В бейсике это пара операторов FOR — NEXT. Общий вид операторов

FOR $v=A_1$ TO $A_2$ STEP $A_3$
NEXT $v$

где $v$ — любая неиндексированная *) переменная — управляющая переменная цикла; $A_1$, $A_2$, $A_3$ — начальное и конечное значения и шаг изменения управляющей переменной цикла — любые арифметические выражения. Если $A_3=1$, то конструкцию STEP $A_3$ можно опустить. Операторы, расположенные между операторами FOR и NEXT, образуют тело цикла и выполняются многократно.

Выполнение цикла, образованного операторами FOR — NEXT, заключается в следующем: переменной $v$ присваивается начальное значение $A_1$, и она сравнивается с конечным значением $A_2$. Если при положительном шаге $A_3$ удовлетворяется условие $A_1 \leqslant A_2$ или при отрицательном шаге

*) Неиндексированная, или простая, переменная — это переменная, имеющая одно значение, в отличие от массивов, элементы которых называются индексированными переменными (см. п. 7).

$A_3$ удовлетворяется условие $A_1 \geqslant A_2$, то выполняются операторы, расположенные между операторами FOR и NEXT, и по оператору NEXT осуществляется возврат к началу цикла. Значение $v$ изменяется на шаг $A_3$, т. е. $v=v+A_3$, и снова проверяется условие. Если условие удовлетворяется, то тело цикла выполняется повторно. В противном случае происходит выход из цикла и переход к оператору, следующему за NEXT. Так как первая проверка условия выхода из цикла осуществляется до первого выполнения тела цикла, то возможна ситуация, когда тело цикла не будет выполнено ни разу.

Так, программа

```
10 FOR J=4 TO 20 STEP 2
20 PRINT J,J*J
30 NEXT J
40 STOP
```

эквивалентна следующей:

```
10 J=4
20 IF J>20 GO TO 60
30 PRINT J,J*J
40 J=J+2
50 GO TO 20
60 STOP
```

**4.5. Оператор-комментарий.** Для облегчения восприятия и большей наглядности программы в нее целесообразно включать комментарии, которые поясняют работу отдельных частей программы, характеризуют используемые переменные и т. д.

Для записи комментариев используется оператор REMARK (или сокращенно REM). В этом операторе за служебным словом REM могут быть записаны любые символы бейсика. Все, что записано за словом REM до конца строки, воспринимается как комментарий и при выполнении программы игнорируется. В качестве примера дополним комментариями приведенную выше программу 3.2.

**Программа 3.4**

```
10 REM ПРОГРАММА ТАБУЛИРОВАНИЯ ФУНКЦИИ Y=X*LOG(X)
20 X=2
30 REM НАЧАЛО ЦИКЛА ПО X
40 Y=X*LOG(X) \ PRINT X,Y
50 X=X+.1
60 IF X<=3 GO TO 40 \ REM КОНЕЦ ЦИКЛА
```

Включение операторов REM в программу никак не влияет на ее выполнение, так что программа 3.4 будет выполняться точно так же, как программа 3.2.

**4.6. Операторы STOP и END.** Оператор STOP прекращает выполнение программы. При этом на экране появляется сообщение

STOP AT LINE N

где N — номер строки, содержащей оператор STOP.

Останов программы позволяет распечатать значения переменных, изменить их значения (в режиме непосредственного исполнения; см. п. 12.5), т. е. оператор STOP является удобным средством для отладки программы.

Выполнение программы может быть продолжено командой GO TO N, которая вызывает продолжение работы программы со строки с номером N.

Допускается использование в программе нескольких операторов STOP. Использование оператора STOP не является обязательным в бейсике. Если оператор STOP отсутствует, то выполнение программы продолжается до ее естественного завершения, т. е. до выполнения операторов, расположенных в строке с наибольшим номером.

Оператор END указывает на физический конец программы. Если в программу включается оператор END, то он должен располагаться в строке с самым большим номером.

При выполнении оператора END закрываются все открытые файлы (см. п. 13.2) и останавливается выполнение программы. Никакого сообщения на экран при этом не выводится.

## 5. Операторы ввода/вывода

Операторы ввода служат для задания исходных данных при выполнении программы. Операторы вывода позволяют получить результаты выполнения программы.

Ввод в бейсике осуществляется одним из двух способов: использованием оператора INPUT или пары операторов READ, DATA.

**5.1. Операторы READ, DATA.** Эти операторы в программе всегда присутствуют одновременно.

Оператор READ имеет вид

READ $v_1$ [ , $v_2$, ...]

где $v_1$, $v_2$, ... — вводимые переменные (квадратные скобки обозначают необязательные параметры).

Оператор DATA имеет вид

DATA $a_1$ [ , $a_2$, ...]

где $a_1$, $a_2$, ... — константы.

Оператор DATA содержит данные, которые при выполнении операторов READ будут вводиться в переменные, перечисленные в списке ввода операторов READ. Данные, подлежащие вводу, могут располагаться в одном операторе DATA или в нескольких операторах DATA, но в том порядке, в котором эти данные должны использоваться оператором (или операторами) READ. Операторы DATA могут располагаться в любом месте программы. В строке, содержащей оператор DATA, не допускается использование других операторов.

Перед выполнением программы все данные, содержащиеся в операторах DATA, помещаются в специальную область памяти (область DATA). Порядок расположения данных в области DATA соответствует порядку следования различных операторов DATA в программе, а для каждого оператора DATA — порядку данных в этом операторе. Например, при наличии в одной программе следующих операторов:

```
10 DATA 10.6
40 DATA 0.1,15,1.2
120 DATA 5.1
```

данные в области DATA будут расположены в такой последовательности: 10.6, 0.1, 15, 1.2, 5.1.

При выполнении операторов READ данные из области DATA по очереди пересылаются в переменные, перечисленные в списке ввода операторов READ. Например, если в программе, содержащей приведенные выше операторы DATA, имеются также операторы

```
20 READ A,B,C
90 READ D
```

то после их выполнения переменные A, B, C, D будут иметь значения

A=10.6, B=0.1, C=15, D=1.2.

Если данных не хватает, то выводится сообщение об ошибке.

5.2. **Оператор RESTORE.** Данные из области DATA можно считывать повторно, используя опертор RESTORE.

Общий вид оператора

RESTORE [*ар. выр.*]

Значение целой части арифметического выражения определяет номер данного, с которого начнется повторное считывание. Если арифметическое выражение отсутствует, осуществляется возврат к началу списка. Например, после выполнения группы операторов

```
10 READ A,B,C,D
20 DATA 10.6,0.1,15,1.2
30 RESTORE     \ READ E,F
```

будет A=10.6, B=0.1, C=15, D=1.2, E=10.6, F=0.1.

5.3. **Оператор INPUT.** При использовании оператора INPUT данные вводятся с клавиатуры дисплея во время выполнения программы. Общий вид оператора

INPUT $v_1$ [ , $v_2$, ...]

где $v_1$, $v_2$, ... — переменные.

При выполнении этого оператора ЭВМ делает паузу, на экране высвечивается «?» и ЭВМ ждет ввода данных. На клавиатуре дисплея необходимо набрать числа, подлежащие вводу, в порядке, задаваемом списком ввода. Если чисел несколько, то они отделяются друг от друга запятыми. После этого нажатием клавиши ВК эти числа вводятся в ячейки памяти, выделенные для переменных, указанных в списке. Если данные не уместились в одной строке или их недостаточно, то на экране опять появляется «?» и система ждет ввода остальных чисел. После ввода значений всех переменных, перечисленных в списке, выполнение программы продолжается.

Пример.

```
10 INPUT A,B,C,D

RUNNH

? 10.6,0.1,15
? 1.2
```

Использование операторов READ, DATA целесообразно во время отладки программы, когда программа многократно выполняется с одними и теми же данными, вводимыми в этом случае один раз при вводе программы. Когда же отлаженная программа многократно выполняется с различными наборами данных, целесообразно использовать оператор INPUT. Использование операторов READ, DATA в

этом случае требовало бы перед каждым выполнением программы с новыми данными замены операторов DATA, т. е. изменения текста программы.

5.4. **Оператор PRINT.** Используется для вывода на экран дисплея (или на принтер (см. п. 13.1)) результатов вычислений. Общий вид оператора

PRINT *список*

Список вывода может содержать имена переменных, числа, арифметические выражения и тексты. Если список отсутствует, то при выполнении оператора PRINT просто осуществляется перевод строки. Тексты используются для печати всевозможных заголовков и пояснений. Текст заключается в кавычки или в апострофы и может содержать любые символы бейсика. Если в списке вывода содержатся арифметические выражения, то они сначала вычисляются, а затем выводятся вычисленные значения.

Для удобства восприятия выводимых результатов экран условно разделен на 5 зон, по 14 позиций каждая, и вывод каждого следующего данного осуществляется в следующую зону. Такая форма вывода обеспечивается использованием «,» (запятой) в качестве разделителя в списке вывода.

Например, при выполнении оператора

```
50 PRINT "АРГУМЕНТ",X,"ФУНКЦИЯ",X^2
```

при X=2 будет выведено

```
АРГУМЕНТ      2             ФУНКЦИЯ       4
```

Появление запятой всегда приводит к переходу к следующей зоне. Поэтому употребление двух запятых подряд или запятой в начале списка приводит к пропуску одной зоны. Например, при выполнении оператора

```
50 PRINT ,"АРГУМЕНТ",X
```

вывод начинается с 15-й позиции, т. е. при X=2 будет напечатано

```
              АРГУМЕНТ      2
```

Такой способ вывода с использованием в качестве разделителя запятой называется выводом в *зонном формате*.

После последнего элемента списка также может употребляться запятая. В этом случае при выполнении следующего оператора PRINT вывод элементов его списка будет

осуществляться в следующей зоне (без перевода строки). Например, при выполнении операторов

```
50 PRINT A,B,

90 PRINT P
```

при A=10.1, B=6.8, P=190 будет выведено в одну строку

```
 10.1          6.8          190
```

Отсутствие запятой после последнего элемента в списке приводит к переводу строки, так что следующий оператор PRINT начинает вывод с новой строки. Например, при выполнении операторов

```
50 PRINT A,B

90 PRINT P
```

при тех же значениях A, B, P будет выведено

```
 10.1          6.8
 190
```

Если в качестве разделителя используется «;», то вывод осуществляется в так называемом *плотном формате*, т. е. каждое следующее данное выводится после предыдущего через пробел. Например, оператор

```
50 PRINT A;B;P
```

приводит к следующей печати:

```
10.1 6.8 190
```

Такой способ вывода рекомендуется для более компактного вывода, когда выводится много данных, содержащих небольшое число значащих цифр.

Замечания. 1. Для удобства работы с программой перед каждым оператором INPUT рекомендуется располагать оператор PRINT, который выводил бы на экран текст, идентифицирующий выполняемый оператор.

Пример.

```
10 PRINT "ВВЕСТИ A,B,C"
20 INPUT A,B,C

RUNNH

ВВЕСТИ A,B,C
? 10.1,6.8,190
```

Если после текста в операторе PRINT расположить точку с запятой, то знак вопроса (?) при выполнении оператора INPUT будет выводиться в ту же строку, что и текст.

Пример.

```
10 PRINT "ВВЕСТИ A,B,C";
20 INPUT A,B,C

RUNNH

ВВЕСТИ A,B,C? 10.1,6.8,170
```

2. Если данных набрано больше, чем требуется, то выдается предупреждающее сообщение. Выполнение программы продолжается.

3. Результаты вычислений выводятся как десятичные числа (в основной форме), если они находятся в диапазоне от 0.01 до 999999. В противном случае числа выводятся в форме с порядком. При этом в стандартной форме вывода число перед порядком содержит обязательно один разряд (ненулевой) перед точкой и пять цифр в качестве дробной части. Под знак числа (кроме числа 0) предусматривается одна позиция, знак + не выводится. Например, при выполнении операторов

```
10 A=0.01       \ B=9.90000E-03 \ C=1.00000E+06
20 PRINT A,B,C
```

на экран будет выведено

.01 9.90000 E—03 1.00000 E 06

**5.5. Оператор PRINT USING.** Дает возможность задать точный образ вывода отдельных элементов списка.

Оператор PRINT USING имеет вид

PRINT USING *"формат", список*

где *формат* задает образ распечатки.

При выводе числовых значений предусматривается символ # для каждой выводимой цифры, а также знака числа. Если предполагается вывод с порядком, то для порядка предусматриваются символы ^^^^. Все остальные символы, включая пробелы, обозначают сами себя и выводятся без изменений. Для вывода числа, содержащего дробную часть, в формате задается столько символов #, сколько всего цифр требуется вывести, и точка для отделения цифр целой части от дробной. То есть использование формата

позволяет отсечь при печати ненужные цифры. Результат при этом округляется.

Примеры.

```
10 PRINT USING "#.## ##.#";5.629,-1.1

RUNNH

5.63 -1.1
10 PRINT USING "X=#.# F=####";.151,366.3

RUNNH

X=0.2 F= 366
```

Если в формате задан только один образ, то он может использоваться многократно для каждого элемента в списке вывода. При этом вывод каждого следующего элемента будет осуществляться в новую строку.

Пример.

```
10 PRINT USING "##.##";10.1,-4.828

RUNNH

10.10
-4.83
```

Использование оператора PRINT USING при выводе массивов (см. п. 7) позволяет выводить массивы в удобной для восприятия форме, что иллюстрирует программа 3.5.

**Программа 3.5**

```
10 DIM F2(4,4)
20 P=.5
30 FOR I=1 TO 4
40 FOR J=1 TO 4
50 P=P+1 \ F2(I,J)=P
60 PRINT USING '##.##  ';F2(I,J);
70 NEXT J
80 PRINT
90 NEXT I
100 STOP

RUNNH

 1.50   2.50   3.50   4.50
 5.50   6.50   7.50   8.50
 9.50  10.50  11.50  12.50
13.50  14.50  15.50  16.50

STOP AT LINE 100
```

Оператор PRINT в строке 80 осуществляет только возврат каретки перед выводом новой строки.

Таким образом, оператор PRINT USING может обеспечить вывод результатов в любом желаемом формате.

**5.6. Использование функции TAB в операторе PRINT.** Функция TAB может использоваться как один из элементов в списке вывода оператора PRINT. Она вызывает вывод следующего за ней элемента, начиная с позиции, номер которой задан как аргумент функции TAB. Общий вид функции

TAB (*ар. выр.*)

Если значение арифметического выражения содержит дробную часть, то при определении позиции печати дробная часть отбрасывается. Аргумент должен быть заключен в диапазоне от 0 до 71 (по числу позиций строки экрана). Если аргумент функции TAB меньше текущей позиции строки, то эта функция игнорируется, печать осуществляется в следующую позицию. Например, при выполнении оператора

```
PRINT TAB(20);5
```

число 5 выводится в 21-ю позицию строки экрана (20-я позиция отводится под знак числа; см. п. 5.4).

При выполнении оператора

```
PRINT TAB(5);5;TAB(12);89
```

в 6-ю позицию выводится 5; начиная с 13-й — 89.

При выполнении оператора

```
PRINT TAB(20);"*";TAB(30);"0"
```

в 20-ю позицию выводится *, в 30-ю — 0.

При выполнении операторов

```
X=20
PRINT TAB(X);"*"
```

в 20-ю позицию выводится *.

При выполнении операторов

```
X=20
PRINT TAB(X);"*";TAB(X-10);"0"
```

* выведется в 20-ю позицию, 0 — в следующую после *. Вторая функция TAB игнорируется, так как она определяет позицию, меньшую текущей (при печати нельзя вернуться по строке назад).

Функцию TAB можно использовать для вывода результатов в наглядной форме в виде графиков, таблиц, гистограмм (см. работу 7 главы 4).

## 6. Символьные переменные

До сих пор мы имели дело только с переменными, значениями которых являются числа. Их называют *числовыми переменными*. В бейсике имеется возможность использовать также переменные, значениями которых являются строки символов — так называемые *символьные* (или *строковые) переменные*.

Символьные переменные обозначаются так же, как и числовые, т. е. буквой или буквой и цифрой с добавлением символа $ (знак денежной единицы). Например, A$ (читается: А символьное), B1$ и т. д. Длина символьной переменной (количество составляющих ее символов) может изменяться от 0 (пустая строка) до 255 символов.

Задание значений символьных переменных осуществляется:

1. При помощи операторов ввода INPUT или READ, DATA.

В операторе DATA строка символов заключается в кавычки. При использовании оператора INPUT значение символьной переменной при вводе должно заключаться в кавычки, если среди составляющих ее символов есть запятая (,). В остальных случаях использование кавычек не обязательно.

2. Оператором присваивания. Например,

```
A$="МОСКВА"
```

Строка символов в правой части оператора задается в кавычках.

Замечание. При задании значений символьных переменных при помощи операторов присваивания или ввода следует иметь в виду ограничение на количество символов в строке (в одном операторе). Текст, не умещающийся в одной строке, можно задавать с использованием операции сочленения (см. ниже). Например, при выполнении операторов

```
10 PRINT "ВВЕДИТЕ ТЕКСТ. ДЛЯ ОКОНЧАНИЯ"
20 PRINT "ВВОДА ИСПОЛЬЗУЙТЕ **"
30 INPUT A$
40 IF A$="**" GO TO 70
50 T$=T$+A$
60 GO TO 30
70 PRINT T$
```

каждая строка текста вводится в переменную A$, которая затем (строка 50) присоединяется к переменной T$, в которой формируется исходный текст.

Значения символьных величин, как и числовых, можно вывести при помощи операторов PRINT и PRINT USING. В списке вывода может быть указано имя символьной переменной или константа (строка символов), заключенная в кавычки или апострофы. Например, если A$="ИВАНОВ", B$="ПЕТРОВ", то при выполнении оператора

PRINT A$; "И"; B$

будет выведено

ИВАНОВ И ПЕТРОВ

В операторе PRINT USING для вывода символьных величин используются знаки C, L, R или E, которым должен предшествовать апостроф. Использование одного или нескольких следующих подряд знаков C обеспечивает вывод в середину отведенного поля, знаков L — с выравниванием по левой границе поля, R — по правой границе поля, E — с выравниванием строки влево и расширением поля, если необходимо.

Размер отведенного поля определяется количеством следующих после апострофа знаков C (L, R или E). Кроме того, использование самого апострофа в качестве разделителя обусловливает расширение поля еще на одну позицию.

Формат вывода в операторе PRINT USING может быть задан также при помощи символьной переменной, значение которой должно быть определено в программе до первого выполнения оператора PRINT USING.

В качестве примера использования указанных средств рассмотрим следующую программу:

```
10 F$="'CCCC'EEEE'LLLL'RRRR"
20 READ A$
30 IF A$=" " GO TO 80
40 PRINT USING F$,A$,A$,A$,A$
50 GO TO 20
60 DATA "ABCD","ABCDEFG","A"
70 DATA "AB"," "
80 STOP
```

и результат ее выполнения

```
:ABCD :ABCD :ABCD : ABCD:
:ABCDE:ABCDEFG:ABCDE:ABCDE:
:  A  :A     :A    :    A:
:  AB  :AB    :AB   :   AB:
```

Пояснения к программе. Формат, используемый в операторе PRINT USING, задается в строке 10 при помощи символьной переменной F$. При выводе по этому формату выводимые цепочки символов будут разделены двоеточием (:). Под первое значение будет выделено поле в пять позиций, и первая цепочка символов будет выведена в середину этого поля. Если количество символов больше пяти, то последние будут утеряны. Под второе значение выделяются также пять позиций, но если выводимая цепочка длиннее — поле будет расширено.

Под третье и четвертое выводимые значения отводится также по пять позиций с выравниванием при выводе по левой и правой границам поля соответственно.

При выполнении программы в переменную A$ по очереди считываются четыре различных значения, задаваемые в операторах DATA (строки 60, 70). Каждое из этих значений выводится оператором PRINT USING четыре раза в одну строку.

Четыре строки вывода соответствуют четырем различным значениям переменной A$.

Символьные переменные можно сравнивать в условных операторах. Сравнение основано на том, что каждый символ в памяти ЭВМ представляется двоичным кодом и при сравнении символьных переменных происходит сравнение кодов составляющих их символов.

Сравнение символьных переменных осуществляется посимвольно, начиная с первого символа. Если первые символы обеих переменных совпадают, то сравниваются вторые и т. д. Если длины (количество символов) сравниваемых переменных не совпадают, то меньшее дополняется пробелами (пробел является самым младшим символом). (В приложении 1 приведены в порядке старшинства символы бейсика и их восьмеричные коды.)

Например, при выполнении оператора

```
IF A$ > B$ THEN 100
```

если A$="ИВАНОВ", B$="АНДРЕЕВ", условие выполняется и происходит переход к строке 100.

Операции и стандартные функции для символьных величин. Для символьных переменных определена одна операция: сочленение (обозначается + или &). При выполнении этой операции две переменные объединяются в одну.

Например, в результате выполнения операторов

```
10 A$="МОС"       \ B$="КВА"
20 C$=A$+B$
30 PRINT C$
```

на экране будет выведено

МОСКВА

Кроме операции сочленения, для символьных переменных определены стандартные функции.

1. ASC (X$) переводит двоичный код символа в 10-ичное число. Применима только для переменной X$, содержащей один символ. Например,

```
10 X$="Q"
20 PRINT ASC(X$)
```

В результате будет напечатано

81

2. LEN (X$) находит длину символьной переменной (т. е. количество составляющих ее символов, включая конечные пробелы). Например,

```
10 T$="МОСКВА  "
20 X=LEN(T$)
```

После выполнения строки 20 будет X=8 (последними символами являются два пробела).

3. POS (X$, Y$, Z) определяет позицию последовательности символов Y$ в символьной переменной X$, если поиск начинается с позиции Z. Если таких последовательностей несколько, то выбирается первая из них. Если такая последовательность не найдена или строка X$ является пустой, выдается 0; если последовательность Y$ является пустой, выдается позиция Z. Например,

```
10 X$="A*(B+C)-B/(C+A)"
20 P=POS(X$,"(",6)
```

После выполнения строки 20 будет P=11. (Поиск символа "(" начинается с шестой позиции, т. е. в данном тексте с символа C.)

4. SEG$ (X$, Y, Z) выдает последовательность символов из строки X$ с позиции Y до позиции Z. Если Y≤0, то принимается Y=1. Если Z≤0 или Y>LEN (X$), выдается пустая строка. Если Z>LEN (X$), принимается Z= =LEN (X$). Например,

```
10 X$="ГОРОД МОСКВА - СТОЛИЦА СССР"
20 Y$=SEG$(X$,7,12) \ PRINT Y$
```

После выполнения этих операторов будет напечатано

МОСКВА

5. STR$(X) преобразует значение X из числовой формы в символьную. При этом значение X представляется в таком виде, как если бы оно выдавалось на печать с помощью оператора PRINT.

6. TRM (X$) выдает строку X$ с отброшенными конечными пробелами.

7. VAL(X$) преобразует последовательность символов, соответствующую записи числа на бейсике, в число. Например,

```
10 X$="126"        \ S=1
20 P=S+VAL(X$)
```

В результате выполнения этих операторов будет P= =127. Если последовательность символов не является числом, то выдается сообщение об ошибке.

X$, Y$, использованные при определении функций, представляют любые символьные выражения; X, Y, Z — любые арифметические выражения.

*Символьным выражением* называется:

— строка символов, заключенная в кавычки;

— символьная переменная;

— элемент символьного массива;

— символьная функция;

— какие-либо из перечисленных элементов, соединенные знаком + или &.

## 7. Массивы

Переменные, которые мы использовали при составлении программ до сих пор, называются *простыми переменными*. Каждая простая переменная имеет имя, и под каждую такую переменную выделяется ячейка памяти, обращение к которой осуществляется по этому имени. Использование только простых переменных затрудняет или делает вообще невозможным решение многих важных задач.

Рассмотрим пример. Предположим, что в памяти ЭВМ каким-то образом получена (может быть, просто введена) последовательность 10 чисел. Пусть требуется найти максимальное из введенных чисел, а затем разделить каждый элемент последовательности на максимальный элемент.

При определении максимального элемента последовательности можно было бы вводить числа по очереди в одну и ту же ячейку памяти (т. е. использовать одну переменную), а максимальное из уже введенных чисел получать в другой переменной (см. рис. 1.11). Однако при таком способе при поступлении нового числа из последовательности старое стирается (пропадает), и теперь, чтобы решить вторую часть задачи, нужно было бы второй раз вводить всю последовательность. Такой способ становится вообще неприемлемым, если члены последовательности вычисляются самой машиной каким-нибудь трудоемким способом.

Нужно, следовательно, каким-то образом запоминать значения, из которых выбирается максимальный элемент. Способ, который нам уже знаком,— использование переменных. Чтобы запомнить десять чисел, нужно десять различных переменных. Пусть это переменные A, B, C, D, E, F, G, H, S, T. Чтобы найти максимальное из этих значений (P), нужно, положив вначале P=A, сравнивать P по очереди со всеми остальными переменными и заменять P на новое значение, если значение очередной переменной больше P (т. е. максимального из всех предшествующих). Программа, считая, что значения A, . . ., T уже находятся в памяти, будет, например, иметь вид

```
**********
200 P=A
210 IF P)=B GOTO 230
220 P=B
230 IF P)=C GOTO 250
240 P=C
250 IF P)=D GOTO 270
260 P=D
**********
```

и т. д., пока не будут просмотрены все переменные.

Изучая эту программу, можно видеть, что она состоит из повторяющихся пар операторов, отличающихся только именем использованной в них переменной. Если бы мы имели возможность указать какой-либо способ перехода к ячейке памяти, в которой находится значение следующей переменной, то в программе можно было бы написать только одну пару операторов, подобных операторам 210, 220 или

230, 240. Такую возможность предоставляет использование массивов.

*Массивом* называется упорядоченная последовательность величин, обозначаемая одним именем. Упорядоченность заключается в том, что элементы массива располагаются в последовательных ячейках памяти.

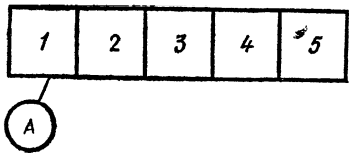



Рис. 3.1

Массив, вспоминая аналогию с ящиками, можно представить себе как несколько одинаковых ящиков, скрепленных вместе. На всю конструкцию как целое повешен один ярлык с именем массива (например, А). Все ящики в одной такой конструкции перенумерованы, начиная с 1. Ящики — это элементы массива. Номер ящика — значение индекса элемента массива (рис. 3.1). Чтобы получить доступ к нужному ящику, нужно указать *имя массива* и его *индекс*. Индекс записывается в круглых скобках после имени массива. Если нужно присвоить, например, значение 5 третьему элементу массива А, нужно написать

```
A(3)=5
```

При выполнении этого оператора будет найден массив А, отсчитан третий ящик (ячейка памяти) с начала и в него положено значение 5. Если нужно всем пяти элементам массива А присвоить значение 5, можно написать

```
30 A(1)=5
40 A(2)=5
50 A(3)=5
60 A(4)=5
70 A(5)=5
```

Индекс может быть переменной, поэтому то же самое можно выполнить и другим способом:

```
10 FOR I=1 TO 5
20 A(I)=5
30 NEXT I
```

Мы, таким образом, имеем простой способ перехода от одного элемента массива к другому, изменяя индексы.

Имя массива образуется так же, как имя переменной (буква или буква и цифра). В бейсике массивы могут быть одномерные и двумерные. В двумерном массиве каждый эле-

мент идентифицируется номером строки и номером столбца, на пересечении которых он расположен. Положение элемента в массиве определяется индексами: одним — для одномерных массивов и двумя — для двумерных. При записи элемента массива индекс (индексы) записывается в круглых скобках после имени массива. Если индексов два, то они разделяются запятой. В качестве индексов могут употребляться любые арифметические выражения. Если значение арифметического выражения содержит дробную часть, то при определении индекса она отбрасывается. Появление отрицательного значения индекса вызывает сообщение об ошибке.

Переменные с индексами (элементы массива) могут употребляться в программе так же, как простые переменные (в левой части оператора присваивания, в арифметических выражениях и т. п.).

Под массивы, так же как и под переменные, машина должна выделять память. Вспомним, что под переменную одна ячейка памяти выделялась, как только эта переменная первый раз встречалась в программе. Для массивов такой способ не годится. Ведь машина должна выделить память сразу под весь массив. А при первом появлении переменной с индексом в программе еще не известно, сколько ячеек памяти требуется выделить под весь массив. Поэтому машина должна получить эту информацию отдельно. Ей нужно сообщить, какие массивы будут использованы в программе и каков размер каждого массива. Эта информация задается в операторе описания массивов DIM. В операторе DIM указываются имя массива и в круглых скобках верхние границы изменения индексов, которые должны быть целыми положительными числами (нижняя граница фиксирована и равна 1*). В бейсике могут использоваться массивы как числовые, так и символьные. Например, оператор

```
DIM A(3),B(4,5),T$(12)
```

описывает два числовых массива: A, состоящий из трех элементов (A(1), A(2), A(3)), и B, содержащий 4 строки и 5 столбцов, и символьный массив T$, состоящий из 12 элементов.

В соответствии с оператором DIM в памяти ЭВМ выделяется место для размещения этих массивов. Так, для массива A будут выделены три ячейки памяти, для массива B — двадцать ячеек, для массива T$ — двенадцать ячеек па-

*) См. замечание 2.

мяти. Под двумерный массив выделяется линейный участок памяти, в котором массив располагается по строкам.

Описание массива должно появляться в программе до первого обращения к элементам этого массива. Для улучшения структуры программы описание всех массивов рекомендуется давать в начале программы.

Замечания. 1. В бейсике допускается использование в одной программе одинаковых имен для обозначения простых переменных и массивов. Однако использование этой возможности на практике приводит к затруднению восприятия программы и может служить источником трудно-обнаруживаемых ошибок. Поэтому настоятельно не рекомендуется использовать одинаковые имена для переменных и массивов.

2. В рассматриваемом варианте языка бейсик фиксированное минимальное значение индексов равно 0 (а не 1, как указано выше), что является некоторым расширением возможностей языка по сравнению с используемым вариантом. Так, в соответствии с оператором DIM C (10), B (4, 5) машина всегда выделяет 11 и 30 ячеек памяти для размещения массивов C и B соответственно. Однако нумерация элементов с нуля для большинства задач неудобна и в дальнейшем изложении не используется. Все элементы массивов, имеющие индексы, равны нулю; будем далее игнорировать без дополнительных замечаний, т. е. будем считать, например, что в соответствии с описанием массивов

```
DIM C(10),B(4,5)
```

массив C содержит десять элементов, массив B содержит четыре строки и пять столбцов.

3. Если описание массива отсутствует, то по умолчанию максимальное значение каждого индекса равно 10, т. е. под одномерный массив выделяются десять ячеек памяти, под двумерный — 100 (10×10).

В бейсике не определены операции с массивами. Поэтому любая обработка массивов, а также ввод-вывод массивов осуществляется поэлементно.

Например, ввод одномерного массива A, содержащего 10 элементов, можно осуществить при помощи операторов

```
10 DIM A(10)
20 FOR I=1 TO 10
30 INPUT A(I)
40 NEXT I
```

Ввод двумерного массива B размером 4×5 можно осуществить при помощи операторов

```
10 DIM B(4,5)
20 FOR I=1 TO 4
30 FOR J=1 TO 5
40 INPUT B(I,J)
50 NEXT J
60 NEXT I
```

При одном выполнении оператора INPUT вводится один элемент массива (после набора на клавиатуре значения очередного элемента массива нужно нажать клавишу ВК).

Ввод двумерного массива в приведенной программе осуществляется по строкам.

Вывод двумерных массивов для повышения наглядности рекомендуется осуществлять по строкам, начиная вывод каждой новой строки массива в новую строку экрана. Например, вывод массива B размером 4×5 можно осуществить при помощи операторов

```
10 DIM B(4,5)
.............
150 FOR I=1 TO 4
160 FOR J=1 TO 5
170 PRINT B(I,J);
180 NEXT J
190 PRINT
200 NEXT I
.............
```

Элементы одной строки будут выводиться в плотном формате. Оператор 190 осуществляет возврат каретки перед печатью новой строки.

Пример. В сессию 10 студентов одной группы сдали по 5 экзаменов. Результаты сессии представлены в виде матрицы M (10×5). Определить средний балл по каждому предмету. Результаты получить в виде одномерного массива C, содержащего 5 элементов.

Решение. Просуммируем элементы каждого столбца матрицы M и, разделив на число студентов, поместим результат в соответствующий элемент массива C (см. программу 3.6).

Для наглядности исходные данные задаются в программе в операторах DATA. В одном операторе DATA заданы оценки всех студентов по одному предмету, т. е. элементы одного столбца матрицы M. Ввод матрицы M осуществляется по столбцам.

**Программа 3.6**

```
10 DIM M(10,5),C(5)
20 DATA 5,4,5,3,4,3,3,3,4,4
30 DATA 5,5,3,5,3,4,5,4,5,4
40 DATA 3,4,5,3,4,5,3,3,3,3
50 DATA 4,4,4,4,4,5,4,4,3,3
60 DATA 4,3,3,3,5,4,3,3,5,5
70 FOR J=1 TO 5
71 FOR I=1 TO 10 \ READ M(I,J) \ NEXT I
72 NEXT J
80 FOR J=1 TO 5
90 S=0
100 FOR I=1 TO 10 \ S=S+M(I,J) \ NEXT I
110 C(J)=S/10
120 NEXT J
130 PRINT '   ТАБЛИЦА СРЕДНИХ БАЛЛОВ ПО ЭКЗАМЕНАМ'
140 FOR I=1 TO 5 \ PRINT C(I), \ NEXT I
150 PRINT
160 STOP

RUNNH

   ТАБЛИЦА СРЕДНИХ БАЛЛОВ ПО ЭКЗАМЕНАМ
 3.8           4.3           3.6           3.9           3.8

STOP AT LINE 160
```

Различные способы в/в массивов, а также простейшие алгоритмы обработки массивов приведены во введении к работе 4 (глава 4).

## 8. Матричные операторы

При решении различных задач на ЭВМ одной из основных структур представления данных являются массивы: одномерные — векторы — и двумерные — матрицы. Существует расширение бейсика, предоставляющее дополнительные средства для работы с массивами — *матричные операторы*. Матричные операторы позволяют компактно (при помощи одного оператора) описывать различные действия, выполняемые с массивами. Это делает программу короче и придает ей более ясную структуру. Кроме того, обработка массивов с использованием матричных операторов обеспечивает увеличение скорости обработки по сравнению с аналогичными программными вариантами на языке бейсик без их использования. При использовании матричных операторов наименованием массива может быть только одна латинская буква.

Матричные операции выполняются оператором MAT (сокращение слова MATRIX — матрица). Все матричные

операции, реализуемые с помощью оператора MAT, можно разделить на три группы:

— задание начальных значений элементов матрицы (вектора);

— выполнение алгебраических действий над матрицами (векторами);

— печать элементов матриц (векторов).

К первой группе относятся следующие операции:

Ввод матриц (векторов).

```
MAT INPUT A1 [, A2, . . .]
```

или

```
MAT READ A1 [, A2, . . .]
```

где A1, A2, . . . — имена массивов.

Первый из этих операторов осуществляет ввод элементов массивов с клавиатуры, второй — из области DATA. Заполнение массивов производится по очереди: сначала заполняется массив A1, затем A2 и т. д. Двумерные массивы заполняются по строкам.

Очистка матрицы (вектора).

```
MAT C=ZER
```

После выполнения этой операции все элементы массива C равны нулю.

Заполнение матрицы (вектора) единицами.

```
MAT C=CON
```

После выполнения этой операции все элементы массива C равны 1.

Задание единичной матрицы.

```
MAT C=IDN
```

После выполнения этой операции единицами заполняются диагональные элементы матрицы C, все остальные элементы — нулевые. Матрица C должна быть квадратной.

Ко второй группе относятся следующие операции:

Матричное присваивание.

```
MAT C=A
```

После выполнения этой операции матрица (вектор) C становится равной матрице (вектору) A. Размерности A и C должны совпадать.

Сложение (вычитание) матриц (векторов).

MAT C=A+B
MAT C=A — B

После выполнения операции значение каждого элемента массива C равно сумме (разности) соответствующих элементов массивов A и B. Размерности всех массивов должны совпадать.

Умножение матрицы (вектора) на скаляр.

MAT C=(K)*A

где K — арифметическое выражение. В результате выполнения этой операции вычисляется значение K и каждый элемент массива A умножается на K.

Умножение матриц.

MAT C=A*B

Умножать можно только матрицы, у которых число столбцов первого сомножителя равно числу строк второго сомножителя. Имя матрицы результата не должно совпадать с именами сомножителей.

Транспонирование матрицы.

MAT C=TRN(A)

(TRN — служебное слово (от TRANSPOSE — транспонировать)). Операция транспонирования — это замена строк матрицы ее столбцами, а столбцов — строками.

Вычисление обратной матрицы.

MAT C=INV(A)

A, C — квадратные матрицы. В результате выполнения этой операции вычисляется матрица C, обратная A.

К третьей группе относится операция:

Печать матрицы (вектора).

MAT PRINT A1, A2, . . .

В результате на экран дисплея выводятся элементы массивов A1, A2, . . . Вывод двумерных массивов осуществляется по строкам, одномерный массив выводится в столбец.

Вывод массива осуществляется в зонном формате, если в списке вывода оператора MAT PRINT один элемент от другого отделяется запятой или отсутствует разделитель в конце списка. Вывод новой строки матрицы осуществляется в новую строку экрана.

Вывод осуществляется в плотном формате, если после имени массива и в конце списка используется точка с запятой.

## 9. Подпрограммы

Когда некоторая совокупность действий должна выполняться в нескольких различных местах программы, то обычно нежелательно каждый раз повторять группу операторов, реализующих эти действия. Чтобы избежать повторений, указанную группу операторов можно записать в программе один раз и обращаться к ней, когда в этом возникает необходимость. Обособленную группу операторов, которую можно выполнять многократно, обращаясь к ней из различных мест программы, называют *подпрограммой*.

Чтобы подпрограмма при обращении к ней выполнялась каждый раз с новыми данными, ее нужно составить в общем виде, а исходные данные для работы передавать в переменные подпрограммы перед обращением к ней. Если, например, в программе требуется решить три квадратных уравнения с различными коэффициентами, то алгоритм нахождения корней квадратного уравнения целесообразно оформить в виде подпрограммы, используя для обозначения коэффициентов переменные. Перед каждым обращением к подпрограмме нужно задать этим переменным числовые значения, соответствующие коэффициентам решаемых уравнений.

Использование подпрограмм уменьшает общее количество операторов в программе, и, следовательно, для размещения программы требуется меньше памяти. Время на выполнение программы при этом практически не изменяется.

Подпрограммы обладают также некоторыми другими преимуществами. Использование подпрограмм улучшает структуру программы. Кроме того, облегчается отладка программы, так как работа каждой подпрограммы может быть проверена по отдельности. Многие подпрограммы имеют дополнительную ценность, поскольку ими может воспользоваться не только тот, кто написал подпрограмму, но и другие лица.

Большинство машин располагает обширными библиотеками стандартных подпрограмм *), которые существенно облегчают программирование реальных задач.

В бейсике подпрограмма оформляется как группа операторов, которая должна выполняться при обращении к ней, и составляет с программой единое целое.

---

*) Стандартные подпрограммы — организованные и оформленные стандартным образом готовые подпрограммы, которые при задании соответствующих параметров могут использоваться для решения типовых задач. Имеющийся набор стандартных подпрограмм, организованный соответствующим образом, называется *библиотекой стандартных подпрограмм.*

Обращение к подпрограмме осуществляется оператором

GOSUB N

где N — номер строки, с которой начинается подпрограмма.

Подпрограмма размещается в последовательных строках, начиная с N-й. Последним оператором, выполняемым в подпрограмме, должен быть оператор RETURN. По оператору RETURN осуществляется возврат в то место программы, из которого произошло обращение к подпрограмме, а именно к оператору, следующему за GOSUB N. Подпрограмма может содержать обращения к другим подпрограммам. При использовании подпрограмм следует следить за тем, чтобы

1. Перед обращением к подпрограмме ее переменные получили нужные значения.

2. Подпрограмма не начала выполняться без обращения к ней, что вызовет ошибку.

П р и м е р. Составить программу для вычисления числа сочетаний

$$C_n^m = \frac{n!}{m!\,(n-m)!}.$$

Вычисление факториала оформить в виде подпрограммы (см. программу 3.7).

*Список используемых переменных.* Исходные данные: N, M — целые неотрицательные числа, N≥M.

Результат: C — число сочетаний из N по M.

Вспомогательные переменные: C1=N!, C2=M!.

Переменные в подпрограмме: L — переменная, факториал которой вычисляется в подпрограмме, P=L! вычисляется в подпрограмме, I — управляющая переменная цикла.

**Программа 3.7**

```
10 PRINT "ВВЕДИТЕ N, M"
20 INPUT N,M
30 L=N \ GOSUB 1000 \ C1=P
40 L=M \ GOSUB 1000 \ C2=P
50 L=N-M \ GOSUB 1000
60 C=C1/(C2*P)
70 PRINT "ЧИСЛО СОЧЕТАНИЙ ИЗ ";N;"ПО";M;"=";C
80 STOP
1000 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ФАКТОРИАЛА
1010 P=1
1020 FOR I=1 TO L
1030 P=P*I
1040 NEXT I
1050 RETURN
```

Пояснения к программе. Подпрограмма располагается в строках с номерами 1000—1050. Оператор REM (строка 1000) не является обязательным, он просто зрительно выделяет подпрограмму. Никаким другим способом начало подпрограммы не помечается.

Обращение к подпрограмме осуществляется три раза при помощи операторов GOSUB 1000. Перед обращением в переменную L, используемую в подпрограмме, передаются соответственно значения N, M, N — M, для которых вычисляется факториал. Переменные C1 и C2 служат для запоминания вычисленных значений N! и M!, которые в противном случае были бы затерты значениями P, вычисляемыми при последующих обращениях к подпрограмме.

Пример. Три группы студентов, в каждой не более 25 человек, в сессию сдавали по 5 экзаменов. Определить лучшую по среднему баллу группу.

Определение среднего балла студентов одной группы осуществить в подпрограмме.

Решение. Число студентов в каждой из групп обозначим через N1, N2, N3. Оценки студентов каждой группы зададим в виде матриц M1, M2, M3, содержащих 5 столбцов (по числу сданных экзаменов) и не более 25 строк (максимальное число студентов в одной группе) (см. программу 3.8).

Для определения среднего балла требуется найти сумму всех элементов матрицы оценок одной группы и разделить ее на число элементов этой матрицы, равное числу студентов, умноженному на число экзаменов.

В подпрограмме используем обозначения: N — число студентов, M — матрица оценок. Перед каждым обращением к подпрограмме будем пересылать в N число студентов, а в M — матрицу оценок конкретной группы.

Для наглядности все исходные данные заданы в операторах DATA: в операторах 30, 100 и 170 — количество студентов в каждой группе, в операторах 40—80 — оценки студентов первой группы (один оператор DATA содержит оценки студентов одной группы по одному предмету), 110—150 — оценки второй группы, 180—220 — оценки третьей группы.

Массивы M1, M2, M3 при вводе заполняются по столбцам (строки 240—260, 280—300, 320—340).

Для того чтобы исключить выполнение подпрограммы (строки 1000—1050) без обращения к ней, после выполнения любой из трех возможных ветвей (строки 530, 550, 560) выполняется оператор STOP.

## Программа 3.8

```
10 DIM M1(25,5),M2(25,5),M3(25,5),M(25,5)
20 REM В ПЕРВОЙ ГРУППЕ 10 СТУДЕНТОВ
30 DATA 10
40 DATA 3,3,5,4,3,4,4,5,5,5
50 DATA 4,4,4,5,5,5,3,4,5,3
60 DATA 3,3,3,3,3,5,5,5,5,5
70 DATA 4,4,5,5,4,5,4,5,4,5
80 DATA 3,5,3,4,5,3,4,3,4,4
90 REM ВО ВТОРОЙ ГРУППЕ 5 СТУДЕНТОВ
100 DATA 5
110 DATA 4,5,3,3,5
120 DATA 3,5,3,5,3
130 DATA 4,4,4,4,5
140 DATA 3,5,4,3,5
150 DATA 5,3,5,4,5
160 REM В ТРЕТЬЕЙ ГРУППЕ 8 СТУДЕНТОВ
170 DATA 8
180 DATA 4,3,5,4,3,4,5,4
190 DATA 4,5,4,4,3,3,5,3
200 DATA 3,5,3,3,3,3,4,5
210 DATA 5,5,5,4,3,5,5,5
220 DATA 4,3,5,4,5,4,5,5
230 READ N1
240 FOR J=1 TO 5
250 FOR I=1 TO N1 \ READ M1(I,J) \ NEXT I
260 NEXT J
270 READ N2
280 FOR J=1 TO 5
290 FOR I=1 TO N2 \ READ M2(I,J) \ NEXT I
300 NEXT J
310 READ N3
320 FOR J=1 TO 5
330 FOR I=1 TO N3 \ READ M3(I,J) \ NEXT I
340 NEXT J
350 REM НАЧИНАЕМ СЧИТАТЬ СРЕДНИЕ БАЛЛЫ ПО ГРУППАМ
360 FOR I=1 TO N1
370 FOR J=1 TO 5 \ M(I,J)=M1(I,J) \ NEXT J
380 NEXT I
390 N=N1 \ GOSUB 1000 \ P1=P
400 PRINT "СРЕДНИЙ БАЛЛ 1-ОЙ ГРУППЫ -";P1
410 FOR I=1 TO N2
420 FOR J=1 TO 5 \ M(I,J)=M2(I,J) \ NEXT J
430 NEXT I
440 N=N2 \ GOSUB 1000 \ P2=P
450 PRINT "СРЕДНИЙ БАЛЛ 2-ОЙ ГРУППЫ -";P2
```

```
460 FOR I=1 TO N3
470 FOR J=1 TO 5 \ M(I,J)=M3(I,J) \ NEXT J
480 NEXT I
490 N=N3 \ GOSUB 1000 \ P3=P
500 PRINT "СРЕДНИЙ БАЛЛ 3-ЕЙ ГРУППЫ -";P3
510 IF P1<P2 GO TO 540
520 IF P1<P3 GO TO 560
530 PRINT 'ГРУППА 1 ЛУЧШАЯ ПО СРЕДНЕМУ БАЛЛУ' \ STOP
540 IF P2<P3 THEN 560
550 PRINT 'ГРУППА 2 ЛУЧШАЯ ПО СРЕДНЕМУ БАЛЛУ' \ STOP
560 PRINT 'ГРУППА 3 ЛУЧШАЯ ПО СРЕДНЕМУ БАЛЛУ' \ STOP
1000 REM ПОДПРОГРАММА ОПРЕДЕЛЕНИЯ СРЕДНЕГО БАЛЛА В ГРУППЕ
1010 P=0
1020 FOR I=1 TO N
1030 FOR J=1 TO 5 \ P=P+M(I,J) \ NEXT J
1040 NEXT I
1050 P=P/(5*N) \ RETURN

RUNNH

СРЕДНИЙ БАЛЛ 1-ОЙ ГРУППЫ - 4.12
СРЕДНИЙ БАЛЛ 2-ОЙ ГРУППЫ - 4.08
СРЕДНИЙ БАЛЛ 3-ЕЙ ГРУППЫ - 4.1
ГРУППА 1 ЛУЧШАЯ ПО СРЕДНЕМУ БАЛЛУ

STOP AT LINE 530
```

## 10. Определение нестандартных функций оператором DEF FN

Помимо стандартных функций (см. таблицу 3.1), в программе можно определить и далее использовать другие (нестандартные) функции. Определить функцию можно оператором DEF FN. Общий вид оператора

DEF FN$v$ ($x_1$ [, $x_2$, ...]) = *ар. выр.*,

где $v$ — имя функции — прописная латинская буква; $x_1$, $x_2$, ... — простые переменные — формальные аргументы функции; *ар. выр.*— формула, по которой вычисляется функция.

Арифметическое выражение в правой части должно обязательно содержать хотя бы один из формальных аргументов $x_1$, $x_2$, ..., но может содержать также и другие переменные, общие для всей программы.

Оператор DEF FN должен располагаться в программе до первого использования определяемой им функции.

Вычисление функции, описанной оператором DEF FN, осуществляется при обращении к ней при помощи записи

указателя функции:

FN$v$ ($a_1$ [, $a_2$, ...]),

где $a_1$, $a_2$, ... — арифметические выражения, заменяющие формальные аргументы $x_1$, $x_2$, ... в арифметическом выражении правой части оператора DEF FN перед вычислением.

Использование функции, заданной оператором DEF FN (нестандартной функции), аналогично использованию стандартных функций (см. п. 3.3). Обращение к нестандартной функции (указатель функции) можно записать в арифметическом выражении, в списке вывода оператора PRINT и т. д., вообще везде, где требуется значение этой функции.

Пример

```
10 DEF FNT(X)=X*X+A
20 A=5
50 P=FNT(3)
```

Перед вычислением значения функции FNT формальный аргумент X заменяется на фактический аргумент 3, указанный в обращении к функции, и с этим значением X (X=3) будет вычислено значение выражения X*X+A. В результате в строке 50 переменной P будет присвоено значение 14.

Пример.

```
10 DEF FNV(X,Y)=X*X+Y*Y+A*A
20 A=2          \ T=3.1 \ Z=8.6
100 P=0.5*SQR(FNV(SIN(T/PI),2*Z))
```

В арифметическом выражении правой части оператора, расположенного в строке 100, имеется обращение к функции FNV, используемой в качестве аргумента стандартной функции SQR. При вычислении функции FNV сначала будут вычислены арифметические выражения SIN (T/PI) и 2*Z, значения которых заменят формальные аргументы X и Y в арифметическом выражении, определяющем функцию FNV (X, Y), а затем вычислено значение этого арифметического выражения, т. е. значение функции FNV, которое заменит обращение к ней в выражении, являющемся аргументом функции SQR, т. е. фактически при вычислении значения P в строке 100 будет выполнена следующая последовательность действий:

X=SIN (T/PI)
Y=2*Z
V=X*X+Y*Y+A*A
P=0.5*SQR (V)

## 11. Вычисляемые переходы. Операторы ON

Рассмотренные ранее операторы GO TO N и GOSUB N позволяют перейти только в одно место программы — к строке с номером N. Это так называемый безусловный переход. Использование операторов ON позволяет выбрать (вычислить) один из нескольких возможных номеров строк, по которому передается управление. Таким образом, переход зависит от результата выполнения предыдущей части программы.

Общий вид операторов

ON *ар. выр.* GO TO $N_1$ [, $N_2$, . . .]

и

ON *ар. выр.* GOSUB $N_1$ [, $N_2$, . . .].

При выполнении этих операторов вычисляется значение арифметического выражения и округляется до целой части. Если полученное значение равно 1, осуществляется переход к строке с номером $N_1$; если 2 — то к строке с номером $N_2$ и т. д. Если значение выражения меньше 1 или больше числа указанных номеров строк, то выдается ошибка.

Пример.

```
40 I=2
50 ON I GO TO 100,150,300
```

В результате будет совершен переход к строке 150.

Пример. Составить программу для вычисления площади одной из трех фигур — квадрата, круга или равностороннего треугольника — по значению X, интерпретируемому как сторона квадрата, радиус окружности или сторона треугольника (см. программу 3.9).

**Программа 3.9**

```
10 PRINT "ВВЕДИТЕ ДЛЯ ВЫЧИСЛЕНИЯ ПЛОЩАДИ :"
20 PRINT "КВАДРАТА - 1, КРУГА -2, ТРЕУГОЛЬНИКА - 3"
30 INPUT K
40 PRINT "ВВЕДИТЕ РАЗМЕР X"
50 INPUT X
60 ON K GO TO 70,80,90
70 PRINT "ПЛОЩАДЬ КВАДРАТА";X*X \ GO TO 100
80 PRINT "ПЛОЩАДЬ КРУГА";PI*X*X \ GO TO 100
90 PRINT "ПЛОЩАДЬ ТРЕУГОЛЬНИКА";X*X*SQR(3)/4
100 STOP
```

Пояснения к программе. Если при выполнении оператора INPUT (строка 30) вводится значение K, равное 1, то после ввода значения X (строка 50) в результате выполнения оператора ON K GO TO (строка 60) осуществляется переход к строке, номер которой является

первым в списке, т. е. к строке 70, вычисляется площадь квадрата, ее значение выводится на экран и выполнение программы заканчивается. Если вводится значение K=2, то в результате выполнения оператора ON K GO TO (строка 60) осуществляется переход к строке, номер которой является вторым в списке, т. е. к строке 80, вычисляется и выводится площадь круга. И наконец, если вводится K=3, то переход осуществляется к строке 90, вычисляется и выводится площадь треугольника.

## 12. Ввод, исправление, выполнение программы. Работа за дисплеем

**12.1. Ввод и выполнение программы.** Ввод программы можно начинать, если на экране высвечивается READY, а в начале следующей строки находится *курсор* — специальный знак, указывающий позицию, в которую будет осуществляться ввод очередного символа с клавиатуры.

Программа вводится по строкам. Для ввода одной строки нужно набрать на клавиатуре номер строки и, далее, содержащиеся в ней операторы, отделяя один оператор от другого, если их несколько, символом \. Допустимое количество символов в строке программы определяется шириной экрана дисплея. Однако не следует стремиться всегда заполнять строку полностью. При разделении программы на строки нужно исходить из соображений удобства восприятия программы и внесения в нее исправлений. При вводе строки пробелы могут вводиться произвольно, это не влияет на выполнение введенной строки. Не допускается использование пробелов в именах переменных и в служебных словах. При высвечивании (или распечатке) текста программы машина игнорирует пробелы, введенные пользователем, и расставляет их стандартным образом.

Ввод набранной строки в оперативную память осуществляется нажатием клавиш ВК (возврат каретки) (или CR в зависимости от клавиатуры). Если при наборе строки сделана ошибка, то, используя клавишу ЗБ (забой) (или RUBOUT), можно стереть ошибочные символы в конце строки и набрать правильные. Всю строку (до нажатия клавиши ВК) можно стереть, нажимая одновременно клавиши СУ (или CTRL) и U (лат.) (это — одна из клавишных команд (см. п. 12.4), которая обозначается СУ/U).

После ввода строки в память (нажатия клавиши ВК) курсор перемещается в начало строки экрана, и можно приступить к набору следующей строки. Из двух строк,

введенных с одинаковыми номерами, в памяти сохраняется последняя.

После ввода всей программы ее можно выполнить командой RUN или RUNNH.

При выполнении команды RUN на экран выводится заголовок, содержащий имя выполняемой программы (см. п. 12.3), дату и текущее время, и после этого начинается выполнение программы.

Команда RUNNH аналогична команде RUN, но заголовок при ее выполнении не выводится.

**12.2. Исправление программы.** В программу, находящуюся в памяти, можно внести следующие исправления:

1. *Вставить новую строку* между соседними строками с номерами $N_1$ и $N_2$. Для этого на клавиатуре набирается пропущенная строка с любым номером, лежащим между $N_1$ и $N_2$, которая автоматически будет размещена в программе на нужном месте.

2. *Стереть строку*. Для этого нужно набрать ее номер и нажать клавишу ВК или воспользоваться командой DEL.

Разновидности команды DEL:

DEL—N — стирает строки от начала программы до строки N включительно;

DEL $N_1$—$N_2$ — стирает строки с $N_1$ до $N_2$;

DEL N— —стирает строки с N-й до конца программы;

DEL N — стирает строку N.

3. *Заменить строку*. Для этого нужно набрать новую строку с тем же номером.

4. *Внести изменения в строку*. Для этого используется команда SUB. Общий вид команды

SUB N $\$\alpha_1$ $\$\alpha_2$ $\$M$

где N — номер исправляемой строки; $\alpha_1$ — последовательность символов, которую нужно заменить; $\alpha_2$ — последовательность символов, на которую нужно заменить последовательность $\alpha_1$; $ — разделитель (любой символ, не входящий в $\alpha_1$ и $\alpha_2$); M — номер вхождения последовательности $\alpha_1$, которую нужно заменить. Если M=1, то $M можно опустить.

Например, в строке

30 A=5\B=3\A=4

допущена ошибка в последнем операторе, где вместо A должно быть C. Эту ошибку можно исправить командой

SUB30/A/C/2

После внесения в программу исправлений, связанных с добавлением или удалением строк, обычно строки перенумеровываются стандартным образом так, чтобы первая строка имела номер 10, а номера последующих строк увеличивались с шагом 10. Это выполняется командой RESEQ.

В общем случае команда RESEQ имеет вид

RESEQ [$N_1$] [,$N_2$] [—$N_3$] [,*ар. выр.*]

где $N_2$ — $N_3$ — интервал номеров строк, которые должны быть перенумерованы; $N_1$ — новый номер первой из перенумеруемых строк ($N_2$); *ар. выр.*— выражение, целая часть которого задает шаг приращения.

Если *ар. выр.* отсутствует, то шаг приращения полагается равным 10. Если $N_1$ отсутствует, то новый номер первой из перенумеруемых строк полагается равным ближайшему номеру перед $N_2$ плюс приращение. Если $N_2$ отсутствует, перенумерация осуществляется с начала программы до строки $N_3$. Если $N_3$ отсутствует, то перенумерация начинается с $N_2$ до конца программы.

**12.3. Просмотр программы на экране. Очистка памяти. Задание имени программы.** Введенную программу можно высветить на экране при помощи команды LIST. При этом строки программы будут расположены на экране в порядке возрастания номеров независимо от последовательности их ввода.

Разновидности команды LIST:

LIST — выводит программу с заголовком; если программа не умещается на экране, то на нем остаются ее последние строки;
LIST —N — выводит строки до N-й включительно;
LIST N— — выводит строки, начиная с N-й;
LIST $N_1$ —$N_2$ — выводит строки с $N_1$-й до $N_2$-й;
LIST N — выводит N-ю строку;
LISTNH — выполняется аналогично команде LIST, но заголовок не выводится.

Если на протяжении одного сеанса заканчивается работа с одной программой и предполагается ввод другой, то оперативная память перед вводом новой программы должна быть очищена. Вообще говоря, если новая программа не короче (по числу строк), чем находящаяся в данный момент в оперативной памяти, и строки вводятся с теми же номерами, то после ввода новой программы в памяти не останется никаких следов старой программы. Если же строки вводятся с другими номерами или новая программа

содержит меньше строк, то часть строк старой программы сохранится (не будет стерта). Попытка выполнить такую программу, в которой присутствуют строки из разных программ, обязательно приведет к ошибкам. Во избежание этого целесообразно придерживаться следующих правил:

— перед вводом новой программы очистить память;

— перед выполнением введенной программы просмотреть ее на экране при помощи команды LIST.

Очистить память можно одной из двух команд NEW или SCR.

При помощи команды NEW можно, кроме того, задать *имя программы*, которую предполагается вводить. Имя программы может быть задано как набор от 1 до 6 символов, включающий прописные латинские буквы и цифры и начинающийся с буквы. Порядок выполнения команды NEW следующий. После ввода служебного слова NEW на экране высвечивается

NEW FILE NAME—

Далее нужно набрать имя программы и нажать клавишу ВК или просто нажать ВК. В последнем случае программа будет иметь стандартное имя NONAME.

Команда SCR только очищает память. Если далее вводится программа, то она будет иметь стандартное имя NONAME.

При помощи команды RENAME можно изменить имя программы, находящейся в оперативной памяти. Общий вид оператора

RENAME *имя*

Таким образом, программа, находящаяся в оперативной памяти, всегда имеет имя: стандартное — NONAME или имя, заданное в команде NEW (или RENAME).

**12.4. Клавишные команды.** Клавишные команды задаются одновременным нажатием двух клавиш СУ (или CTRL в зависимости от клавиатуры) и одной из клавиш, соответствующих латинским буквам C, D, S, Q, U. За каждой из перечисленных клавиш закреплена определенная функция. К клавишным командам относится также команда, выполняемая при нажатии клавиши ЗБ.

Клавишные команды позволяют оперативно вмешиваться в выполнение программы или команды, а также используются для исправления ошибок набора при вводе программы. Ниже перечисляются функции клавишных команд и даются необходимые пояснения.

ЗБ стирает последний введенный символ.

СУ/U стирает текущую (набираемую) строку.

СУ/С прекращает выполнение программы или команды (команда вводится двукратным нажатием клавиш СУ/С). После выполнения этой команды на экране появляется READY и система переходит в состояние ожидания следующей команды. Использование этой команды необходимо, когда естественное завершение программы не происходит, например, из-за зацикливания или программа выполняется слишком долго, или при выполнении оператора INPUT неправильно набраны данные и нажатие клавиши ВК не приводит к продолжению выполнения программы, и т. п.

СУ/D прекращает вывод на терминал. Выполнение программы или команды продолжается до завершения, после чего на экране появляется READY.

СУ/S временно прекращает вывод на терминал. Выполнение программы или команды прерывается.

СУ/Q возобновляет вывод на терминал (отменяется действие СУ/S).

**12.5. Режим непосредственного исполнения и его использование при отладке программ.** Режим непосредственного исполнения позволяет использовать ЭВМ как обычный калькулятор, а также в процессе отладки программы для контроля значений переменных и проведения промежуточных вычислений.

В режиме непосредственного исполнения строка набирается без номера и выполняется сразу же после ввода в ЭВМ. Очевидно, что в этом режиме не могут использоваться операторы, содержащие ссылку на номер строки.

Пример.

```
A=2    \ PRINT A*A;A*A*A
```

После выполнения этой строки будет напечатано

```
 4  8
```

или

```
FOR I=1 TO 5    \ PRINT I,SQR(I) \ NEXT I
```

В результате выполнения этой строки будет напечатано

```
1       1
2       1.41421
3       1.73205
4       2
5       2.23607
```

Для отладки программы в нее рекомендуется включить несколько операторов STOP в наиболее ответственных точках программы. По оператору STOP программа будет остановлена, и в режиме непосредственного исполнения можно просмотреть промежуточные результаты выполнения программы, провести для проверки необходимые расчеты, после чего внести необходимые изменения в программу или изменить значения переменных и продолжить выполнение программы по команде GO TO N.

## 13. Работа с внешними устройствами

К внешним устройствам рассматриваемых типов ЭВМ относятся *дисплеи*, *печатающее устройство* (ПУ), *внешние запоминающие устройства*. В качестве внешних запоминающих устройств чаще всего используются *магнитные диски* (НМД — накопители на магнитных дисках или ГМД — гибкие магнитные диски (дискеты)) и *магнитные ленты*.

**13.1. Использование печатающего устройства.** При помощи ПУ можно напечатать на бумажной ленте текст программы, находящейся в оперативной памяти, или результаты работы программы. (Вспомним, что стандартно результаты работы программы выводятся на экран дисплея.)

Перед началом вывода на ПУ нужно убедиться в том, что ПУ подключено к машине.

Вывод текста программы, находящейся в оперативной памяти, на ПУ осуществляется командой

SAVE LP:

(LP: — условное обозначение ПУ).

Вывод результатов работы программы на ПУ требует выполнения следующих действий:

1. Открыть канал связи с ПУ при помощи оператора

OPEN "LP:"FOR OUTPUT AS FILE #*n*

где *n* — номер канала связи — любое целое число от 1 до 6.

2. Для вывода вместо операторов PRINT и PRINT USING использовать операторы

PRINT # *n* [, *список*]

и

PRINT #*n*, USING *"формат"*, *список*

где *n* — номер канала связи, объявленный в операторе OPEN.

3. После окончания вывода результатов закрыть канал связи оператором

CLOSE #*n*

Обычно в процессе отладки программы результаты ее работы целесообразно выводить на экран. Вывод на ПУ используется для печати результатов работы готовой программы в окончательном виде. При этом операторы PRINT (PRINT USING) нужно заменить на PRINT #*n* (PRINT # *n*, USING). При этом изменять нужно не все операторы PRINT, а только те, которые выводят результаты работы программы. Операторы PRINT, организующие диалог с ЭВМ, например, идентифицирующие операторы INPUT по-прежнему должны осуществлять вывод на экран дисплея и должны оставаться без изменений. Для контроля за работой программы можно одновременно осуществлять вывод на экран дисплея и на ПУ. Для этого в программу нужно добавить операторы PRINT #*n* для вывода на ПУ, сохранив также все операторы PRINT для вывода на экран без изменений.

В качестве примера рассмотрим модификацию программы 3.6 для вывода результатов работы программы на ПУ (см. программу 3.10).

**Программа 3.10**

```
10 DIM M(10,5),C(5)
20 DATA 5,4,5,3,4,3,3,3,4,4
30 DATA 5,5,3,5,3,4,5,4,5,4
40 DATA 3,4,5,3,4,5,3,3,3,3
50 DATA 4,4,4,4,4,5,4,4,3,3
60 DATA 4,3,3,3,5,4,3,3,5,5
70 FOR J=1 TO 5 \ FOR I=1 TO 10 \ READ M(I,J) \ NEXT I \ NEXT J
80 FOR J=1 TO 5
90 S=0
100 FOR I=1 TO 10 \ S=S+M(I,J) \ NEXT I
110 C(J)=S/10
120 NEXT J
130 OPEN "LP:" FOR OUTPUT AS FILE #4
140 PRINT #4,"   ТАБЛИЦА СРЕДНИХ БАЛЛОВ ПО ЭКЗАМЕНАМ"
150 FOR I=1 TO 5 \ PRINT #4,C(I), \ NEXT I
160 PRINT #4
170 CLOSE #4
180 STOP
```

Пояснения к программе. Оператор в строке 130 открывает канал с номером 4 для связи с ПУ. Операторы PRINT # 4 в строках 140, 150 и 160 осуществляют вывод на ПУ. Оператор в строке 170 освобождает канал с номером 4 от связи с ПУ.

В качестве еще одного примера модифицируем программу 3.9 для вывода результатов ее работы на ПУ (см. программу 3.11).

**Программа 3.11**

```
10 PRINT "ВВЕДИТЕ ДЛЯ ВЫЧИСЛЕНИЯ ПЛОЩАДИ :"
20 PRINT "КВАДРАТА - 1, КРУГА -2, ТРЕУГОЛЬНИКА - 3"
30 INPUT K
40 PRINT "ВВЕДИТЕ РАЗМЕР X"
50 INPUT X
51 OPEN "LP:" FOR OUTPUT AS FILE #1
60 ON K GO TO 70,80,90
70 PRINT "ПЛОЩАДЬ КВАДРАТА";X*X
71 PRINT #1,"ПЛОЩАДЬ КВАДРАТА";X*X \ GO TO 100
80 PRINT "ПЛОЩАДЬ КРУГА";PI*X*X
81 PRINT #1,"ПЛОЩАДЬ КРУГА";PI*X*X \ GO TO 100
90 PRINT "ПЛОЩАДЬ ТРЕУГОЛЬНИКА";X*X*SQR(3)/4
91 PRINT #1,"ПЛОЩАДЬ ТРЕУГОЛЬНИКА";X*X*SQR(3)/4
100 CLOSE #1
110 STOP
```

Пояснения к программе. Операторы 10, 20, 40, организующие диалог с ЭВМ при вводе данных, оставлены без изменений и будут осуществлять вывод текстов на экран дисплея. Результаты выводятся на экран дисплея (операторы в строках 70, 80, 90) и печатаются с использованием ПУ (операторы в строках 71, 81, 91). Оператор OPEN располагается в строке 51 до разветвления, выполняемого оператором ON K GO TO, чтобы открыть канал связи с ПУ до вывода на печать, осуществляемого любым из операторов печати (строки 71, 81, 91).

**13.2. Использование внешних запоминающих устройств.** В качестве устройств внешней памяти далее рассматриваются только магнитные диски *).

На диске можно хранить тексты программ (точнее, последовательность программных строк) и наборы данных.

Работа с программными файлами. При работе с программой возникает необходимость сохранить ее до следующего сеанса для продолжения работы с ней или сохранить готовую программу для последующего использования.

Ограниченный объем оперативной памяти ЭВМ часто не позволяет размещать в ней достаточно длинные программы. В таких случаях программа разбивается на отдельные части (модули), хранящиеся на диске. Модули по

*) Работа с магнитной лентой во многом осуществляется аналогично работе с диском.

очереди вызываются в оперативную память и выполняются. Подробнее вопросы выполнения программ сложной структуры рассматриваются в п. 14.

Запись программы из оперативной памяти на диск осуществляется при помощи команды

SAVE [*имя*]

где *имя* — имя программы — образуется так, как указано выше (см. п.12.3).

Имя в команде SAVE, под которым программа будет записана на диск, не обязательно должно совпадать с именем, которое программа имеет в оперативной памяти. Если имя в команде SAVE отсутствует, то программа записывается на диск с тем именем, которое она имеет в оперативной памяти.

При выполнении команды SAVE на диске формируется *программный файл* (последовательность строк программы с именем), информация о котором заносится в *каталог* (оглавление) диска. При записи в каталог к имени программного файла автоматически добавляется набор символов .BAS (спецификация программного файла).

Считывание программного файла с заданным именем с диска в оперативную память осуществляется при помощи команды

OLD *имя*

При выполнении команды OLD оперативная память очищается. Затем в оперативную память с диска считывается программа. Эта программа будет иметь имя, указанное в команде OLD. После этого программу можно выполнить командой RUN или RUNNH.

Считать программный файл с диска в оперативную память можно также при помощи команды

RUN *имя* (или RUNNH *имя*)

Выполнение этой команды осуществляется аналогично команде OLD. При этом после загрузки сразу начинается выполнение программы, т. е. выполнение команды

RUN *имя*

аналогично последовательному выполнению двух команд:

OLD *имя*

и

RUN

Стирание программного файла с заданным именем с диска осуществляется командой

UNSAVE *имя*.

Запись новой версии программы с тем же именем на диск осуществляется командой

REPLACE *имя*

При этом старая версия программы уничтожается.

Работа с файлами данных. Наборы данных целесообразно хранить на диске в тех случаях, когда

— должны обрабатываться большие объемы данных, которые целиком не помещаются в оперативной памяти; в этом случае данные предварительно записываются на диск и небольшими порциями вызываются в оперативную память для обработки;

— одни и те же данные используют различные программы; тогда каждая программа может получать данные считыванием их с диска;

— целесообразно сохранить результаты выполнения программы, например, для последующего использования другими программами.

Для идентификации и удобства дальнейшей работы наборам данных присваиваются имена. Набор данных с именем называется *файлом*. При формировании файла данных на диске имя файла записывается в каталог. При этом к имени файла автоматически добавляется набор символов .DAT (спецификация файла данных).

Для работы с файлами данных (формирования файла или считывания из файла) файл нужно объявить в программе («открыть» файл).

Открытие файла данных осуществляется при помощи оператора

$$\text{OPEN } \textit{"имя"} \text{ FOR } \left\{ \begin{matrix} \text{INPUT} \\ \text{OUTPUT} \end{matrix} \right\} \text{ AS FILE } \#n$$

где $n$ — номер канала связи (целое число от 1 до 6); *имя* — имя файла данных — образуется так же, как имя программного файла; INPUT указывается при считывании из файла; OUTPUT указывается при записи в файл (формирование файла).

Закрытие файла данных должно осуществляться после окончания работы с ним при помощи оператора

CLOSE $\#n$

где *n* — номер канала связи, использованный для обмена информацией (тот же, что и в операторе OPEN).

Стирание файла данных осуществляется командой

UNSAVE *имя* .DAT

В бейсике имеется возможность работать с файлами данных последовательного и прямого доступа.

*Файл последовательного доступа* характеризуется тем, что порядок следования данных в нем определяется последовательностью, в которой данные записываются на диск. Считывание из файла последовательного доступа возможно только в том же порядке, в котором данные хранятся в этом файле.

Пересылка данных в файл последовательного доступа осуществляется при помощи оператора

PRINT #*n*, *список*

Считывание из файла последовательного доступа осуществляется при помощи оператора

INPUT #*n*, *список*

*Список* здесь имеет тот же смысл, что и для операторов PRINT и INPUT.

Данные, передаваемые в файл при помощи одного оператора PRINT #*n*, называются *записью*. Таким образом сформированный файл состоит из последовательности записей. Считывание из файла также осуществляется записями, т. е. при пересылке данных в следующий элемент списка оператора INPUT #*n* выбирается следующая запись. Это необходимо учитывать при формировании файла. Рекомендуется, как правило, при одном выполнении оператора PRINT #*n* записывать в файл одно данное, т. е. список оператора PRINT #*n* должен содержать только один элемент.

После окончания пересылки данных в файл последовательного доступа и выполнения оператора CLOSE на диске формируется признак конца файла.

В качестве примера составим программу для формирования файла последовательного доступа с именем D1. В файл D1 должны быть переданы десять чисел, вводимых с клавиатуры (см. программу 3.12).

Пояснения к программе. Оператор в строке 10 открывает файл D1 для записи и определяет канал с номером 1 для передачи данных. Далее, в цикле по I в переменную X последовательно вводятся числа. Каждое

**Программа 3.12**

```
10 OPEN "D1" FOR OUTPUT AS FILE #1
20 FOR I=1 TO 10
30 PRINT "ВВЕДИТЕ";I;" -ОЕ ЧИСЛО"
40 INPUT X
50 PRINT #1,X
60 NEXT I
70 CLOSE #1
80 STOP
```

число сразу после ввода передается в файл D1 по каналу с номером 1. Оператор в строке 70 закрывает файл.

Составим теперь программу 3.13, которая формирует массив T из данных, содержащихся в файле D1.

**Программа 3.13**

```
10 DIM T(10)
20 OPEN "D1" FOR INPUT AS FILE #2
30 FOR I=1 TO 10
40 INPUT #2,T(I)
50 PRINT T(I);
60 NEXT I
70 PRINT
80 CLOSE #2
90 STOP
```

При выполнении этой программы для каждого значения I в соответствующий элемент массива T будет передаваться очередное число из файла D1.

Если при считывании из файла количество данных в нем заранее не известно, то для окончания считывания можно использовать оператор перехода по концу файла. Этот оператор имеет вид

$$\text{IF END } \#n \begin{Bmatrix} \text{THEN} \\ \text{GO TO} \end{Bmatrix} N_i$$

где $N_i$ — номер строки, к которой будет осуществлен переход при обнаружении признака конца файла. В качестве примера составим программу, которая формирует массив T из данных, размещенных в файле последовательного доступа D2. Количество данных не превышает 100 (см. программу 3.14).

Если файл пуст, то сразу осуществляется переход по концу файла к строке 80. При этом будет I=0.

*Файл прямого доступа* характеризуется тем, что доступ к нужной информации можно получить непосредственно, зная только ее расположение в файле. Это делает файлы прямого доступа в бейсике похожими на обычные массивы.

**Программа 3.14**

```
10 DIM T(100)
20 OPEN "D2" FOR INPUT AS FILE #1
30 I=0
40 IF END #1 GO TO 80
50 I=I+1
60 INPUT #1,T(I)
70 GO TO 40
80 PRINT "КОНЕЦ ФАЙЛА"
90 PRINT "СЧИТАНО";I;"ЭЛЕМЕНТОВ"
100 CLOSE #1
110 STOP
```

Для работы с файлами прямого доступа в программе должен быть описан фиктивный массив при помощи оператора DIM #*n* (*n* — номер канала связи). Можно использовать одно- и двумерные массивы числовых или символьных величин. Для символьных массивов должна быть указана длина элемента массива (максимальное количество составляющих его символов, но не более 255).

Например, операторы

```
10 DIM #1,A(25)
20 OPEN "D1" FOR OUTPUT AS FILE #1
30 DIM #2,B$(4)=4
40 OPEN "D2" FOR INPUT AS FILE #2
```

обеспечивают возможность работы с файлами прямого доступа с именем D1 (для записи на диск) и именем D2 (для считывания с диска в оперативную память). Для передачи информации в файл D1 используется одномерный числовой массив A, а для файла D2 — символьный массив B$() с длиной каждого элемента, равной 4.

Запись в файл прямого доступа осуществляется оператором присваивания, в левой части которого указывается элемент фиктивного массива.

Считывание из файла прямого доступа осуществляется также оператором присваивания. При этом элемент фиктивного массива указывается в правой его части.

Если после операторов 10—40, приведенных выше, следуют операторы

```
50 FOR I=1 TO 25
60 A(I)=SIN(.5*I)
70 NEXT I
80 D$=B$(3)
```

то при их выполнении в файл D1 помещаются 25 чисел и переменной D$() считыванием из файла D2 присваивается значение третьего элемента файла.

После окончания работы с файлами D1 и D2, обмен с которыми осуществлялся по каналам с номерами 1 и 2, необходимо освободить каналы («закрыть» файлы). Для приведенного примера можно использовать операторы

```
150 CLOSE #1     \ CLOSE #2
```

Замечание. Если одновременно заканчивается работа со всеми файлами и ПУ, то можно использовать один оператор CLOSE без указания номера канала, что приведет к освобождению всех каналов («закрытию» всех файлов). Такой же эффект дает использование оператора END (см. п. 6).

## 14. Специальные приемы разработки и организации программ сложной структуры

**14.1. Методы разработки программ.** При разработке программ сложной структуры продуктивным является использование принципов модульного и структурного программирования.

Принцип модульного программирования заключается в том, что исходную задачу разбивают на более простые подзадачи. Это разбиение по возможности стараются выполнить так, чтобы в качестве отдельных подзадач фигурировали задачи, для решения которых уже имеются готовые (разработанные ранее) программы. Разработку новых программ при этом необходимо выполнять так, чтобы при необходимости их можно было также ипользовать в качестве составных частей для решения других задач. Решение исходной задачи на ЭВМ, таким образом, будет состоять в последовательном выполнении отдельных программ. Для того чтобы обеспечить возможность последовательного выполнения программ с автоматической передачей данных из одной программы в другую, необходимо придерживаться определенных правил в их написании. Программа, рассчитанная на ее многократное использование в качестве составной части при решении различных задач и оформленная соответствующим образом, называется *модулем*.

Каждый модуль должен удовлетворять следующим требованиям:

1) модуль должен реализовывать алгоритм решения самостоятельной задачи;

2) модуль должен иметь один вход и один выход;

3) должны быть четко определены способы передачи в модуль исходных данных и получения результатов после его выполнения;

4) модуль должен быть независим от других модулей; взаимодействие с другими модулями осуществляется только путем передачи данных;

5) модуль должен быть обозримым, т. е. содержать определенное, небольшое по количеству число операторов (обычно 50—200 операторов).

Принцип структурного программирования заключается в поэтапной разработке программы (или модуля), обычно «сверху вниз», т. е. сначала разрабатывается укрупненная структура программы, а далее отдельные блоки программы детализируются. При этом используются лишь типовые структуры, имеющие один вход и один выход. Часто поэтапную разработку называют «методом пошаговой детализации».

Пусть теперь после программирования и отладки отдельных модулей они хранятся на ВЗУ под различными именами. Далее можно или объединить их в единую программу (см. команду APPEND), или организовать поочередное выполнение модулей в определенном порядке. Во втором случае можно организовать программу так, что каждый выполняемый модуль будет вызывать для выполнения следующий (см. оператор CHAIN), или можно создать специальную так называемую «головную» программу, которая будет по очереди вызывать модули для выполнения (см. оператор OVERLAY).

Далее рассматриваются средства бейсика, используемые для организации взаимодействия программных модулей.

**14.2. Команда APPEND.** При помощи команды APPEND можно объединить разные модули в одну программу. Общий вид команды

APPEND *имя*

где *имя* — имя модуля (набор от 1 до 6 символов, начинающийся с буквы).

При выполнении команды APPEND с ВЗУ в оперативную память загружается модуль с указанным именем, который объединяется с программой, находящейся в оперативной памяти. При этом, если в программах имеются строки с одинаковыми номерами, то после объединения в программе останется строка вызванного с ВЗУ модуля,

т. е. нужно следить за тем, чтобы отдельные модули не перекрывались (не имели строк с одинаковыми номерами).

**14.3. Оператор COMMON.** При поочередном выполнении модулей, как правило, исходными данными для следующего модуля являются результаты выполнения предыдущего. При загрузке нового модуля (при помощи оператора CHAIN; см. п. 14.4) значения всех переменных и массивов стираются, кроме тех, которые в предшествующем модуле были объявлены в операторе COMMON. Таким образом, оператор COMMON можно использовать для передачи данных из одного программного модуля в другой.

Общий вид оператора

COMMON *список*

где *список* содержит имена переменных или массивов.

Для массивов нужно указывать размеры (как в операторе DIM). Массивы, объявленные в операторе COMMON, не должны описываться оператором DIM.

**14.4. Оператор CHAIN.** Оператор CHAIN вызывает с ВЗУ в оперативную память программный модуль с указанным именем. Общий вид оператора

CHAIN *имя* [LINE N]

При выполнении оператора CHAIN все открытые предыдущим модулем файлы закрываются, все строки старой программы уничтожаются, значения переменных и массивов, не описанных в предыдущем модуле оператором COMMON, стираются. Если необязательный параметр LINE N отсутствует, то управление передается строке вызванного модуля с младшим номером. В противном случае управление передается строке с номером N.

З а м е ч а н и е. Кроме оператора COMMON, для передачи данных можно использовать файлы прямого и последовательного доступа, сформированные на ВЗУ. Имя файла может быть передано при помощи оператора COMMON или при помощи диалога с пользователем непосредственно с клавиатуры.

**14.5. Оператор OVERLAY.** Оператор OVERLAY вызывает с ВЗУ в оперативную память программный модуль с указанным именем и объединяет его с программой, находящейся в оперативной памяти. Общий вид оператора

OVERLAY *имя* [LINE N]

При выполнении оператора OVERLAY строки загружаемой с ВЗУ программы замещают строки с теми же номерами

программы, находящейся в ОЗУ. Остальные строки старой программы сохраняются без изменения. Все переменные и массивы сохраняют свои значения, открытые файлы остаются открытыми. В загружаемых строках не должно быть операторов DIM и DEF FN.

Оператор OVERLAY можно использовать для изменения алгоритма, по которому обрабатываются данные; для задания переменным исходных значений с помощью операторов присваивания, через операторы DATA или через группу операторов считывания данных из файла на ВЗУ; для организации вызова следующего модуля и т. п.

Пример. Составить программу, которая в зависимости от значения некоторого признака либо определяет математическое ожидание и дисперсию для выборки из 10 чисел, либо обрабатывает тест, либо заканчивает работу. В качестве признака используется переменная С, принимающая одно из трех значений:

$$C = \begin{cases} 0 — \text{конец работы}, \\ 1 — \text{обработка данных пользователя}, \\ 2 — \text{обработка теста}. \end{cases}$$

В качестве теста используются следующие данные: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

Решение. При составлении программы решения этой задачи продемонстрируем возможности использования различных средств для организации взаимодействия модулей.

Выделим два модуля. В первом (MODUL1) будем вводить признак С и данные пользователя (если С=1). При С=0 работа программы заканчивается.

Обработка данных пользователя (или теста) осуществляется в программе MODUL2. Вызов этого модуля осуществляется в MODUL1 (если $C \neq 0$) при помощи оператора CHAIN. После выполнения обработки в программе MODUL2 снова вызывается программа MODUL1 (оператором CHAIN).

Далее приводятся тексты программ MODUL1 и MODUL2.

*Список переменных, используемых в программе* MODUL1. Исходные данные: С — признак, определяющий условия работы программы, N$ — имя программного файла для хранения данных пользователя, A — массив размером 10 (данные пользователя).

Результат: программный файл N○, содержащий данные пользователя.

Вспомогательные переменные: N1○ — для формирования имени программного файла, B○ — строка программного файла, I — управляющая переменная цикла.

**Программа MODUL1**

```
10 COMMON C,N$
20 DIM A(10)
30 PRINT "ВВЕДИТЕ УСЛОВИЯ РАБОТЫ"
40 PRINT "0 - КОНЕЦ РАБОТЫ ; 1 - ОБРАБОТКА ЭКСПЕРИМЕНТА"
50 PRINT "2 - ТЕСТ"
60 INPUT C
70 IF C=0 GO TO 260
80 IF C=2 GO TO 250
90 IF C()1 GO TO 30
100 FOR I=1 TO 10
110 PRINT "ВВЕДИТЕ";I;" ДАННОЕ"
120 INPUT A(I)
130 NEXT I
140 PRINT "ВВЕДИТЕ ИМЯ ФАЙЛА"
150 INPUT N$
160 N1$=N$+'.BAS'
170 OPEN N1$ FOR OUTPUT AS FILE #1
180 B$='30 DATA'
190 FOR I=1 TO 9
200 B$=B$+STR$(A(I))+','
210 NEXT I
220 B$=B$+STR$(A(10))
230 PRINT #1,B$
240 CLOSE #1
250 CHAIN "MODUL2"
260 PRINT "КОНЕЦ РАБОТЫ"
270 STOP
```

*Список используемых переменных* (программа MODUL2). Исходные данные: C — признак условия обработки данных, N○ — имя программного файла, сформированного в MODUL1.

Результат: M — математическое ожидание, D — дисперсия.

Вспомогательные переменные: V — массив размером 10 (данные для обработки), I — управляющая переменная цикла.

Пояснения к программам. Рассмотрим вначале программу MODUL2, осуществляющую обработку.

Если C=2, то обрабатывается тест, заданный в операторе DATA (строка 30). Если C≠2 (т. е. C=1), то выполняется оператор OVERLAY (строка 70), который вызывает с ВЗУ программный файл с именем N○. Этот файл содержит

**Программа MODUL2**

```
10 COMMON C,N$
20 DIM V(10)
30 DATA 1,2,3,4,5,6,7,8,9,10
40 PRINT "ДЛЯ ДАННЫХ"
50 M=0 \ D=0
60 IF C=2 GO TO 80
70 OVERLAY N$
80 FOR I=1 TO 10
90 READ V(I) \ PRINT V(I);
100 M=M+V(I)
110 D=D+V(I)*V(I)
120 NEXT I
130 PRINT  \ PRINT
140 M=M/10
150 D=D/10-M*M
160 PRINT "МАТЕМАТИЧЕСКОЕ ОЖИДАНИЕ M=";M
170 PRINT "ДИСПЕРСИЯ                D=";D
180 CHAIN "MODUL1"
```

строку 30 с оператором DATA, в котором заданы данные пользователя. Эта строка заменяет в оперативной памяти строку 30 с данными теста (на ВЗУ программа MODUL2 сохраняется в исходном виде!), и управление передается строке, следующей за оператором OVERLAY, на начало цикла обработки.

Далее рассмотрим программу MODUL1. В этой программе осуществляется формирование на ВЗУ файла N○ (строки 140—240). Имя формируемого файла вводится в переменную N○. В строке 160 к имени файла добавляется спецификация .BAS *), используемая для программных файлов.

Далее, в строках 180—220 формируется символьная переменная B○ поочередным включением символов (или наборов символов), из которых должна состоять программная строка. Оператор 230 записывает эту строку в качестве программного файла с именем N○.BAS на ВЗУ.

Передача значений C и N○, задаваемых в MODUL1, в MODUL2, осуществляется с помощью оператора COMMON.

Программы MODUL1 и MODUL2 вызываются с ВЗУ поочередно (одна, закончив работу, вызывает другую), пока в программе MODUL1 не будет введено C=0.

---

*) На мини-ЭВМ серий СМ и «Электроника» используется понятие *спецификация файла*, которая записывается сразу после имени файла. Спецификация применяется для определения типа файла, в частности, программные файлы на бейсике имеют стандартную спецификацию .BAS, а файлы с данными для программ на бейсике — стандартную спецификацию .DAT.

# Глава 4. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА БЕЙСИКЕ

Глава 4 содержит 10 постепенно усложняющихся работ различной тематики, предназначенных для самостоятельного выполнения. Каждая работа включает теоретическое введение, задания трех уровней, каждое из которых содержит 15 задач, и указания к решению наиболее трудных задач. К задачам работ 1, 2 и частично работ 3, 5, 6 заданы исходные данные, на которых необходимо проверить работу программы сначала вручную, а затем на ЭВМ. Предварительно необходимо при помощи ручных просчетов определить, какие результаты должна давать программа для каждого набора исходных данных, и использовать далее эти результаты при проверке работы программы. Для остальных задач необходимо подготовить тесты (см. п. 7 главы 2), обеспечивающие всестороннюю проверку работы программы. Тесты должны быть подготовлены заранее (до выполнения программы) с использованием, если необходимо, ручных просчетов.

Рекомендуется следующий порядок выполнения работ:

— изучить теоретическое введение к работе и рекомендованные разделы глав 1 и 2;

— внимательно прочитать задание соответствующего уровня;

— самостоятельно выполнить все пункты задания (в случае затруднения см. указания к решению задач);

— подготовить отчет по работе, включающий следующие разделы:

1. Формулировка задачи.
2. Математическая постановка задачи (если требуется).
3. Выбор переменных программы.
4. Схема алгоритма.
5. Программа (для работ 7—10 распечатка программы на ПУ).
6. Тесты.
7. Результаты тестирования.

В разделе «Результаты тестирования» необходимо привести исходные данные, указанные в задаче или подготовленные в качестве теста, и результаты работы программы на каждом наборе исходных данных.

## *Работа 1.* ПРОСТЕЙШИЕ ПРОГРАММЫ ЛИНЕЙНОЙ СТРУКТУРЫ. РЕЖИМ НЕПОСРЕДСТВЕННОГО ИСПОЛНЕНИЯ. ПРОГРАММНЫЙ РЕЖИМ

Теоретическое введение. *Режимом непосредственного исполнения* называется такой режим работы ЭВМ, когда каждый оператор выполняется сразу после ввода в ЭВМ.

Например, после ввода строки

```
P=1
```

(для этого нужно последовательно нажать на клавиши P, =, 1 и, далее, на клавишу ВК) в памяти ЭВМ выделяется ячейка под переменную P и этой переменной присваивается значение 1.

Если далее ввести строку

```
PRINT P
```

то значение P (равное 1) будет выведено на экран дисплея.

Если далее ввести строки

```
P=P+1
PRINT P
```

значение P изменится на 1 и новое значение (равное 2) будет выведено на экран дисплея.

В программном режиме строки программы нумеруются и сначала все вводятся в память ЭВМ. Выполнение введенных строк начинается по команде RUNNH. Например, введем строки

```
10 P=1
20 PRINT P
30 P=P+1
40 PRINT P
```

Строки программы после этого находятся в памяти ЭВМ, но не выполняются. Введем команду RUNNH. Для этого наберем на клавиатуре RUNNH и нажмем клавишу ВК. Программа начинает выполняться со строки с наименьшим номером. После выполнения очередного оператора переход к следующему происходит автоматически. В результате на экране дисплея появится

```
1
2
```

Строки программы могут вводиться в память в любой последовательности, но выполняются они в порядке возрастания номеров,

начиная со строки с наименьшим номером. Например, если ввести следующие строки в той последовательности, как они написаны:

```
40 PRINT P
20 PRINT P
10 P=1
30 P=P+1
```

то выполнение их все равно будет происходить в порядке возрастания номеров и на экране дисплея, как и в предыдущем примере, появится

1
2

Рассмотренная выше программа является примером программы линейной структуры. Ее операторы выполняются строго в порядке возрастания номеров строк. Каждый оператор выполняется один раз.

В качестве еще одного примера составим программу для вычисления выражения

$F = x^3 + 2x^4 + 6$ при $x = 3$

(см. программу 4.1).

**Программа 4.1**

```
10 X=3
20 P=X*X
30 F=P+2*P*P+6
40 PRINT F
50 STOP
```

В программе используется вспомогательная переменная P, которая вычисляется заранее (P=X*X) и используется далее при вычислении F (для вычисления $x^4$).

Задание I уровня. Предсказать, что появится на экране дисплея при последовательном вводе строк. Ввести строки в память ЭВМ. Сравнить результат с ожидаемым. Варианты задач приведены ниже.

*Варианты задач I уровня.*

1.
```
Q=5
Z=Q+8
PRINT Q;Z
```

2.
```
T=0
L=3*T
PRINT T;L
```

3.
```
A=9
B=36-A
PRINT A,B
```

4.
```
F=4
F=F+1
PRINT F
```

5.
```
L=9
X=L*L+4
PRINT L,X
```

6.
```
H=15
P=9
H=H+P
PRINT P;H
```

```
7. L=3
   L=L+2
   P=L*L
   PRINT P,L
```

```
8. A=1
   B=A*A
   PRINT A,B
   A=A+1
   B=A*A
   PRINT A,B
```

```
9. C=8
   H=C/4
   H=H*H
   PRINT C,H
```

```
10. P=2
    P=P*P
    PRINT P
    P=P+6
    PRINT P
```

```
11. P=3
    H=P+7
    H=H/2
    PRINT P,H
```

```
12. A=15
    B=A/3+1
    A=A+B
    PRINT A,B
```

```
13. X=7
    P=X*X
    PRINT X,P
    X=X+1
    P=X*X
    PRINT X,P
```

```
14. F=2
    P=3
    F=P+F
    PRINT P,F
```

```
15. R=16
    PRINT R
    R=R/8
    R=R+3
    PRINT R
```

Задание II уровня. Пронумеровать строки из вариантов задач I уровня, начиная с 10, с шагом. 10. Ввести их в память. Выполнить по команде RUNNH. Стереть программу из памяти командой SCR. Для этого набрать на клавиатуре SCR и нажать клавишу ВК.

Задание III уровня. Составить программу для решения задачи. Предсказать результат выполнения программы. Ввести программу в ЭВМ и выполнить ее. Сравнить результат с ожидаемым.

*Варианты задач III уровня.*

Вычислить:

1. $F=(x+1)^2+3(x+1)$ при $x=3$.
2. $F=6x^2+3(x^2+1)^2$ при $x=3$.
3. $F=2(x+3)+3(x+3)^2$ при $x=2$.
4. $F=x^2(x^2+1)$ при $x=6$.
5. $F=4x^2+2(x^2+1)^2$ при $x=3$.
6. $F=3(x+1)^2+2(x+1)+2$ при $x=2$.
7. $F=x^2+2x+3$ при $x=4$.
8. $F=x/2+(x/2)^2$ при $x=8$.
9. $F=(x+1)/5+(x+1)^2$ при $x=4$.
10. $F=x/3+(x/3)^2+1$ при $x=6$.
11. $F=((x+1)^2+2(x+1))/4$ при $x=3$.
12. $F=x^2/2+(x^2/2)^2+3$ при $x=4$.
13. $F=(x+1)/3+2(x+1)^2$ при $x=5$.

14. $F=(x+1)/3+(x+1)^2$ при $x=5$.
15. $F=(x^2+1)/2+27/x^2$ при $x=3$.

Вопросы для самопроверки.

1. Что такое переменная? Как обозначаются переменные на бейсике? Как записываются числа?
2. Что такое режим непосредственного исполнения? Что такое программный режим?
3. Какие операторы бейсика могут выполняться в режиме непосредственного исполнения?
4. Как ввести строку программы в память ЭВМ?
5. Какова структура программы на бейсике?
6. В каком порядке выполняются строки программы?
7. Какая команда запускает программу на выполнение?
8. Что такое программа линейной структуры?
9. Какой командой можно стереть программу из памяти?
10. Как заменить неправильно введенный символ?

## *Работа* 2. ПРОСТЕЙШИЕ ПРОГРАММЫ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Теоретическое введение. Перед выполнением заданий работы 2 необходимо ознакомиться с п. 3 главы 1, пп. 1, 2 главы 2.

*Циклом* называется многократно выполняемая последовательность действий (операторов). Цикл — типичная структура, характерная для программ, реализуемых на ЭВМ.

Структура цикла представлена на рис. 1.3 (цикл *До*) и на рис. 1.4 (цикл *Пока*).

Особенность цикла *До* в том, что тело цикла всегда выполняется хотя бы один раз. В цикле *Пока* тело цикла может не выполниться ни разу (если условие выхода из цикла оказывается выполненным при первой его проверке). При решении большинства задач можно использовать структуру цикла любого типа. В отдельных случаях использование цикла *Пока* является единственно возможным.

Решение задач работы 2 требует использования приведенных ниже типовых алгоритмов циклической структуры. Задачи сформулированы в общем виде. Алгоритмы приводятся в виде схем. Вывод результатов на печать на схеме, как правило, не предусматривается. Для большинства алгоритмов приводятся примеры их реализации в виде программ на бейсике.

В приведенных типовых алгоритмах (пп. 1—4) циклы организуются при помощи управляющей переменной цикла (см. п. 1 главы 2). Для таких циклов может быть предложена более детальная структура (рис. 4.1), в которой выделены блоки, организующие цикл: задание начального значения ($I_{нач}$) управляющей переменной цикла (I), изменение управляющей переменной цикла на шаг (H) и проверка достижения конечного значения ($I_{кон}$) управляющей переменной цикла. В частном случае, когда управляющая перемен-

ная цикла изменяется, начиная от 1, с шагом 1, ее называют *счетчиком*, а организуемый при помощи нее цикл — *циклом по счетчику*. В блоке «Начальные присваивания» должны задаваться значения переменных, используемых в теле цикла (или в блоке проверки условия), из тех соображений, чтобы при первом прохождении цикла получался правильный результат. Например, при вычислении суммы $n$ слагаемых (см. п. 1) после первого прохождения цикла сумма $S$

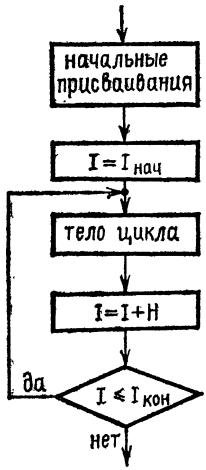



Рис. 4.1

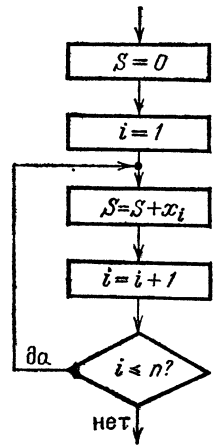



Рис. 4.2

должна быть равна первому слагаемому, т. е. после первого выполнения оператора $S=S+x_i$ (при $i=1$) должно быть $S=x_1$, а для этого в блоке «Начальные присваивания» нужно задать $S=0$ (см. рис. 4.2). Аналогично, при вычислении произведения $n$ сомножителей (п. 2) после первого выполнения оператора $P=P\cdot x_i$ (при $i=1$) должно быть $P=x_1$, а для этого в блоке «Начальные присваивания» нужно задать $P=1$ (см. рис. 4.4) и т. д.

*1. Вычисление суммы n слагаемых.* Требуется вычислить $S=\sum_{i=1}^{n} x_i$, где $x_i$ — $i$-й член суммы, зависящий от $i$. Например, при $x_i=i$ $S$ равно сумме первых $n$ натуральных чисел.

Задача сводится к организации цикла по $i$. При каждом прохождении цикла значение $S$ увеличивается на очередной член суммы. До входа в цикл нужно задать $S=0$ (рис. 4.2).

Программа 4.2 реализует приведенный алгоритм для вычисления суммы первых 15 натуральных чисел ($x_i=i$, $n=15$).

В некоторых случаях для уменьшения общего количества выполняемых арифметических операций целесообразно последующий

**Программа 4.2**

```
10 S=0
20 FOR I=1 TO 15
30 S=S+I
40 NEXT I
50 PRINT S
60 STOP
```

член суммы вычислять, используя предыдущий. При этом для обозначения члена суммы нужно использовать вспомогательную переменную. Например, при вычислении $S=\sum_{i=1}^{n} x^i$ целесообразно обозначить $c=x^i$ и для вычисления следующего члена домножить предыдущий на $x$ ($c=c\cdot x$) (рис. 4.3).

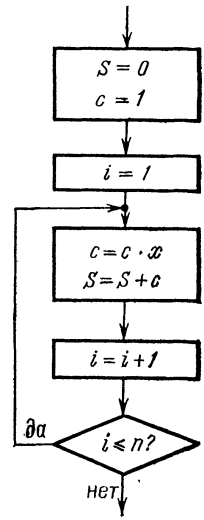


Рис. 4.3

В некоторых случаях, например, при $x_i=(-1)^i i^2$ такой прием целесообразно применять к одному из сомножителей члена суммы, а именно вычислять $c=(-1)^i$ для следующего члена суммы как $c=-c$. До входа в цикл (для первого члена) нужно положить $c=1$.

В качестве примера, иллюстрирующего рассмотренный прием, составим программу для вычисления следующей суммы $S=\sum_{i=1}^{10} (-1)^i (1+p)^{2i}/i$ при заданном значении $p=3{,}5$.

Член суммы представим как $c/i$, где $c=(-1)^i(1+p)^{2i}$. Для каждого следующего члена суммы $c$ будем вычислять по рекуррентной формуле $c=-c(1+p)^2$, где в правой части используется значение $c$ для предыдущего члена. До входа в цикл положим $c=1$ и $c$ для первого члена суммы будем вычислять по общей формуле (программа 4.3).

**Программа 4.3**

```
10 P=3.5 \ S=0 \ C=1
20 X=1+P \ X1=X*X
30 FOR I=1 TO 10
40 C=-C*X1
50 S=S+C/I
60 NEXT I
70 PRINT S
80 STOP
```

В приведенной программе вспомогательные переменные X, X1 используются для уменьшения общего объема вычислений.

*2. Вычисление произведения n сомножителей.* Требуется вычислить $P=\prod_{i=1}^{n} x_i = x_1 \cdot x_2 \cdot \ldots \cdot x_n$.

Эта задача также сводится к организации цикла по $i$. При каждом прохождении цикла произведение ($P$) умножается на очередной член. До входа в цикл нужно задать $P=1$ (рис. 4.4).

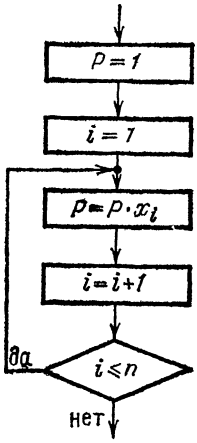


Рис. 4.4

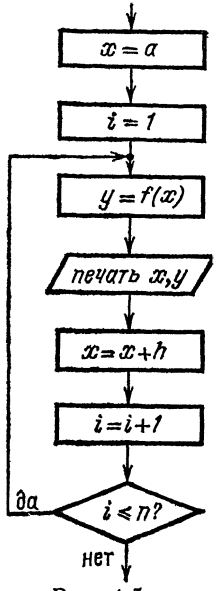


Рис. 4.5

К этому алгоритму сводится задача вычисления $P=n!$ ($x_i=i$, $i$ — натуральное). Программа 4.4 реализует этот алгоритм при $n=6$.

**Программа 4.4**

```
10 P=1
20 N=6
30 FOR I=2 TO N
40 P=P*I
50 NEXT I
60 PRINT N;"!=";P
70 STOP
```

Начальное значение управляющей переменной цикла I в операторе FOR полагается равным 2, чтобы исключить лишнюю операцию (умножение 1 на 1).

*3. Табулирование функции.* Требуется вычислить функцию $y=f(x)$ на отрезке $[a, b]$ с шагом $h$.

Для решения задачи нужно предварительно определить число точек, для которых требуется вычислить функцию ($n=(b-a)/h+1$)

и организовать цикл для вычисления значений функции в каждой точке. Печать получающихся значений нужно осуществлять в цикле каждый раз после вычисления значения функции в очередной точке (рис. 4.5).

Программа 4.5 реализует приведенный алгоритм табулирования для получения таблицы функции $y=x \sin x$ при изменении $x$ на отрезке от $-\pi$ до $\pi$ с шагом $\pi/5$ ($n=11$).

**Программа 4.5**

```
10 PRINT "X","Y"
20 PRINT
30 X=-PI \ H=PI/5 \ N=11
40 FOR I=1 TO N
50 Y=X*SIN(X)
60 PRINT X,Y
70 X=X+H
80 NEXT I
90 STOP
```

*4. Приближенное вычисление площади.* Алгоритм приближенного вычисления площади геометрической фигуры (линии, ограничивающие фигуру, заданы аналитически) является объединением алгоритмов суммирования и табулирования.

Пусть требуется вычислить приближенно площадь фигуры, ограниченной графиком функции $y=f(x)$, осью абсцисс и двумя прямыми: $x=a$ и $x=b$ ($f(x) \geqslant 0$).

Разобьем отрезок $[a, b]$ на $n$ частей и вычислим площадь фигуры приближенно как сумму $n$ прямоугольников с основаниями $h=(b-a)/n$ и высотами, определяемыми значением функции $y=f(x)$ в середине соответствующего прямоугольника.

Программа 4.6 реализует приведенный алгоритм для вычисления площади одной арки синусоиды ($a=0$, $b=\pi$, $n=10$).

**Программа 4.6**

```
10 S=0
20 H=PI/10
30 FOR I=1 TO 10
40 S=S+SIN((I-1)*H+H/2)*H
50 NEXT I
60 PRINT "ПЛОЩАДЬ ОДНОЙ АРКИ СИНУСОИДЫ";S
70 STOP
```

Структуру цикла *До* рекомендуется использовать при организации циклов с известным до выполнения цикла числом его повторений (цикл по счетчику). Если число повторений цикла до его выполнения неизвестно, то рекомендуется использовать структуру цикла *Пока*.

Пример. Мощность радиоактивного излучения пропорциональна концентрации радиоактивного вещества. Период полураспада одного из изотопов углерода составляет 8 дней. В начальный

момент времени мощность излучения составляет 2 рентгена/час. Определить, через сколько дней мощность излучения снизится до безопасной для человека величины 0,15 рентгена/час.

Изменение концентрации и, следовательно, мощности излучения описывается формулой

$$Q = Q_0 e^{-\lambda t},$$

где $Q_0$ — начальная мощность, $\lambda = \ln 2/P$, $P$ — период полураспада.

Р е ш е н и е. Для решения задачи будем изменять $t$ с заданным шагом (например, 5 дней), вычислять $Q$ и сравнивать его с допустимой величиной. Число повторений цикла заранее не известно, поэтому используем структуру цикла *Пока* (программа 4.7).

Программа 4.7

```
10 Q0=2 \ T=0 \ Q=Q0 \ L=LOG(2)/8
20 IF Q<=.15 GO TO 50
30 T=T+5 \ Q=Q0*EXP(-L*T)
40 GO TO 20
50 PRINT "КОНЦЕНТРАЦИЯ БУДЕТ БЕЗОПАСНОЙ ЧЕРЕЗ";T;"ДНЕЙ"
60 STOP
```

З а д а н и е I у р о в н я. Для приведенной программы определить, какие операторы образуют тело цикла, какая переменная является управляющей переменной цикла. Заполнить таблицу 2.1. Предсказать, что будет выведено на экран при выполнении программы. Ввести и выполнить программу. Сравнить результат с ожидаемым.

*Варианты задач I уровня.*

1.
```
10 T=1
20 PRINT T,T*T
30 T=T+2
40 IF T<=10 GOTO 20
50 STOP
```

2.
```
10 F=0
20 PRINT 2*F,F*F
30 F=F+3
40 IF F<=15 GOTO 20
50 STOP
```

3.
```
10 H=1
20 PRINT H,H+1
30 H=H+1
40 IF H<=7 GOTO 20
50 STOP
```

4.
```
10 J=4
20 PRINT J,J*7
30 J=J+2
40 IF J<=20 GOTO 20
50 STOP
```

5.
```
10 I=5
20 PRINT I,I*I
30 I=I+5
40 IF I<=20 GOTO 20
50 STOP
```

6.
```
10 A=1
20 PRINT A,A*A
30 A=A+1
40 IF A<=4 GOTO 20
50 STOP
```

**7.**
```
10 P=1
20 M=2
30 P=P*M
40 PRINT M,P
50 M=M+1
60 IF M<=6 GOTO 30
70 STOP
```

**8.**
```
10 S=0
20 I=1
30 S=S+I
40 PRINT I,S
50 I=I+1
60 IF I<=7 GOTO 30
70 STOP
```

**9.**
```
10 I=1
20 P=7*I
30 PRINT I,P
40 I=I+1
50 IF I<=7 GOTO 20
60 STOP
```

**10.**
```
10 T=12
20 C=36-T
30 PRINT T,C
40 T=T-2
50 IF T>=0 GOTO 20
60 STOP
```

**11.**
```
10 H=20
20 C=H-8
30 PRINT H,C
40 H=H-6
50 IF H>=0 GOTO 20
60 STOP
```

**12.**
```
10 A=1
20 B=A*A
30 PRINT A,B
40 A=A+2
50 IF A<=12 GOTO 20
60 STOP
```

**13.**
```
10 P=0
20 PRINT P,P*P
30 P=P+1
40 IF P<=8 GOTO 20
50 STOP
```

**14.**
```
10 P=16
20 PRINT P,P*P
30 P=P-2
40 IF P>6 GOTO 20
50 STOP
```

**15.**
```
10 P=16
20 PRINT P,P*P
30 P=P-2
40 IF P>=6 GOTO 20
50 STOP
```

**Задание II уровня.** Для предлагаемой задачи выделить повторяющиеся действия (цикл). Выбрать управляющую переменную цикла. Заполнить таблицу 2.1. Составить список переменных программы. Заполнить таблицу 2.5. Составить схему и программу для решения задачи. В программе предусмотреть печать заголовка. Выполнить программу вручную, заполняя таблицу выполнения. Проверить работу программы на ЭВМ.

*Варианты задач II уровня.*

1. Напечатать таблицу перевода температуры из градусов по шкале Цельсия ($C$) в градусы шкалы Фаренгейта ($F$) для значений от 15 °С до 30 °С с шагом 1 °С. (Перевод осуществляется по формуле $F=1{,}8C+32$.)

2. Напечатать таблицу соответствия между весом в фунтах и весом в кг для значений от 1 до 10 фунтов с шагом 1 фунт (1 фунт = = 400 г).

3. Напечатать таблицу перевода расстояний в дюймах в сантиметры (1 дюйм=2,54 см) для значений от 1 до 10 дюймов с шагом 1.

4. Вычислить сумму первых 10 натуральных чисел.

5. Вычислить произведение первых 10 натуральных чисел.

6. Составить таблицу умножения для числа 12.

7. Вычислить сумму квадратов первых 7 натуральных чисел.

8. Возвести в 7-ю степень число 3, не используя операцию возведения в степень.

9. Напечатать таблицу значений функции $y=x^3$ при изменении $x$ от 2 до 12 с шагом 2.

10. Вычислить произведение натуральных чисел, начиная от 12 до 18.

11. Вычислить $S=\sum_{i=1}^{6} 2^i$:

а) используя операцию возведения в степень;

б) не используя операцию возведения в степень.

12. Вычислить $S=\sum_{n=1}^{10} (-1)^n n^2$, не используя операцию возведения в степень.

13. Составить таблицу стоимости порций сыра весом 50, 100, 150, . . ., 1000 г (цена 1 кг 3 руб.).

14. Известно, что в 1 г живой клетчатки (например, дерева) содержится $7{,}5 \cdot 10^{10}$ ядер радиоактивного углерода. После гибели организма (дерева) радиоактивные ядра начинают распадаться и их концентрация уменьшается по закону $N=N_0 e^{-\lambda t}$, где $N_0$ — начальная концентрация, $\lambda=(\ln 2)/T$, $T$ — период полураспада (для радиоактивного углерода равен 5570 годам).

Построить таблицу зависимости концентрации радиоактивных ядер от времени для интервала времени от 0 до 6000 лет с шагом 500 лет, считая за 0 момент гибели организма (дерева).

15. Плотность воздуха убывает с высотой по закону $\rho=\rho_0 e^{-hz}$. Считая, что $\rho_0=1{,}29$ кг/м$^3$, $z=1{,}25 \cdot 10^{-4}$ 1/м, напечатать таблицу зависимости плотности от высоты для значений от 0 до 1000 м с шагом 100 м.

Указания к решению задач II уровня.

6. Таблица умножения для числа $N$ содержит результаты умножения $1 \cdot N$, $2 \cdot N$, . . ., $N \cdot N$.

12. Для обозначения сомножителя $(-1)^n$ нужно использовать вспомогательную переменную и вычислять каждое ее следующее значение умножением на —1.

14. За единицу измерения концентрации принять $10^9=1$ млрд и указать ее в шапке таблицы. Тогда начальная концентрация численно равна 75. Значение $\lambda=(\ln 2)/5570$ вычислять до входа в цикл, используя для $\lambda$ допустимое в бейсике обозначение.

Задание III уровня. Сформулировать задачу математически. Составить список переменных программы. Заполнить таблицу 2.5. Выделить циклы. Продумать их организацию. Заполнить таблицу 2.1. Составить схему и программу, если возможно, двумя способами (с использованием цикла *До* и цикла *Пока*). Указать более предпочтительный способ. Выполнить программу на ЭВМ. Проверить правильность полученных результатов.

*Варианты задач III уровня.*

1. Вычислить приближенно площадь одной арки синусоиды, разделив отрезок от 0 до $\pi$ на 10 частей и суммируя площади десяти прямоугольников с основанием $\pi/10$ и высотой, равной значению функции на правой границе каждого интервала.

2. В задаче 1 высоту каждого прямоугольника считать равной значению функции на левой границе его основания.

3. Вычислить приближенно площадь фигуры, ограниченной функцией $y=x^2$ и прямой $y=25$, разбивая отрезок изменения $x$ на 10 частей и суммируя площади прямоугольников с основанием, равным 1/10 отрезка изменения $x$, и высотой, определяемой значением функции в середине основания.

4. Вычислить приближенно площадь фигуры, ограниченной функцией $y=x^2$ и прямой $y=5+x/2$, разбивая отрезок изменения $x$ на 10 частей и суммируя площади прямоугольников с основанием, равным 1/10 отрезка, и высотой, определяемой значениями функций в середине основания.

5. Около стены наклонно стоит палка длиной $x$. Один ее конец находится на расстоянии $y$ от стены. Определить значение угла $a$ между палкой и полом для значений $x=4{,}5$ м и $y$, изменяющегося от 2 до 3 м с шагом 0,2 м.

6. Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый следующий день он увеличивал дневную норму на 10 % от нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?

7. В задаче 6 определить, через сколько дней спортсмен будет пробегать в день больше 20 км.

8. В задаче 6 определить, через сколько дней спортсмен пробежит суммарный путь 100 км.

9. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько клеток будет через 3, 6, 9, 12, . . ., 24 часа.

10. Концентрация хлорной извести в бассейне объемом $V$ м$^3$ составляет 10 г/л. Через одну трубу в бассейн вливают чистую воду с объемной скоростью $Q$ м$^3$/час, через другую трубу с такой же скоростью вода выливается. При условии идеального перемешивания концентрация хлорной извести изменяется по закону

$$C=C_0e^{-Qt/V},$$

где $t$ — время, $C_0$ — начальная концентрация.

Определить, через какое время концентрация хлорной извести достигнет безопасной для человека величины 0,1 г/л. Задачу решить при $Q=150$ м³/час, $V=10\,000$ л, $C_0=10$ г/л.

11. Для задачи 10 напечатать таблицу изменения концентрации хлорной извести для интервала времени от 0 до 5 часов с шагом 0,5 часа.

12. Определить суммарный объем в литрах 12 вложенных друг в друга шаров со стенками 5 мм. Внутренний диаметр внутреннего шара равен 10 см. Считать, что шары вкладываются друг в друга без зазоров.

13. Определить, сколько различных сигналов может быть подано $m$ флажками различных цветов. Отличие сигналов заключается в порядке расположения разноцветных флажков на мачте. Решить при $m=6$.

14. В 1985 году урожай ячменя составил 20 ц с га. В среднем каждые 2 года за счет применения передовых агротехнических приемов урожай увеличивается на 5 %. Определить, через сколько лет урожайность достигнет 25 ц с га.

15. Плотность воздуха $\rho$ с высотой $h$ убывает по закону $\rho=\rho_0 e^{-hz}$ (см. задачу 15 уровня II). Определить, на какой высоте плотность будет меньше 1 кг/м³.

Указания к решению задач III уровня.

1. В качестве управляющей переменной цикла использовать номер прямоугольника (общее их число равно 10).

3. Для определения границ изменения $x$ нужно определить точки пересечения параболы $y=x^2$ и прямой $y=25$. Высота прямоугольника в точке $x$ равна $25-x^2$.

4. См. указания к задаче 3. Высота прямоугольника в точке $x$ равна $5+x/2-x^2$.

5. Сделать геометрическое построение. Выразить $\mathrm{tg}\,a$ через отношение катетов (для определения второго катета воспользоваться теоремой Пифагора). Угол $a$ далее определять, используя стандартную функцию ATN (arctg).

6. При каждом прохождении цикла необходимо вычислять норму очередного дня умножением нормы предыдущего дня на 1,1 и прибавлять ее к суммарному пути.

7. Использовать переменную для подсчета числа прохождений цикла (результат) до первого выполнения заданного условия. См. также указание к задаче 6.

8. См. указание к задаче 7.

9. Считать, что в начальный момент имеется одна клетка. Организовать цикл с шагом изменения управляющей переменной цикла, равным 3. При каждом прохождении цикла количество клеток увеличивается в два раза.

10. Для решения задачи изменять $t$ с шагом 0,5 часа и вычис-

лять концентрацию по формуле $C=10e^{-Qt/V}$, пока не будет выполнено условие $C\leqslant 0{,}1$. (Согласовать единицы измерения всех величин!)

11. Предварительно определить количество точек таблицы и цикл организовать по номеру точки (см. также «Табулирование функции»).

12. Необходимо в цикле по номеру шара вычислять объем каждого шара и прибавлять его к суммарному объему. Диаметр каждого следующего шара отличается от диаметра вложенного в него шара на 10 мм.

13. Задача сводится к вычислению $m!$.

14. См. указание к задаче 7.

15. Изменять $h$, начиная с 0, с шагом 100 м, для каждого значения $h$ вычислять $\rho$ и сравнивать с граничным значением. Для $\rho$ и $\rho_0$ использовать допустимые в бейсике обозначения.

Вопросы для самопроверки.

1. Что такое цикл? Циклы *До* и *Пока*, различие между ними.

2. Какая структура программы на бейсике соответствует циклу *До*, циклу *Пока*?

3. Какие данные необходимы для организации цикла? Что такое управляющая переменная цикла?

4. Операторы цикла FOR и NEXT и порядок их выполнения.

5. В чем особенность записи и порядка выполнения операций в арифметическом выражении?

6. Каковы форма записи и порядок выполнения операторов присваивания, безусловного перехода, условного?

7. Как осуществляется вывод на печать? Какие элементы могут использоваться в списке вывода? Что такое вывод в зонном формате, в плотном формате?

8. Какие стандартные функции можно использовать в бейсике?

9. Типовые алгоритмы циклической структуры. Вычисление суммы $n$ слагаемых. Использование рекуррентного соотношения для вычисления следующего члена суммы или одного из его сомножителей.

10. Алгоритм вычисления произведения $n$ сомножителей. Вычисление факториала.

11. Алгоритм табулирования функции. Вывод заголовка. Вывод результатов в последовательные строки.

12. Алгоритм приближенного вычисления площади фигуры, ограниченной кривой, заданной аналитически, осью абсцисс и двумя вертикальными линиями.

13. Клавишные команды. Их роль при вводе и отладке программы.

14. Что такое зацикливание? Как прервать выполнение программы при зацикливании?

15. Основные типы ошибок в программах.

16. Как можно имитировать работу ЭВМ по выполнению программы? Что такое таблица выполнения программы и как ею пользоваться при проверке правильности программы?

17. Каков порядок разработки программы для решения задачи?

## *Работа 3.* ВВОД ДАННЫХ. РАЗВЕТВЛЕНИЯ. ЦИКЛЫ И РАЗВЕТВЛЕНИЯ

Теоретическое введение. Перед выполнением заданий работы 3 необходимо ознакомиться с п. 3 главы 1, пп. 3—6 главы 2 и п. 5.3 главы 3.

Использование ввода данных позволяет составить программу в общем виде. Такая программа может без внесения в нее каких-либо изменений использоваться для обработки различных наборов данных. Ввод данных осуществляется в процессе выполнения программы оператором INPUT (см. п. 5.3 главы 3). При остановке программы на операторе INPUT данные, подлежащие вводу, должны набираться на клавиатуре в порядке, указанном в списке ввода оператора INPUT. Ввод этих данных в память — в переменные списка ввода — осуществляется при нажатии клавиши ВК.

Независимость программы от данных часто требует в зависимости от конкретного набора данных (или промежуточных результатов) выбирать один из двух или более различных вариантов вычислительного процесса (т. е. осуществлять *разветвление* вычислительного процесса).

Разветвлением называется структура, представленная на рис. 1.5. В зависимости от того, удовлетворяется или не удовлетворяется *условие*, выполняется *действие* 1 или *действие* 2, после чего вычислительный процесс опять сводится в одно русло. Для каждого конкретного набора данных выполняется только одна ветвь вычислительного процесса, причем в каждой ветви может выполняться не одно, а последовательность действий, в частности, циклы или другие разветвления.

Программный эквивалент разветвления приведен в п. 5 главы 2.

Частным случаем разветвления является *обход* (рис. 1.6), когда в одной из ветвей не содержится никаких действий. Программный эквивалент структуры *обход* приведен в п. 5 главы 2.

Обобщением разветвления является множественный выбор (рис. 1.7). При его программировании используется оператор ON (см. п. 11 главы 3).

Примеры программ, содержащих разветвления, приведены в п. 5 главы 2.

В задачах, предлагаемых для решения в работе 3, требуется использовать, помимо числовых, еще и символьные переменные (см. п. 6 главы 3). Символьные переменные можно сравнивать (в частности, на совпадение) друг с другом и использовать в операторе условного перехода. Это позволяет придать программе диалоговые свойства.

Пример. Составить программу, суммирующую штрафное время команд при игре в хоккей. В программе предусмотреть ввод

названий команд в символьные переменные и вывод их на экран в итоговом сообщении.

При очередном удалении нужно вводить в ЭВМ название команды, игрок которой удален, и время, на которое он удален. Так как название команды состоит из нескольких символов, то для облегчения ввода их можно закодировать. Например, цифрами 1 и 2. Код команды и время удаления будем вводить различными операторами ввода. Для окончания ввода будем использовать значение того же типа, что и код команды (в нашем случае число, например 0; см. программу 4.8).

*Список используемых переменных.* Исходные данные: A□, B□ — названия команд, K — код команды, T — время удаления.

Результат: P, Q — суммарное время удалений в командах A□ и B□.

**Программа 4.8**

```
10 PRINT "ВВЕДИТЕ НАЗВАНИЯ КОМАНД"
20 INPUT Ax;Bx
30 P=0 \ Q=0
40 PRINT "ВВЕДИТЕ КОД КОМАНДЫ - 1 ИЛИ 2"
50 PRINT "ДЛЯ ОКОНЧАНИЯ ИСПОЛЬЗУЙТЕ 0"
60 INPUT K
70 IF K=0 GO TO 130
80 PRINT "ШТРАФНОЕ ВРЕМЯ"
90 INPUT T
100 IF K=1 GO TO 120
110 Q=Q+T \ GO TO 40
120 P=P+T \ GO TO 40
130 PRINT "ИГРА ОКОНЧЕНА"
140 PRINT "СУММАРНОЕ ШТРАФНОЕ ВРЕМЯ КОМАНД"
150 PRINT Ax;" -";P;"МИНУТ  ,  ";Bx;" -";Q;"МИНУТ"
160 STOP
```

З а д а н и е I у р о в н я. Формализовать постановку задачи (уяснить, что должна делать программа). Составить список используемых переменных. Составить схему и программу. Выполнить программу вручную для различных исходных данных. Ввести программу в ЭВМ и проверить ее работу.

*Варианты задач I уровня.*

1. Составить программу, которая спрашивает имя и здоровается с его обладателем.

2. На плоскости расположена окружность радиуса $R$ с центром в начале координат. Ввести заданные координаты точки и определить, лежит ли она на окружности. Результат присвоить символьной переменной. Решить задачу при $R=2$ для точек с координатами (0; 2), (—1,5; 0,7), (1; 1), (3; 0).

3. Определить, принадлежит ли заданная точка фигуре, представленной на рис. 4.6*а*. Решить задачу для точек с координатами (—0,5; 0,8), (0,5; 0,4).

4. Определить, принадлежит ли точка фигуре, представленной на рис. 4.6*б*. Решить задачу для точек с координатами (0,2; 0,3), (—2,5; 0,4), (1,5; —0,2).

5. Заданы координаты двух точек. Определить, лежат ли они на одной окружности с центром в начале координат. Результат

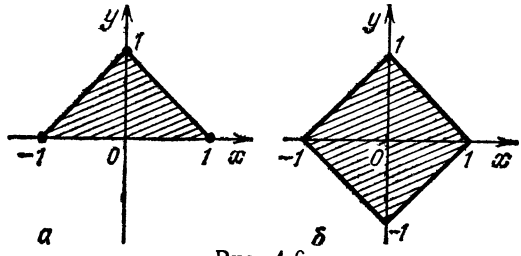



Рис. 4.6

присвоить символьной переменной. Задачу решить для точек с координатами (0; 2), (2; 0), (1; 3), (2; 2).

6. Заданы площади круга $R$ и квадрата $S$. Определить, поместится ли квадрат в круге. Задачу решить при 1) $R=70$, $S=36{,}74$; 2) $R=0{,}86$, $S=0{,}64$.

7. Для задачи 6 определить, поместится ли круг в квадрате. Задачу решить при 1) $R=3{,}2$, $S=3{,}5$; 2) $R=3{,}2$, $S=4$; 3) $R=6$, $S=9$.

8. Написать программу, которая спрашивала бы сокращенное имя, а печатала полное (например, Саша — Александр) для пяти ваших друзей. Ввод незнакомого имени должен провоцировать заявление типа: «Я с Вами не знакома». Необходимые данные задать самостоятельно.

9. Составить программу для определения подходящего возраста кандидатуры для вступления в брак, используя следующее соображение: возраст девушки равен половине возраста мужчины плюс 7, возраст мужчины определяется соответственно как удвоенный возраст девушки минус 14. Данные для проверки работы программы задать самостоятельно.

10. Составить программу, реализующую эпизод сказки: спрашивает, куда предпочитает пойти герой (налево, направо или прямо), и печатает, что его ждет в каждом случае. Ответ ЭВМ присвоить символьной переменной и напечатать. Текст вопросов и ответов ЭВМ задать самостоятельно.

11. В киоске продается газета стоимостью 3 коп. и журнал стоимостью 20 коп. Составить программу, которая спрашивает о желании покупателя (журнал или газета?), принимает деньги (сумма денег вводится с клавиатуры) и печатает причитающуюся сдачу. Исходные данные задать самостоятельно.

12. В продаже книг в книжном магазине принимает участие ЭВМ. Составить программу, которая запрашивает стоимость книг, сумму денег, внесенную покупателем, а далее определяет причитающуюся сдачу (если денег внесено больше), печатает «спасибо», если сдачи не требуется, или выдает сообщение о недостаточности внесенной суммы. Исходные данные задать самостоятельно.

13. В ЭВМ поступают результаты соревнований по плаванию для трех спортсменов. Выбрать и напечатать лучший результат. Решить задачу для следующих наборов данных: 1) 11,3; 10,6; 11; 2) 10; 10,9; 13; 3) 16; 18; 13.

14. Определить, принадлежит ли точка $D$ треугольнику $ABC$. (Треугольник задан координатами своих вершин.) Решить задачу при $A(2;2)$, $B(4;5)$, $C(7;3)$ и 1) $D(4;3)$; 2) $D(6;4,5)$; 3) $D(3;4)$.

15. Составить программу, контролирующую знание закона Ома. Обучаемый вводит формулу закона Ома в символьную переменную, которая далее сравнивается с правильным ответом, хранящимся в другой символьной переменной.

Указания к решению задач I уровня.

1. Имя ввести в символьную переменную, а затем напечатать ее значение, добавив в списке вывода оператора PRINT дополнительно одно или два слова, которыми принято выражать приветствие.

2. Считать, что точка с координатами $x$, $y$ лежит на окружности радиуса $R$, если $|\sqrt{x^2+y^2}-R|\leqslant\varepsilon$, где $\varepsilon$ — точность, с которой осуществляется проверка на равенство (можно принять $\varepsilon=10^{-5}$).

3. Уравнение прямой, ограничивающей фигуру слева: $y=1+x$ ($x<0$), справа: $y=1-x$ ($x\geqslant 0$). Следовательно, точка принадлежит фигуре, если $y\geqslant 0$ и $y+|x|\leqslant 1$.

4. Точка принадлежит фигуре, если $|y|+|x|\leqslant 1$.

5. Две точки лежат на одной окружности, если длины радиус-векторов, соединяющих эти точки с началом координат, равны. Проверку на равенство осуществлять с точностью $\varepsilon=10^{-5}$.

6. Квадрат поместится в круге, если диагональ квадрата меньше или равна диаметру окружности. Остается выразить диагональ квадрата и диаметр окружности через заданные площади этих фигур.

7. Чтобы круг поместился в квадрат, диметр круга должен быть меньше или равен стороне квадрата.

8. Ответ, содержащий сокращенное имя, нужно помещать в символьную переменную, а затем последовательно сравнивать ее значение с сокращенными именами пяти знакомых. Если введенное имя совпадает с каким-либо из использованных в программе, то нужно осуществить переход к оператору, печатающему соответствующее полное имя.

9. В начале выполнения программы на экране должен появляться вопрос «Мужчина или женщина?». Введите МУЖ или ЖЕН.

В зависимости от ответа после ввода возраста выводятся соответствующие рекомендации.

10. См. указание к задаче 9.

11. Покупателю нужно задать по крайней мере два вопроса: 1. «Что хотите купить? Журнал или газету?». После ввода ответа нужно показать стоимость соответствующего издания и задать вопрос 2. «Сколько Вы платите?». Количество денег покупателя вводится с клавиатуры (в числовую переменную). Далее нужно сравнить это количество со стоимостью покупки и напечатать соответствующее итоговое сообщение.

12. См. указание к задаче 11.

13. Задача сводится к определению минимального из трех чисел.

14. Прямая, отрезком которой является сторона треугольника, делит плоскость на две полуплоскости. Если заданная точка и противоположная этой стороне вершина треугольника находятся в разных полуплоскостях, то точка не может принадлежать треугольнику. Проверив это условие для всех трех сторон, мы решим задачу, т. е. определим, лежит точка внутри или вне треугольника.

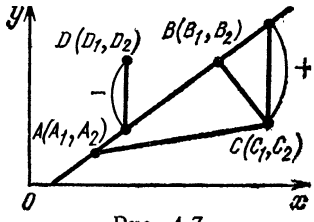


Рис. 4.7

Прямая, проходящая через вершины $A(A_1, A_2)$ и $B(B_1, B_2)$ (рис. 4.7), описывается уравнением

$$(B_1 - A_1)/(B_2 - A_2) = (B_1 - X)/(B_2 - Y),$$

определяющим зависимость $Y = f(X)$. Если заданная точка $D(D_1, D_2)$ и противоположная вершина $C(C_1, C_2)$ лежат по разные стороны от этой прямой, то произведение $(f(D_1) - D_2)(f(C_1) - C_2)$ будет отрицательным, так как его сомножители имеют противоположные знаки (см. рис. 4.7). Если для всех трех сторон произведение окажется положительным, то точка принадлежит треугольнику.

15. Перед вводом формулы (выполнением оператора INPUT) на экране должно появиться точное указание о том, в каком виде и с использованием каких обозначений нужно осуществлять ввод. Для этого можно использовать операторы PRINT, например, такого вида:

```
PRINT "ФОРМУЛУ ВВОДИТЬ БЕЗ ПРОБЕЛОВ."
PRINT "ИСПОЛЬЗОВАТЬ ОБОЗНАЧЕНИЯ:"
PRINT "I - СИЛА ТОКА; U - НАПРЯЖЕНИЕ; R - СОПРОТИВЛЕНИЕ"
```

Замечание. Прием, продемонстрированный в задаче 15, можно использовать для контроля знаний. При этом, однако, указания, выводимые на экран, должны быть сформулированы более четко, чтобы исключить неоднозначность. Например, для закона

Ома следовало бы указать, какую именно величину необходимо выразить через другие. Можно также, не давая подробных указаний, организовать в программе сравнение введенной формулы со всеми возможными правильными вариантами ее записи, которые должны храниться в памяти ЭВМ в различных символьных переменных. Последний вариант приводит к более громоздкой программе.

Задание II уровня. Задачи II уровня требуют сочетания циклов и разветвлений. Все программы нужно составить в общем виде так, чтобы число данных вводилось оператором ввода в начале программы и использовалось далее при проверке условия окончания цикла. Подготовить тесты. При отладке программы нужно задать $n=3$, 4 или 5. Для решения задачи необходимо также выполнить все пункты задания I уровня.

*Варианты задач II уровня.*

1. Определить средний рост девочек и мальчиков одного класса. В классе учится $n$ учеников.

2. Вводя в цикле по 5 оценок каждого студента, подсчитать число студентов, не имеющих оценок 2 и 3. В группе учится $n$ студентов.

3. Вводя в цикле по 4 оценки, полученные студентами в сессию, определить число неуспевающих студентов и средний балл группы по всем экзаменам.

4. Траектория снаряда, вылетающего из орудия под углом $\alpha$ с начальной скоростью $v_0$, описывается уравнениями

$$x = v_0 \cos \alpha t,$$
$$y = v_0 \sin \alpha t - gt^2/2,$$

где $g=9{,}8$ м/с$^2$ — ускорение свободного падения, $t$ — время. Вводя $n$ заданных пар $v_0$, $\alpha$, определить, сколько снарядов поразит цель высотой $P$, расположенную в вертикальной плоскости ствола орудия на расстоянии $R$ на высоте $H$ (рис. 4.8).

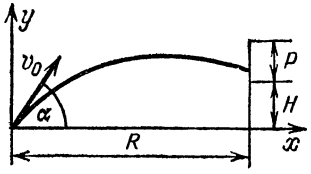


Рис. 4.8

5. Задано $n$ троек чисел $a$, $b$, $c$. Вводя их по очереди и интерпретируя как длины сторон треугольника, определить, сколько троек может быть использовано для построения треугольника (числа $a$, $b$, $c$ при вводе расположить в порядке возрастания: $a \leqslant b \leqslant c$).

6. В ЭВМ по очереди поступают результаты соревнований по плаванию, в которых участвует $n$ спортсменов. Выдавать на печать лучший результат после ввода результата очередного спортсмена.

7. В ЭВМ вводятся по очереди координаты $n$ точек. Определить, сколько из них попадет в круг радиусом $R$ с центром в точке $(a, b)$.

8. Ученикам 1-го класса назначается дополнительно стакан молока (200 мл), если их вес составляет меньше 30 кг. Определить, сколько литров молока потребуется ежедневно для одного класса, состоящего из $n$ учеников. После взвешивания вес каждого ученика вводится в ЭВМ.

9. В ЭВМ вводятся фамилии и рост учеников 7-го класса. Вывести на экран фамилии тех учеников, рост которых больше 170 см (для определения кандидатов в баскетбольную команду).

10. В ЭВМ вводятся по очереди координаты $n$ точек. Определить, сколько из них попадет в кольцо с внутренним радиусом $R_1$ и внешним $R_2$.

11. В соревнованиях по бегу принимают участие 30 спортсменов. Вводя по очереди фамилии и результаты участников в ЭВМ, определить, сколько из них выполнило норму ГТО, и напечатать их фамилии.

12. Стоимость платья зависит от материала, а также от фасона. Предполагается, что в ателье имеется 2 вида материала: шерсть по 30 руб. за метр и шелк по 16 руб. за метр. На пошив платья требуется 3 м материала. Стоимость пошива базового фасона (с минимальной отделкой) — 10 руб. За дополнительные детали отделки

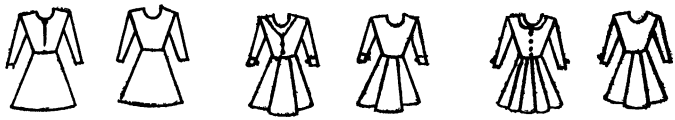


Рис. 4.9

взимается дополнительная плата. Так, 1 пуговица стоит 20 коп., 1 складка — 1 руб. 50 коп. Составить программу, которая определяет стоимость платья для $n$ заказчиков, и, используя эту программу, при $n=3$ определить стоимость платьев следующих фасонов (рис. 4.9).

13. В ЭВМ по очереди вводятся координаты $n$ точек. Определить, сколько из них принадлежит фигуре, ограниченной осью абсцисс и аркой синусоиды, построенной для аргумента от 0 до $\pi$.

14. Окружность с центром в начале координат имеет заданный радиус $R$. Вводя последовательно координаты $n$ точек, являющихся центрами других окружностей того же радиуса $R$, определить, сколько из этих окружностей пересекает заданную.

15. Группа, состоящая из $n$ студентов, сдает нормы ГТО по метанию гранаты. Вводя в цикле результат каждого студента, определить, сколько студентов выполнило норму ГТО.

Указания к решению задач II уровня.

1. Нужно просуммировать отдельно рост мальчиков и рост девочек, а также подсчитать число мальчиков и девочек, а затем

разделить суммарный рост на число девочек или мальчиков. Рост вводить в числовую переменную, признак пола («М» или «Д») — в символьную и в зависимости от значения последней прибавлять очередной рост к соответствующей сумме.

2. Ввод оценок одного студента осуществлять в пять различных переменных. Значение каждой переменной далее нужно сравнить с числом 3. При значении $\leqslant 3$ следует перейти к вводу оценок следующего студента. Если ни для одной оценки одного студента это условие не выполнено (т. е. все оценки $>3$), то количество студентов, не имеющих оценок 2 и 3, следует увеличить на 1.

3. Организовать внешний цикл по номеру студента, внутренний — по номеру оценки одного студента. Если очередная оценка — 2, то к числу неуспевающих студентов нужно прибавить 1. При этом, если у того же студента встречается вторая оценка 2, то число неуспевающих студентов не должно меняться. Для этого нужно предусмотреть специальную переменную и изменить ее значение, только если встретилась первая оценка 2. Перед началом ввода оценок очередного студента значение этой переменной нужно восстановить.

4. В первое уравнение нужно подставить $x=R$ и выразить из него $t$. Далее, для каждой пары $v_0$, $\alpha$ вычислить $t$ и, подставив во второе уравнение, определить $y$. Снаряд поразит цель, если $H<y<H+P$.

5. Для каждой тройки чисел нужно проверять условие $c \geqslant a+b$.

6. Очередной результат нужно сравнивать с минимальным из введенных ранее. Первоначально в переменную для лучшего результата можно поместить какое-либо большое положительное число, которое заведомо будет больше первого реального результата.

7. Точка с координатами $(x, y)$ попадет в круг, если $(x-a)^2+(y-b)^2<R^2$.

8. Если вес очередного ученика меньше 30 кг, то к числу учеников, которым назначается молоко, нужно прибавить 1.

9. Фамилии вводить в символьную переменную, рост — в числовую. Если рост больше 170 см, то вывести на экран значение символьной переменной.

10. После ввода координат очередной точки $(x, y)$ нужно проверить два условия: $R_1 < \sqrt{x^2+y^2}$ и $\sqrt{x^2+y^2} < R_2$.

11. В начале программы норму ГТО по данному виду ввести в числовую переменную. Для каждого спортсмена вводить его фамилию и результат. Результат сравнивать с нормой ГТО. Если норма выполнена, то на экран вывести фамилию спортсмена.

12. В цикле, тело которого должно выполняться $n$ раз, организовать диалог с заказчиком и в зависимости от его ответов на вопросы: «Какой материал?», «Какой фасон? Сколько пуговиц?

Сколько складок?» прибавлять к суммарной стоимости платья соответствующую плату. Приведенные фасоны использовать как исходные данные для проверки работы программы.

13. Для каждой точки проверять ограничения на $x$ ($0 \leqslant x \leqslant \pi$) и на $y$ ($0 \leqslant y \leqslant \sin x$). Можно использовать условный оператор вида

IF *условие* THEN *оператор*

где *оператор* является в свою очередь условным оператором (см. п. 4.3 главы 3).

Например,

```
IF 0<=X THEN
IF X<=PI THEN
IF 0<=Y THEN
IF Y<=SIN(X) THEN N=N+1 *)
```

(N — число точек, принадлежащих фигуре).

14. Для каждой новой окружности нужно проверять условие $\sqrt{x^2+y^2} < 2R$, где $x$, $y$ — вводимые координаты центра окружности.

15. См. указание к задаче 11.

Задание III уровня. Программы для задач III уровня должны работать для произвольного числа данных. Для решения задачи необходимо выполнить все пункты задания I и II уровней. Во всех программах осуществлять контроль правильности ввода.

*Варианты задач III уровня.*

1. Составить программу, суммирующую штрафное время команд при игре в хоккей. Выводить на экран суммарное штрафное время обеих команд после любого его изменения. После окончания игры выдать итоговое сообщение.

2. В ЭВМ по очереди вводятся фамилии спортсменов и их результаты в соревнованиях по прыжкам в длину. Число участников произвольно. Выдавать на печать лучший результат после выступления очередного спортсмена. После окончания соревнования напечатать итоговое сообщение о победителе.

3. Составить программу помощника кассира в универсальном магазине. ЭВМ должна запрашивать цену товара и его количество, подсчитывать суммарную стоимость купленных товаров, запрашивать сумму денег, внесенных покупателем, и определять причитающуюся ему сдачу.

4. Вы решили достаточно много раз подбросить монету, чтобы убедиться в равновероятности выпадения орла и решки. Составить программу, которая помогает вам и подсчитывает, сколько раз выпал орел, сколько — решка. Если выпадает орел, вводится «Н»;

---

*) При вводе в ЭВМ оператор следует набирать на одной строке, не нажимая клавишу ВК.

если решка — «Т». После окончания эксперимента вывести на печать итоговое сообщение.

5. Составить программу, которая ведет учет очков, набранных каждой командой при игре в баскетбол. После любого изменения счет выводить на экран. После окончания игры выдать итоговое сообщение. Предусмотреть ввод названий команд в символьные переменные и высвечивание их на экране.

6. Составить программу, подсчитывающую число удалений в каждой команде при игре в хоккей. После каждого удаления выводить на экран фамилию хоккеиста, время, на которое он удаляется с поля, и суммарное число удалений в каждой команде. После окончания игры выдать итоговое сообщение.

7. Составить программу, подсчитывающую число посещений в поликлинике врачей-специалистов (отоларинголога, окулиста и хирурга). В конце дня выдавать итоговое сообщение. В начале работы программы должна вводиться дата, которая будет фигурировать в итоговом сообщении. Использовать множественный выбор.

8. ЭВМ используется при разборе коллекции марок. Составить программу, подсчитывающую число марок по спорту, по искусству и прочих. Подсчитать общее количество марок в коллекции. Использовать множественный выбор.

9. Пассажирский самолет может поднять груз общим весом 30 т. Составить программу для определения веса почтового груза, который можно поместить в самолет после посадки пассажиров и загрузки их багажа. Во время регистрации пассажиров ЭВМ должна подсчитывать количество пассажиров (условный вес одного человека 100 кг) и суммировать вес багажа.

10. В сборе картофеля принимают участие 3 бригады. Учет ведется с помощью ЭВМ. Составить программу, которая определяет, сколько картофеля собрала каждая бригада (вес каждой новой порции прибавляется к уже накопленной сумме), выводит на экран текущий результат по трем бригадам после любых изменений и подсчитывает суммарный результат по всем бригадам в конце дня.

11. Для задачи 9 уровня I составить программу, которая может выдать необходимые сведения для всех желающих вступить в брак, количество которых заранее не известно.

12. Для задачи 12 уровня I составить программу, которая может обслуживать поток покупателей в течение дня, определяя в конце рабочего дня общую сумму, полученную от продажи книг.

13. Для задачи 11 уровня I составить программу обслуживания потока покупателей в течение дня и определения в конце дня общей суммы выручки.

14. Составить программу обслуживания соревнований по прыжкам в высоту. Для каждого выступающего спортсмена в ЭВМ вво-

дится фамилия и результат. На экране должны появиться фамилия и результат лучшего на данном этапе спортсмена.

15. При продаже граммпластинок ведется учет количества проданных пластинок с классической музыкой, эстрадной и детских Составить программу, ведущую этот учет за рабочий день. Использовать множественный выбор.

Указания к решению задач III уровня.

1. Названия команд вводить в символьные переменные. Для суммирования штрафного времени двух команд предусмотреть две числовые переменные. Признак команды и штрафное время вводить двумя операторами ввода. Для окончания использовать значение того же типа, что и признак команды. В зависимости от признака прибавлять время к соответствующей переменной. После этой операции каждый раз выводить названия команд и их суммарное штрафное время.

2. Фамилию и результат каждого спортсмена вводить двумя различными операторами INPUT. Если сначала вводится фамилия, то для окончания ввода использовать какой-либо символ (например, X).

3. Цену товара и его количество вводить различными операторами ввода. Для окончания ввода использовать специальное значение переменной, которая вводится первой.

4. В качестве признака конца целесообразно использовать значение того же типа, что и тип данных в потоке. В данной задаче можно использовать какой-либо символ, например «E».

5. См. указание к задаче 1.

6. При каждом удалении в ЭВМ вводить код команды (его можно использовать для окончания ввода; см. указание к задаче 1), фамилию игрока и время, на которое он удаляется (фамилию и время сразу после ввода выводить на экран). В зависимости от кода команды изменять на 1 суммарное число удалений одной из команд.

7. Дату ввести в символьную переменную. Ее значение напечатать в итоговом сообщении. Названия специалистов закодировать числами 1, 2, 3. Для перехода в зависимости от введенного значения к выполнению нужного оператора использовать оператор ON. Для окончания ввода можно использовать специальное значение, например 0.

8. См. указание к задаче 7.

9. Вес почтового груза определяется как разность между общим весом груза, который может поднять самолет, и весом всех пассажиров с их багажом. Для окончания ввода в качестве веса багажа можно ввести любое отрицательное число.

10. В цикле вводить номер бригады и количество собранного ею картофеля. Ввод осуществлять различными операторами ввода.

В зависимости от номера бригады прибавлять введенное количество картофеля к одной из трех переменных (суммарному количеству картофеля, собранного каждой бригадой). После чего значения всех трех переменных выводить на экран. Для окончания ввода в качестве номера бригады ввести, например, 0.

11. См. указание к задаче 9 уровня I. Для окончания цикла использовать специальный символ, например «Е».

12. Для окончания цикла в качестве стоимости книг можно вводить 0.

13. Для окончания цикла при ответе на вопрос «Журнал или газета?» вводить специальное значение, например «L».

14. Фамилию и результат лучшего на каждом этапе спортсмена хранить в символьной и числовой переменных. После ввода фамилии и результата очередного спортсмена сравнивать его результат с лучшим до него и в случае необходимости изменять фамилию и результат лучшего спортсмена. Фамилию и результат вводить двумя операторами ввода. Специальное символьное значение использовать для окончания ввода.

15. См. указания к задаче 7.

Вопросы для самопроверки.

1. Что такое разветвление и обход? Какая структура программы на бейсике соответствует разветвлению и обходу?

2. Ввод данных. Оператор INPUT. Для чего рекомендуется выводить на экран текст при выполнении оператора INPUT?

3. Что такое символьная переменная? На чем основано сравнение символьных величин?

4. Выполнение оператора INPUT. Порядок ввода данных. Ввод числовых и символьных величин.

5. Для чего необходима защита программы от неправильного ввода и как она осуществляется?

6. Как реализуется в программах на бейсике диалог с ЭВМ?

7. Особенности организации программ для обработки потока данных: а) заранее известной длины; б) заранее не известной длины.

8. Особенности организации программ для обработки потока данных произвольной длины, если при одном прохождении цикла вводится несколько данных.

9. Что такое множественный выбор? Какой оператор бейсика можно использовать для организации множественного выбора?

## *Работа 4.* ПРОСТЕЙШИЕ АЛГОРИТМЫ ОБРАБОТКИ МАССИВОВ

Теоретическое введение. Перед выполнением заданий работы 4 необходимо ознакомиться с п. 7 главы 3.

В работе рассматриваются простейшие алгоритмы работы с массивами, особенности организации циклов при обработке одномерных массивов (векторов) и двумерных массивов (матриц), организация

ввода/вывода массивов, а также общие вопросы организации программ для обработки массивов.

Рассматриваемые в работе алгоритмы являются базовыми для реализации более сложных алгоритмов обработки массивов, рассматриваемых в последующих работах. Алгоритмы оформлены в виде фрагментов программ. При самостоятельном выполнении заданий к работе может потребоваться корректировка приведенных программ, в частности изменение номеров строк. Все алгоритмы описаны в общем виде применительно к массивам произвольных размеров. Для обозначения границ изменения индексов используются переменные. При реализации приведенных фрагментов программ на ЭВМ необходимо в начале программы описать все используемые массивы оператором DIM и присвоить значения переменным, обозначающим границы изменения индексов.

Ввод/вывод массивов, а также обработка массивов связаны, как правило, с последовательным просмотром элементов массива, что реализуется организацией циклов. При этом, как правило, индексы используются в качестве управляющих переменных циклов.

*1. Ввод массивов.* Элементами ввода операторов INPUT (READ) могут быть только переменные — неиндексированные (простые) или индексированные (элементы массивов). Поэтому для ввода массивов нужно составить программу, обеспечивающую изменение индексов и последовательное заполнение элементов массивов данными.

Ввод одномерного массива

```
90 PRINT "ВВЕДИТЕ";N;" ЭЛЕМЕНТОВ МАССИВА A"
100 FOR I=1 TO N
110 INPUT A(I)
120 NEXT I
```

N должно быть определено в программе до выполнения строки 90.

Особенностью выполнения этой программы является то, что оператор INPUT выполняется N раз и при каждом его выполнении вводится одно число (очередной элемент массива), т. е. после набора на клавиатуре значения очередного элемента необходимо каждый раз нажимать клавишу ВК.

Если массив содержит небольшое известное заранее число элементов (например, 4), то в списке ввода оператора INPUT можно указать явно все элементы массива.

Пример.

```
90 PRINT "ВВЕДИТЕ 4 ЭЛЕМЕНТА МАССИВА A"
100 INPUT A(1),A(2),A(3),A(4)
```

При выполнении этой программы четыре числа можно набрать на клавиатуре через запятую и один раз нажать клавишу ВК. (Можно также нажимать ВК после набора каждого числа.)

При вводе двумерного массива необходимо организовать двойной цикл: внешний — по номеру строки, внутренний — по номеру столбца.

Ввод матрицы по строкам

```
90 PRINT "ВВЕДИТЕ МАССИВ В ПО СТРОКАМ"
100 FOR I=1 TO N
110 PRINT "ВВЕДИТЕ";I;"-УЮ СТРОКУ"
120 FOR J=1 TO M
130 INPUT B(I,J)
140 NEXT J
150 NEXT I
```

**N** — число строк, M — число столбцов матрицы.

Оператор INPUT в приведенной программе выполняется $N \times M$ раз и обеспечивает каждый раз ввод одного элемента массива, т. е. после набора на клавиатуре значения очередного элемента нужно нажимать клавишу ВК. Порядок ввода элементов определяется порядком изменения индексов в программе и для приведенной программы является следующим: B(1,1), B(1,2), . . ., B(1,M), B(2,1), B(2,2), . . ., B(2,M), . . ., B(N,1), B(N,2), . . ., B(N,M).

Ввод по строкам, реализованный в приведенной программе, является наиболее естественным. В случае необходимости можно организовать заполнение массива по столбцам. Для этого внешний цикл должен быть организован по номеру столбца (J), а внутренний — по номеру строки (I) с соответствующими границами изменения индексов.

```
90 PRINT "ВВЕДИТЕ МАССИВ В ПО СТОЛБЦАМ"
100 FOR J=1 TO M
110 PRINT "ВВЕДИТЕ";J;"-ЫЙ СТОЛБЕЦ"
120 FOR I=1 TO N
130 INPUT B(I,J)
140 NEXT I
150 NEXT J
```

В соответствии с приведенной программой элементы массива должны набираться на клавиатуре в следующем порядке: B(1,1), B(2,1), . . ., B(N,1), B(1,2), B(2,2), . . ., B(N,2), . . ., B(1,M), B(2,M), . . ., B(N,M).

Если число вводимых элементов невелико, можно не использовать циклы, а все элементы указать явно в списке ввода. Например, для ввода массива B размером $2 \times 2$ можно использовать операторы

```
90 PRINT "ВВЕДИТЕ МАССИВ В ПО СТРОКАМ"
100 INPUT B(1,1),B(1,2),B(2,1),B(2,2)
```

Ввод нескольких массивов одного размера можно осуществлять в одном цикле.

Пример.

```
90 PRINT "ВВЕДИТЕ МАССИВЫ A,C"
100 PRINT "НАБИРАТЬ ЭЛЕМЕНТЫ МАССИВОВ ПООЧЕРЕДНО"
110 FOR I=1 TO N
120 INPUT A(I),C(I)
130 NEXT I
```

Нажатие клавиши ВК можно осуществлять после набора на клавиатуре двух чисел (значений A(I), C(I)).

Однако такой способ ввода часто является причиной ошибок. Более естественно вводить сначала все элементы одного массива, затем другого. Для этого ввод каждого массива нужно осуществлять в отдельном цикле.

```
90 PRINT "ВВЕДИТЕ МАССИВ A"
100 FOR I=1 TO N
110 INPUT A(I)
120 NEXT I
130 PRINT "ВВЕДИТЕ МАССИВ C"
140 FOR I=1 TO N
150 INPUT C(I)
160 NEXT I
```

Если вводимые массивы имеют разные размеры, то второй способ является практически единственно возможным.

*2. Вывод (печать) массивов.* При выводе массивов необходимо обеспечить наглядность и удобство восприятия полученных результатов.

Вывод одномерного массива, как правило, целесообразно осуществлять в строку, сопровождая поясняющим текстом *).

Пример.

```
200 PRINT "МАССИВ A"
210 FOR I=1 TO N
220 PRINT A(I);
230 NEXT I
240 PRINT
```

В приведенной программе вывод массива A в строку обеспечивается использованием точки с запятой в операторе PRINT (строка 220). PRINT без списка (строка 240) осуществляет возврат каретки после окончания вывода массива A.

При выводе двух или нескольких одномерных массивов одного размера часто удобно вывести их как расположенные параллельно столбцы.

Пример.

```
200 PRINT "МАССИВ A";"МАССИВ B"
210 FOR I= TO N
220 PRINT A(I),B(I)
230 NEXT I
```

*) Поясняющий текст зависит от решаемой задачи.

Вывод двух или более массивов различных размеров, как правило, осуществляется в строку. Вывод нового массива начинается с новой строки.

Пример.

```
200 PRINT "МАССИВ A:";
210 FOR I=1 TO N
220 PRINT A(I);
230 NEXT I
240 PRINT
250 PRINT "МАССИВ B:";
260 FOR I=1 TO M
270 PRINT B(I);
280 NEXT I
290 PRINT
```

Приведенная программа обеспечивает печать элементов массива A в ту же строку, в которую выводится заголовок «МАССИВ A:», что обеспечивается использованием точки с запятой в строке 200. (Аналогично для массива B.)

Если массив A (или B) не умещается в одной строке, вывод будет автоматически продолжен в следующей.

Двумерные массивы необходимо выводить в привычном виде (по строкам), начиная вывод новой строки массива в новую строку экрана.

Пример.

```
200 PRINT "МАССИВ B"
210 FOR I=1 TO N
220 FOR J=1 TO M
230 PRINT B(I,J);
240 NEXT J
250 PRINT
260 NEXT I
```

После вывода очередной строки матрицы оператор PRINT (строка 250) обеспечивает возврат каретки.

Однако наглядность вывода может быть нарушена, если элементы массива B в памяти ЭВМ представляются различным числом значащих цифр. Тогда элементы одного столбца могут занимать различное число позиций и не будут располагаться строго друг под другом. Для улучшения наглядности в этом случае можно предложить два способа.

1) В операторе 230 вместо точки с запятой использовать запятую. Тогда вывод каждого элемента строки будет осуществляться в новую зону, так что элементы одинаковых столбцов будут располагаться в одинаковых зонах, т. е. строго друг под другом.

Этот способ целесообразно использовать, если $M \leqslant 5$ (число зон в строке равно 5). При $M > 5$ наглядность нарушается.

2) Вывод всех элементов матрицы по единому формату с использованием оператора PRINT USING (см. п. 5.5 главы 3, программа 3.5).

*3. Суммирование элементов массива.* Для одномерного массива $A=\{a_1, \ldots, a_n\}$ необходимо вычислить $S=\sum_{i=1}^{n} a_i$ (см. программу 4.9).

**Программа 4.9**

```
100 S=0
110 FOR I=1 TO N
120 S=S+A(I)
130 NEXT I
```

Для двумерного массива B размером N×M необходимо вычислить $S=\sum_{i=1}^{N}\sum_{j=1}^{M} b_{ij}$ (см. программу 4.10).

**Программа 4.10**

```
90 REM ВЫЧИСЛЕНИЕ СУММЫ ЭЛ-ТОВ МАТРИЦЫ
100 S=0
110 FOR I=1 TO N
120 FOR J=1 TO M
130 S=S+B(I,J)
140 NEXT J
150 NEXT I
```

4. *Суммирование диагональных элементов матрицы (вычисление следа матрицы).* Для матрицы B размером N×N необходимо вычислить $S=\sum_{i=1}^{N} b_{ii}$ (см. программу 4.11).

**Программа 4.11**

```
90 REM ВЫЧИСЛЕНИЕ СЛЕДА МАТРИЦЫ
100 S=0
110 FOR I=1 TO N
120 S=S+B(I,I)
130 NEXT I
```

*5. Суммирование двух массивов.* Для одномерных массивов A и B размером N необходимо вычислить $c_i=a_i+b_i$, $i=1, 2, \ldots, N$ (см. программу 4.12).

**Программа 4.12**

```
90 REM СУММИРОВАНИЕ ВЕКТОРОВ
100 FOR I=1 TO N
110 C(I)=A(I)+B(I)
120 NEXT I
```

Для двумерных массивов A и B размером N×M необходимо вычислить $c_{ij}=a_{ij}+b_{ij}$, $i=1, 2, \ldots, N$; $j=1, 2, \ldots, M$ (см. программу 4.13).

**Программа 4.13**

```
90 REM СУММИРОВАНИЕ МАТРИЦ
100 FOR I=1 TO N
110 FOR J=1 TO M
120 C(I,J)=A(I,J)+B(I,J)
130 NEXT J
140 NEXT I
```

*6. Суммирование элементов заданной строки матрицы.* Для матрицы B размером N×M необходимо вычислить $S=\sum_{j=1}^{M} b_{ij}$ (см. программу 4.14).

**Программа 4.14**

```
90 REM ВЫЧИСЛЕНИЕ СУММЫ I-ОЙ СТРОКИ
100 S=0
110 FOR J=1 TO M
120 S=S+B(I,J)
130 NEXT J
```

*7. Суммирование элементов строк матрицы.* Необходимо вычислить сумму элементов каждой строки матрицы B размером N×M. Результат получим в виде вектора D, т. е. вычислить $d_i=\sum_{j=1}^{M} b_{ij}$, $i=1, \ldots, N$ (см. программу 4.15).

**Программа 4.15**

```
90 REM СУММИРОВАНИЕ МАТРИЦЫ ПО СТРОКАМ
100 FOR I=1 TO N
110 S=0
120 FOR J=1 TO M
130 S=S+B(I,J)
140 NEXT J
150 D(I)=S
160 NEXT I
```

Замечание. На поиск в массиве элемента с заданными значениями индексов затрачивается время. (Адрес I-го элемента определяется прибавлением к адресу начала массива значения I.) Поэтому для повышения эффективности в приведенной выше программе при вычислении суммы каждой строки используется простая переменная S, что исключает многократное обращение к элементам массива D.

*8. Транспонирование матрицы.* Необходимо заменить строки матрицы ее столбцами, а столбцы — строками, т. е. вычислить $b_{ij}=a_{ji}$, $i=1, \ldots, N$; $j=1, \ldots, M$ (см. программу 4.16).

Транспонированную матрицу можно получить в исходном массиве A. Для квадратной матрицы размером N×N для этого необходимо поменять местами каждый элемент верхнего треугольника с соответствующим элементом нижнего (диагональные элементы

Программа 4.16

```
90 REM ТРАНСПОНИРОВАНИЕ МАТРИЦЫ
100 FOR I=1 TO N
110 FOR J=1 TO M
120 B(I,J)=A(J,I)
130 NEXT J
140 NEXT I
```

переставлять не нужно). При этом для каждой строки нужно выполнять перестановку для элементов, расположенных правее главной диагонали, с элементами соответствующего столбца, расположенными ниже главной диагонали. При перестановке используем вспомогательную переменную P, помещая в нее для временного хранения один из переставляемых элементов, чтобы не потерять его значение, т. е. необходимо выполнять операции $P=a_{ij}$, $a_{ij}=a_{ji}$, $a_{ji}=P$, $i=1, \ldots, N-1$; $j=i+1, \ldots, N$ (см. программу 4.17).

Программа 4.17

```
90 REM ТРАНСПОНИРОВАНИЕ КВАДРАТНОЙ МАТРИЦЫ
100 FOR I=1 TO N-1
110 FOR J=I+1 TO N
120 P=A(I,J) \ A(I,J)=A(J,I) \ A(J,I)=P
130 NEXT J
140 NEXT I
```

Замечание. Для прямоугольной матрицы алгоритм усложняется.

*9. Умножение матрицы на вектор.* Для вычисления произведения C матрицы A размером N×M на вектор B размером M необходимо вычислить $c_i = \sum_{j=1}^{M} a_{ij}b_j$, $i=1, \ldots, N$ (см. программу 4.18).

Программа 4.18

```
90 REM УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР
100 FOR I=1 TO N
110 S=0
120 FOR J=1 TO M
130 S=S+A(I,J)*B(J)
140 NEXT J
150 C(I)=S
160 NEXT I
```

Использование вспомогательной переменной S позволяет уменьшить время выполнения программы за счет исключения обращения в цикле по J к элементам массива C.

*10. Умножение матрицы на матрицу.* Для умножения матрицы A размером N×K на матрицу B размером K×M необходимо вычислить $c_{ij} = \sum_{l=1}^{K} a_{il} \cdot b_{lj}$, $i=1, \ldots, N$; $j=1, \ldots, M$ (см. программу 4.19).

Программа 4.19

```
90 REM УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ
100 FOR I=1 TO N
110 FOR J=1 TO M
120 S=0
130 FOR L=1 TO K
140 S=S+A(I,L)*B(L,J)
150 NEXT L
160 C(I,J)=S
170 NEXT J
180 NEXT I
```

*11. Удаление элемента из массива.* Требуется удалить K-й элемент из массива A размером N. Удалить элемент, расположенный на K-м месте в массиве, можно, сдвинув весь «хвост» массива, начиная с (K+1)-го элемента, на одну позицию влево, т. е. выполняя операции $a_i=a_{i+1}$, $i=K, K+1, \ldots, N-1$ (см. программу 4.20).

Программа 4.20

```
90 REM УДАЛЕНИЕ ЭЛЕМЕНТА
100 N=N-1
110 FOR I=K TO N
120 A(I)=A(I+1)
130 NEXT I
```

*12. Включение элемента в заданную позицию массива.* Перед включением элемента в K-ю позицию необходимо раздвинуть массив, т. е. передвинуть «хвост» массива вправо на одну позицию, выполняя операцию $a_{i+1}=a_i$, $i=N, N-1, \ldots, K$. Перемещение элементов массива нужно начинать с конца. В противном случае весь «хвост» будет заполнен элементом $a(K)$. Далее, K-му элементу присваивается заданное значение B. Размер массива увеличивается на 1 (см. программу 4.21).

Программа 4.21

```
90 REM ВКЛЮЧЕНИЕ ЭЛЕМЕНТА В МАССИВ
100 FOR I=N TO K STEP -1
110 A(I+1)=A(I)
120 NEXT I
130 A(K)=B
140 N=N+1
```

*13. Включение элемента в массив, упорядоченный по возрастанию, с сохранением упорядоченности.* Сначала необходимо найти элемент, перед которым необходимо включать заданное значение B. Для этого нужно проверять условие $B<A(I)$ для $I=1, 2, \ldots$ При выполнении условия текущее значение индекса (I) определяет позицию нового элемента. Далее, включение элемента осуществляется в соответствии с алгоритмом, описанным в п. 12 (см. программу 4.22).

Программа 4.22

```
90 REM ВКЛЮЧЕНИЕ ЭЛЕМЕНТА В УПОРЯДОЧЕННЫЙ МАССИВ
100 I=1
110 IF I>N THEN A(I+1)=B \ GO TO 180
120 IF B>=A(I) THEN I=I+1 \ GO TO 110
130 REM ВКЛЮЧЕНИЕ B В КАЧЕСТВЕ I-ГО ЭЛЕМЕНТА
140 FOR J=N TO I STEP -1
150 A(J+1)=A(J)
160 NEXT J
170 A(I)=B
180 N=N+1
```

Пояснение к программе. Если $B \geqslant A(I)$ для всех I от 1 до N, то оператор 110 при $I=N+1$ включает B в качестве $(N+1)$-го элемента.

*14. Удаление строки из матрицы.* Требуется удалить строку с заданным номером K. Решение задачи аналогично удалению элемента из одномерного массива. Все строки, начиная с $(K+1)$-й, нужно переместить вверх. Число строк уменьшается на 1 (см. программу 4.23).

Программа 4.23

```
90 REM УДАЛЕНИЕ СТРОКИ
100 N=N-1
110 FOR I=K TO N
120 FOR J=1 TO M
130 B(I,J)=B(I+1,J)
140 NEXT J
150 NEXT I
```

Примечание. Удаление столбца осуществляется аналогично.

*15. Включение строки в матрицу.* Включаемая строка задана как вектор (C). Включение строки в матрицу аналогично включению элемента в одномерный массив (см. п. 12; см. программу 4.24).

Программа 4.24

```
90 REM ВКЛЮЧЕНИЕ СТРОКИ В МАТРИЦУ
100 FOR I=N TO K STEP -1
110 FOR J=1 TO M
120 B(I+1,J)=B(I,J)
130 NEXT J
140 NEXT I
150 FOR J=1 TO M
160 A(K,J)=C(J)
170 NEXT J
180 N=N+1
```

Пояснение к программе. Операторы 100—140 перемещают строки, начиная с K-й, вниз (в обратном порядке). Перемещение одной строки связано с пересылкой всех элементов этой строки, что требует организации цикла по номеру столбца (строки 110—130). Включение строки в качестве K-й осуществляют операторы 150—170.

Примечание. Включение столбца осуществляется аналогично.

*16. Перестановка элементов в векторе.* Перестановка I-го и J-го элементов осуществляется с использованием вспомогательной переменной (P), в которую временно помещается один из элементов массива:

```
100 P=A(I) \ A(I)=A(J) \ A(J)=P
```

*17. Перестановка строк матрицы.* Рассмотрим два способа.

1) С использованием вспомогательной переменной P перестановка осуществляется во всех столбцах двух строк (см. программу 4.25).

**Программа 4.25**

```
90 REM ПЕРЕСТАНОВКА СТРОК
100 FOR K=1 TO M
110 P=A(I,K) \ A(I,K)=A(J,K) \ A(J,K)=P
120 NEXT K
```

2) При использовании вспомогательного массива (C) одна из строк (в программе — I-я) целиком пересылается в этот массив для временного хранения (см. программу 4.26).

**Программа 4.26**

```
90 REM ПЕРЕСТАНОВКА СТРОК
100 FOR K=1 TO M
110 C(K)=A(I,K)
120 NEXT K
130 FOR K=1 TO M
140 A(I,K)=A(J,K)
150 NEXT K
160 FOR K=1 TO M
170 A(J,K)=C(K)
180 NEXT K
```

Первый способ предпочтительнее, так как требует меньше памяти (не используется вспомогательный массив) и практически всегда меньше времени для выполнения программы (один цикл вместо трех).

*18. Преобразования матрицы.*

1) Умножение (деление) строки на число. Требуется умножить (разделить) все элементы строки (в программе — I-й) на одно и то же число (B), в частности, на какой-либо элемент этой же строки (см. программу 4.27).

**Программа 4.27**

```
90 REM УМНОЖЕНИЕ СТРОКИ НА ЧИСЛО
100 FOR J=1 TO N
110 A(I,J)=A(I,J)*B
120 NEXT J
```

Примечание. Умножение (деление) столбца на число осуществляется аналогично.

2) Сложение строк. Требуется к элементам K-й строки прибавить элементы L-й строки, умноженные на число (B) (см. программу 4.28).

**Программа 4.28**

```
90 REM СЛОЖЕНИЕ СТРОК
100 FOR J=1 TO N
110 A(K,J)=A(K,J)+A(L,J)*B
120 NEXT J
```

*19. Поиск минимального (максимального) элемента в массиве.* Требуется найти минимальный элемент в массиве и его значение поместить в переменную P, а индекс — в переменную K.

Для одномерного массива поиск минимального элемента осуществляется аналогично определению минимального элемента в потоке чисел, с той разницей, что в рассматриваемом случае все числа находятся в памяти (в программе — массив A), и, чтобы перейти к следующему элементу, достаточно изменить индекс на 1 (см. программу 4.29).

**Программа 4.29**

```
90 REM ПОИСК МИНИМАЛЬНОГО ЭЛЕМЕНТА ВЕКТОРА
100 P=A(1) \ K=1
110 FOR I=2 TO N
120 IF P<=A(I) GO TO 140
130 P=A(I) \ K=I
140 NEXT I
```

Если в массиве несколько элементов имеют минимальное значение, то в K будет запоминаться индекс первого из них. Если в строке 120 проверять условие P<A(I), то будет запоминаться индекс последнего элемента.

Для поиска максимального элемента в строке 120 нужно проверить условие P>=A(I).

Для двумерного массива алгоритм аналогичный, но нужно для каждой строки просматривать элементы всех столбцов (что требует организации двойного цикла) и запоминать два индекса: номер строки (K) и номер столбца (L) (см. программу 4.30).

**Программа 4.30**

```
90 REM ПОИСК МИНИМАЛЬНОГО ЭЛЕМЕНТА МАТРИЦЫ
100 P=A(1,1) \ K=1 \ L=1
110 FOR I=1 TO N
120 FOR J=1 TO M
130 IF P<=A(I,J) GO TO 150
140 P=A(I,J) \ K=I \ L=J
150 NEXT J
160 NEXT I
```

*20. Преобразование матрицы в одномерный массив.* Обработка одномерных массивов осуществляется быстрее, чем двумерных того же размера, что часто требует выполнения указанного преобразования.

Требуется переслать элементы матрицы размером N×M в одномерный массив того же размера по строкам с сохранением порядка следования элементов. Для этого нужно соответствующим образом согласовать индексы исходного массива A и формируемого X (см. программу 4.31).

**Программа 4.31**

```
90 REM ПРЕОБРАЗОВАНИЕ МАТРИЦЫ В ОДНОМЕРНЫЙ МАССИВ
100 FOR I=1 TO N
110 FOR J=1 TO M
120 X((I-1)*M+J)=A(I,J)
130 NEXT J
140 NEXT I
```

Пример 1. Задана матрица размером 4×3. Найти максимальный по модулю элемент матрицы; строку, содержащую этот элемент, переслать в вектор и напечатать.

*Список используемых переменных.* Исходные данные: X — заданная матрица размером 4×3.

Результат: T — вектор размером 3 (строка, содержащая максимальный по модулю элемент матрицы).

Вспомогательные переменные: P — модуль максимального по модулю элемента из части матрицы, Q — модуль текущего элемента матрицы, K, L — индексы максимального по модулю элемента, I, J — управляющие переменные циклов (текущие значения индексов).

Для решения задачи используем алгоритм, приведенный в п. 19 с некоторой модификацией (см. программу 4.32).

Пояснения к программе. Операторы PRINT (строки 110, 360) обеспечивают вывод пустой строки после заголовка для повышения наглядности.

Переменной Q в строке 220 присваивается значение модуля текущего элемента массива, чтобы исключить повторное выполнение этой операции в строке 240.

Пример 2. В заданной матрице размером не более 30×30 переставить местами столбец с заданным номером и последний. Полученную матрицу напечатать. Исходные данные после ввода напечатать. Предусмотреть контроль правильности ввода (см. программу 4.33).

*Список используемых переменных.* Исходные данные: N, M — размеры исходного массива (N, $M \leqslant 30$), G — заданный массив размером N×M, K — номер столбца.

Результат: G — преобразованный массив.

Программа 4.32

```
10 DIM X(4,3),T(3)
20 REM ВВОД ДАННЫХ
30 FOR I=1 TO 4
40 PRINT "ВВЕДИТЕ";I;"-УЮ СТРОКУ"
50 FOR J=1 TO 3
60 INPUT X(I,J)
70 NEXT J
80 NEXT I
90 REM ПЕЧАТЬ ИСХОДНЫХ ДАННЫХ
100 PRINT "ИСХОДНЫЙ МАССИВ"
110 PRINT
120 FOR I=1 TO 4
130 FOR J=1 TO 3
140 PRINT X(I,J);
150 NEXT J
160 PRINT
170 NEXT I
180 REM ПОИСК МАКСИМАЛЬНОГО ПО МОДУЛЮ ЭЛЕМЕНТА
190 P=ABS(X(1,1)) \ K=1 \ L=1
200 FOR I=1 TO 4
210 FOR J=1 TO 3
220 Q=ABS(X(I,J))
230 IF P)=Q GO TO 250
240 P=Q \ K=I \ L=J
250 NEXT J
260 NEXT I
270 PRINT "МАКСИМАЛЬНЫЙ ПО МОДУЛЮ ЭЛЕМЕНТ";
280 PRINT X(K,L);"НАХОДИТСЯ В";K;"-ОЙ";
290 PRINT "СТРОКЕ";L;"-ОМ СТОЛБЦЕ"
300 REM ФОРМИРОВАНИЕ МАССИВА T
310 FOR J=1 TO 3
320 T(J)=X(K,J)
330 NEXT J
340 REM ПЕЧАТЬ РЕЗУЛЬТАТА
350 PRINT "СТРОКА , СОДЕРЖАЩАЯ МАКСИМАЛЬНЫЙ ПО МОДУЛЮ ЭЛЕМЕНТ"
360 PRINT
370 FOR J=1 TO 3
380 PRINT T(J);
390 NEXT J
400 PRINT
410 STOP
```

Вспомогательные переменные: I, J — управляющие переменные циклов (индексы), P — переменная, используемая при перестановке столбцов.

Пояснения к программе. Приведенная программа составлена в общем виде и может использоваться для обработки массивов произвольных размеров в пределах ограничений, указанных в постановке задачи ($N \leqslant 30$, $M \leqslant 30$). Для этого в начале программы описан массив максимальных размеров, а далее, размеры

**Программа 4.33**

```
10 DIM G(30,30)
20 REM ВВОД ДАННЫХ
30 PRINT "ВВЕДИТЕ ЧИСЛО СТРОК И СТОЛБЦОВ"
40 INPUT N,M
50 PRINT "РАЗМЕР ИСХОДНОЙ МАТРИЦЫ";N;"*";M
60 A$="РАЗМЕР МАТРИЦЫ БОЛЬШЕ 30.ПОВТОРИТЕ ВВОД"
70 IF N>30 THEN PRINT A$ \ GO TO 30
80 IF M>30 THEN PRINT A$ \ GO TO 30
90 PRINT "ВВЕДИТЕ МАТРИЦУ ПО СТРОКАМ"
100 FOR I=1 TO N
110 FOR J=1 TO M
120 INPUT G(I,J)
130 NEXT J
140 NEXT I
150 REM ПЕЧАТЬ ИСХОДНОЙ МАТРИЦЫ
160 PRINT TAB(10);"ИСХОДНАЯ МАТРИЦА"
170 FOR I=1 TO N
180 FOR J=1 TO M
190 PRINT G(I,J);
200 NEXT J
210 PRINT
220 NEXT I
230 PRINT "ВВЕДИТЕ НОМЕР СТОЛБЦА"
240 INPUT K
250 IF K>M THEN PRINT "ТАКОГО СТОЛБЦА НЕТ" \ STOP
260 IF K=M THEN PRINT "ПЕРЕСТАНОВКА НЕ ТРЕБУЕТСЯ" \ STOP
270 REM ПЕРЕСТАНОВКА СТОЛБЦОВ
280 FOR I=1 TO N
290 P=G(I,K) \ G(I,K)=G(I,M) \ G(I,M)=P
300 NEXT I
310 REM ПЕЧАТЬ РЕЗУЛЬТАТА
320 PRINT TAB(10);"ПРЕОБРАЗОВАННАЯ МАТРИЦА"
330 FOR I=1 TO N
340 FOR J=1 TO M
350 PRINT G(I,J);
360 NEXT J
370 PRINT
380 NEXT I
390 STOP
```

конкретного массива вводятся в переменные (N, M), которые фигурируют далее везде, где используются границы изменения индексов.

В программе осуществляется контроль вводимых данных, в частности, в строках 70, 80 контролируются вводимые размеры массива, в строке 250 — номер переставляемого столбца. Программа не должна выполняться, если данные не удовлетворяют наложенным на них ограничениям.

Текст, выводимый в строках 70 или 80, задается в строке 60 как значение символьной переменной A$ (см. п. 6 главы 3).

При помощи операторов REM программа разделена на логически самостоятельные части, что облегчает ее восприятие.

Перестановка столбцов осуществляется аналогично перестановке строк (см. п. 17) с использованием одной вспомогательной переменной (строки 280—300).

Задание I уровня. Это задание требует использования одного из приведенных во введении алгоритмов для конкретного размера массива, указанного в задаче (см. пример 1).

Для решения задачи составить программу. В программе предусмотреть ввод и печать исходных данных и печать полученного результата с поясняющим текстом. Использовать циклы для ввода, вывода и обработки массивов. Привести список использованных переменных. Подготовить тесты. Проверить работу программы на ЭВМ.

*Варианты задач I уровня.*

1. Найти сумму элементов одномерного массива размером 4. Разделить каждый элемент исходного массива на полученное значение. Результат получить в том же массиве. Напечатать в одной строке.

2. Вычислить сумму и разность двух заданных одномерных массивов размером 5. Результат напечатать в виде двух параллельных столбцов.

3. Просуммировать элементы строк матрицы размером 4×3. Результат получить в одномерном массиве размером 4.

4. Задан массив X размером 5. Вычислить значения функции $y=0{,}5 \ln x$ при значениях аргумента, заданных в массиве X, и поместить их в массив Y. Напечатать результат (массивы X и Y) в виде двух столбцов.

5. Найти среднее значение элементов заданного массива размером 5. Преобразовать исходный массив, вычитая из каждого элемента среднее значение.

6. Решить уравнение $ax=b$ для пяти пар значений $a$ и $b$, заданных в виде двух массивов. Результат поместить в массив X.

7. Вычислить скалярное произведение двух векторов (X, Y) размером 4. (Скалярное произведение вычисляется по формуле $S=\sum_{i=1}^{4} x_i y_i$.)

8. Вычислить длину вектора X размером 4. (Длина вектора вычисляется по формуле $L=\sqrt{x_1^2+\ldots+x_4^2}$.)

9. Вычислить сумму двух заданных матриц размером 3×3.

10. Найти сумму всех элементов матрицы размером 4×3.

11. Просуммировать элементы столбцов заданной матрицы размером 4×3. Результат получить в одномерном массиве размером 3.

12. Определить среднее значение элементов массива. Найти далее индекс элемента массива, наиболее близкого к среднему значению.

13. Задан массив размером 10. Сформировать два массива размером 5, включая в первый элементы исходного массива с четными индексами, а во второй — с нечетными.

14. Заданы матрица размером 5×5 и число K. Разделить элементы K-й строки на диагональный элемент, расположенный в этой строке.

15. Заданы матрица A размером 4×4 и числа K и L ($K \neq L$, $1 \leqslant K, L \leqslant 4$). Из L-й строки вычесть K-ю, умноженную на $a_{LK}/a_{KK}$.

Указания к решению задач I уровня.

4. При выводе результатов оператором PRINT в каждую строку выводить пару X(I), Y(I) в зонном формате.

5. Для вычисления среднего значения найти сумму элементов и разделить на число элементов.

6. Ввести массивы A и B. В цикле по I вычислять X(I)= =B(I)/A(I) при I=1, . . ., 5.

13. В цикле выполнять P(I)=X(2*I), T(I)=X(2*I—1) при I=1, . . ., 5.

Задание II уровня. Требует применения алгоритмов, приведенных во введении, и их сочетаний. Программу необходимо составить в общем виде так, чтобы она могла быть использована для обработки массивов различных размеров в пределах заданных ограничений на размеры (для одномерных не более 100, для двумерных не более 10 по каждому измерению) (см. пример 2).

В программе предусмотреть ввод и печать исходных данных и печать полученных результатов с необходимыми поясняющими текстами. Предусмотреть контроль ввода. Привести список использованных переменных. Подготовить тесты. Проверить работу программы на ЭВМ.

*Варианты вадач II уровня.*

1. Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов. Найти след матрицы, суммируя элементы одномерного массива.

Преобразовать исходную матрицу по правилу: четные строки разделить на полученное значение, нечетные оставить без изменения. Преобразованную матрицу напечатать по строкам.

2. Задана прямоугольная матрица. Получить транспонированную матрицу и напечатать ее по строкам.

3. Заданы матрица и вектор. Получить их произведение. Напечатать в строку.

4. Заданы два одномерных массива различных размеров. Объединить их в один массив, включив второй массив между K-м и (K+1)-м элементами первого (K задано).

5. Заданы две матрицы A и B размером N×N. Сформировать из них прямоугольную матрицу X размером N×2N, включая в первые N столбцов матрицу A, в следующие — матрицу B.

6. Задан массив A размером N×M и вектор B размером M. Элементы первого столбца массива A упорядочены по убыванию. Включить массив B в качестве новой строки в массив A с сохранением упорядоченности по элементам первого столбца.

7. Матрица размещена в одномерном массиве по строкам. Удалить K-ю строку матрицы (K задано) из одномерного массива. Результат напечатать по строкам.

8. В задаче 7 удалить K-й столбец. Результат напечатать по строкам.

9. В задаче 7 поменять местами K-ю и L-ю строки. Результат вывести по строкам.

10. В задаче 7 поменять местами K-й и L-й столбцы. Результат вывести по строкам.

11. Из заданной матрицы удалить K-ю строку и L-й столбец.

12. В заданной матрице заменить K-ю строку и L-й столбец нулями, кроме элемента, расположенного на их пересечении.

13. В одномерном массиве размещены: в первых N элементах значения аргумента в порядке возрастания, в следующих — соответствующие им значения функции, и задана пара чисел — значения аргумента и функции. Поместить их в массив с сохранением упорядоченности по значениям аргумента. Напечатать полученный массив в виде двух параллельных столбцов (аргумент и функция).

14. Заданную квадратную матрицу преобразовать, используя умножение строки на число и сложение строк, таким образом, чтобы все элементы первого столбца обратились в нуль, кроме элемента, расположенного на главной диагонали (см. п. 18 введения к работе).

15. Заданную квадратную матрицу преобразовать, используя умножение столбца на число и сложение столбцов, таким образом, чтобы все элементы первой строки обратились в нуль, кроме элемента, расположенного на главной диагонали (см. п. 18 введения к работе).

Указания к решению задач II уровня.

1. Для формирования одномерного массива X из диагональных элементов матрицы A размером N×N необходимо выполнять X(I)= =A(I, I) при I=1, . . ., N.

4. «Хвост» первого массива (начиная с (K+1)-го элемента) переместить вправо на число позиций, равное размеру второго массива. Далее заполнить первый массив, начиная с (K+1)-го элемента, элементами второго массива.

5. Внутренний цикл (по номеру столбца J) целесообразно выполнять от 1 до N, присваивая для каждого J значения двум элементам строки: X(I, J)=A(I, J), X(I, N+J)=B(I, J).

6. Место включения новой строки определяется местом включения в упорядоченный первый столбец первого элемента новой строки. Новую строку нужно включить перед строкой (I), для которой выполняется условие A(I, 1)<B(1) (см. п. 13 теоретического введения).

7. K-я строка матрицы расположена в одномерном массиве в элементах с ((K—1)*M+1)-го до (K*M)-го.

8. K-й столбец матрицы расположен в элементах с индексами (I—1)*M+K, I=1, 2, . . ., N.

9. См. указание к задаче 7.

10. См. указание к задаче 8.

12. Перед заполнением нулями K-й строки и L-го столбца целесообразно элемент, лежащий на их пересечении, переслать во вспомогательную переменную, а после заполнения нулями восстановить его значение.

13. Значение аргумента включить в первую половину массива с сохранением упорядоченности. Значение функции включить на соответствующее место во вторую половину массива перед (N+I+1)-м элементом, где I — индекс элемента, в который помещено значение аргумента, N — исходное число значений аргумента и функции.

14. Из каждой строки, начиная со второй, необходимо вычитать элементы первой строки, умноженные на коэффициент P=A(K, 1)/A(1, 1), т. е. в цикле по номеру строки K (K=2, . . ., N) организовать цикл по номеру столбца J (J=1, . . ., N), в котором выполнять A(K, J)=A(K, J)—A(1, J)*P. (Предполагается, что A(1, 1)≠0.)

15. Из каждого J-го столбца, начиная со второго, необходимо вычитать элементы первого умноженного на P=A(1, J)/A(1, 1), т. е. в цикле по номеру столбца J (J=2, . . ., N) организовать цикл по номеру строки I (I=1, . . ., N), в котором выполнять A(I, J)=A(I, J)—A(I, 1)*P.

Задание III уровня. Требует использования алгоритмов, приведенных во введении к работе, а также некоторого творческого подхода. Печать массивов осуществлять по формату, используя оператор PRINT USING (см. п. 5.5 главы 3).

Необходимо также выполнить все пункты задания II уровня.

*Варианты вадач III уровня.*

1. Задан массив X размером N. Сформировать из него матрицу A, содержащую по L элементов в строке. Недостающие элементы в последней строке (если такие будут) заполнить нулями. Напечатать матрицу по строкам.

2. Вычислить значения функции $y=\cos x+x\sin x$ в $n$ точках отрезка $[a, b]$. Вычисляемые значения помещать в одномерный массив парами $x_i$, $y_i$ ($i=1, \ldots, n$). Напечатать полученный массив

в два столбца (аргумент и функция), используя для аргумента вывод по формату с фиксированной точкой, а для функции — по формату с плавающей точкой (с порядком).

3. Задана матрица A размером N×N. Сформировать два одномерных массива. В один переслать по строкам верхний треугольник матрицы, включая элементы главной диагонали, в другой — нижний треугольник. Распечатать верхний и нижний треугольники по строкам.

4. Квадратная матрица задана в виде одномерного массива по строкам. Напечатать верхний треугольник матрицы (включая элементы главной диагонали) по строкам.

5. Матрица, симметричная относительно главной диагонали, задана верхним треугольником в виде одномерного массива по строкам. Восстановить исходную квадратную матрицу и напечатать по строкам.

6. Задана квадратная матрица. Переставить строку с максимальным элементом на главной диагонали со строкой с заданным номером.

7. Задана квадратная матрица. Исключить из нее строку и столбец, на пересечении которых расположен максимальный элемент главной диагонали.

8. Заданы матрица (размером N×N) и число K ($1 \leqslant K \leqslant N$). Строку с максимальным по модулю элементом в K-м столбце переставить с K-й строкой.

9. Заданы матрица (размером N×N) и число K ($1 \leqslant K \leqslant N$). Столбец с максимальным по модулю элементом в K-й строке переставить с K-м столбцом.

10. Задана матрица размером N×N. Найти максимальный по модулю элемент матрицы. Переставить строки и столбцы матрицы таким образом, чтобы максимальный по модулю элемент был расположен на пересечении K-й строки и K-го столбца.

11. Используя преобразования, описанные в п. 18 введения к работе, привести заданную квадратную матрицу к такому виду, чтобы все элементы ниже главной диагонали были нулевыми.

12. Используя преобразования, описанные в п. 18 введения к работе, привести заданную квадратную матрицу к такому виду, чтобы все элементы выше главной диагонали были нулевыми.

13. Используя преобразования, описанные в п. 18 введения к работе, привести заданную квадратную матрицу к такому виду, чтобы элементы ниже и выше главной диагонали были нулевыми.

14. Найти произведение двух заданных матриц.

15. Задан одномерный массив. Преобразовать его таким образом, чтобы все его элементы принадлежали отрезку [—1; 1]. Вывести на печать параметры преобразования и полученный массив.

В программе предусмотреть возможность обратного преобразования.

Указания к решению задач III уровня.

1. Необходимо определить число полных строк матрицы K=INT(N/L) и число недостающих элементов в последней строке L1=N—K*L. Заполнять матрицу по строкам. (Соотношение между индексами одномерного и двумерного массивов см. в п. 20 введения к работе.)

Если L1≠0, то отдельно заполнить последнюю строку (первые L1 элементов — последними элементами одномерного массива, последние L—L1 элементов — нулями).

2. Для каждого I помещать X(I) в (2*I—1)-й, Y(I) — в (2*I)-й элементы одномерного массива.

3. Для решения первой части задачи нужно в цикле по номеру строки I организовать цикл по номеру столбца, изменяемому в пределах от I до N.

Для определения соотношения между индексами одномерного и двумерного массивов следует иметь в виду, что длина пересылаемой части строки зависит от номера строки и составляет L=N—I+1, т. е. элемент A(I, J) пересылается в ((I—1)*L+J)-й элемент одномерного массива.

Для второй части задачи решение аналогично, но цикл по J организуется в пределах от 1 до I—1 и длина пересылаемой части строки на 1 меньше.

4. См. указание к задаче 3.

5. См. указание к задаче 3.

Верхний и нижний треугольники заполнять в одном цикле, помещая один и тот же элемент одномерного массива в два расположенных симметрично относительно диагонали элемента матрицы. Диагональ матрицы заполнить в отдельном цикле.

6. Диагональ переслать в одномерный массив. Найти максимальный элемент и его индекс M. Далее переставить M-ю строку со строкой с заданным номером.

7. См. указание к задаче 6.

8. K-й столбец переслать в одномерный массив. Найти в этом массиве номер элемента, являющегося максимальным по модулю. Если максимальным по модулю является K-й элемент, перестановку не выполнять.

9. См. указание к задаче 8.

10. Найти положение максимального по модулю элемента матрицы и выполнить последовательно перестановку строк и перестановку столбцов.

11. Необходимо организовать цикл по номеру столбца и при каждом его прохождении обращать в 0 элементы, расположенные ниже главной диагонали (см. указание к задаче 14 уровня II).

Чтобы исключить возможность деления на 0 и повысить точность вычислений перед преобразованием очередного J-го столбца, нужно найти максимальный по модулю элемент в J-м столбце и перевести его на главную диагональ соответствующей перестановкой строк.

12. Необходимо организовать цикл по номеру строки и при каждом его прохождении обращать в 0 элементы, расположенные правее главной диагонали (см. указание к задаче 15 уровня II). Перед преобразованием очередной I-й строки нужно найти максимальный по модулю элемент в этой строке и соответствующей перестановкой столбцов перевести его на главную диагональ.

13. Сначала заполнить нулями нижний треугольник матрицы (см. указание к задаче 11), затем верхний (см. указание к задаче 12).

14. Перед выполнением умножения проверить соответствие умножаемых матриц друг другу по размеру. Если число столбцов первой матрицы не равно числу строк второй, умножение не выполнять.

15. Для выполнения преобразования найти максимальный $a_{max}$ и минимальный $a_{min}$ элементы в массиве. Далее, каждый элемент преобразовать по формуле

$$a_i^* = \alpha a_i + \beta,$$

где $\alpha$, $\beta$ — параметры преобразования — вычисляются из соотношений

$$\alpha a_{max} + \beta = 1,$$
$$\alpha a_{min} + \beta = -1.$$

Вопросы для самопроверки.

1. Что такое массив? Как осуществляется доступ к элементам массива?

2. Для чего нужно описывать массивы? Как осуществляется описание массивов на бейсике?

3. Как осуществить ввод вектора, матрицы: а) по строкам; б) по столбцам?

4. Как организовать ввод двух массивов одинакового размера; разных размеров?

5) Как организовать вывод вектора: а) в строку; б) в столбец; матрицы — по строкам? Какова роль оператора PRINT без списка?

6. Как вывести два вектора одинакового размера в два параллельных столбца?

7. Типовые алгоритмы обработки массивов: суммирование элементов: а) вектора; б) матрицы; вычисление следа матрицы (суммирование диагональных элементов).

8. Алгоритм суммирования: а) векторов; б) матриц.

9. Алгоритм транспонирования матрицы с получением результата: а) в другом массиве; б) в том же массиве.

10. Алгоритм умножения: а) матрицы на матрицу; б) матрицы на вектор.

11. Алгоритм удаления: а) элемента из вектора; б) строки из матрицы.

12. Алгоритм включения элемента: а) в заданную позицию вектора; б) в упорядоченный массив с сохранением упорядоченности.

13. Как поменять местами: а) элементы в векторе; б) строки в матрице?

14. Алгоритм поиска минимального (максимального) элемента массива: а) вектора; б) матрицы.

15. Алгоритм преобразования: а) матрицы в вектор; б) вектора в матрицу.

16. В чем особенности организации программ для обработки массивов произвольных размеров?

## *Работа 5.* ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ И ФУНКЦИЙ

Теоретическое введение. При решении задач на ЭВМ часто приходится в различных местах программы выполнять одну и ту же последовательность операторов, причем обычно с разными значениями переменных. В этих случаях указанную последовательность операторов, как правило, целесообразно оформить в виде подпрограммы, т. е. записать их один раз и выполнять по мере необходимости. Если при этом последовательность состоит из одного оператора присваивания, то ее можно оформить как функцию (см. п. 10 главы 3). Использование подпрограмм также оправдано, когда программируется некоторый алгоритм, представляющий самостоятельный интерес при решении других задач. В этом случае алгоритм программируется в общем виде, оформляется как подпрограмма и используется по мере надобности в различных задачах (см. пп. 9—11 главы 3).

Программа на бейсике состоит как единое целое из всех своих операторов, используемых подпрограмм и функций, т. е. на бейсике подпрограмма (функция) — это часть общей программы. Тем не менее будем условно разделять подпрограмму (функцию) и остальные операторы. Последние назовем *основной* или *вызывающей* программой.

По отношению к каждой подпрограмме все переменные программы условно разделим на четыре группы — входные, выходные, внутренние и прочие. *Входные* — это переменные, которым присваиваются значения исходных данных, необходимые для решения задачи, реализуемой данной подпрограммой. *Выходные* — это переменные, в которых получаются результаты работы подпрограммы. То есть через входные и выходные переменные основная программа осуществляет обмен информацией с подпрограммой. Таким образом, имена входных и выходных переменных обязательно присутствуют как в основной программе, так и в подпрограмме. При этом

подпрограмму желательно составлять так, чтобы она не портила входные переменные, т. е. чтобы после работы подпрограммы входные переменные имели те же значения, что и при обращении к подпрограмме, так как может возникнуть потребность использовать эти же значения при дальнейшей работе программы. *Внутренние* — это переменные подпрограммы, которые являются вспомогательными, т. е. не являются входными и выходными и используются только в подпрограмме. Внутренняя переменная должна обладать тем свойством, что если везде в данной подпрограмме заменить имя этой переменной на любое другое, не фигурирующее в основной программе, то работа в целом всей программы не изменится. *Прочие* — это все остальные переменные основной программы.

Важным моментом при использовании подпрограмм является взаимосвязь основной программы с подпрограммой. Здесь необходимо помнить следующее:

1. Перед обращением к подпрограмме всем ее входным переменным необходимо присвоить нужные значения.

2. В основной программе имя прочей переменной не должно совпадать с именем внутренней переменной подпрограммы, если эта прочая переменная не должна быть испорчена в процессе работы подпрограммы.

3. Подпрограмма не должна начать выполняться без обращения к ней.

Приведем пример неправильного использования подпрограммы. Пусть нам поставлена задача вычислить 5!, 6!, 7!, 8!. И пусть в нашем распоряжении имеется следующая (правильная!) подпрограмма вычисления N!, составленная в общем виде.

```
1000 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЯ N!
1001 REM
1002 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: N
1003 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: K  (=N!)
1004 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: I
1005 REM
1006 K=1
1007 FOR I=1 TO N
1008 K=K*I
1009 NEXT I
1010 RETURN
```

З а м е ч а н и е. K и I начинаем менять с 1 для того, чтобы подпрограмма «умела» вычислять 1!.

Теперь, имея указанную подпрограмму, естественно в цикле от 5 до 8 вычислять и печатать результаты. Однако следующая реализация этого является неверной:

```
10 REM ВЫЧИСЛЕНИЕ 5!,6!,7!,8!
20 REM
30 FOR I=5 TO 8
40 GOSUB 1000
50 PRINT K
60 NEXT I
70 STOP
```

Здесь имеются две ошибки. Первая — входной переменной N не присвоено необходимое значение. Вторая — основная программа и подпрограмма одновременно используют переменную I, взаимно ее портя, т. е. «мешая» друг другу. Правильной будет, например, следующая реализация основной программы:

```
10 REM ВЫЧИСЛЕНИЕ 5!,6!,7!,8!
20 REM
30 FOR J=5 TO 8
40 N=J \ GOSUB 1000
50 PRINT K
60 NEXT J
70 STOP
```

В случае, когда в задаче используется несколько подпрограмм и функций, целесообразно схематически изобразить их взаимосвязи и их связи с основной программой. При этом основную программу, каждую подпрограмму и функцию изобразим прямоугольником, обращение вызывающей программы к вызываемой изобразим сплошной линией, возврат в вызывающую программу — пунктирной. Над соответствующей линией можно указать входные и выходные переменные. Такое изображение дает наглядное представление о связях отдельных подпрограмм. При этом следует избегать обращения подпрограммы самой к себе *) как непосредственно, так и через другие подпрограммы (см. рис. на стр. 172).

Для того чтобы составленную подпрограмму можно было использовать в других задачах, к каждой подпрограмме составляется инструкция по ее использованию. В инструкцию, как правило, включаются следующие пункты:

1. Назначение подпрограммы.
2. Используемый метод.
3. Переменные, используемые подпрограммой (с разделением на входные, выходные и внутренние).
4. Используемые данной подпрограммой другие подпрограммы и функции.

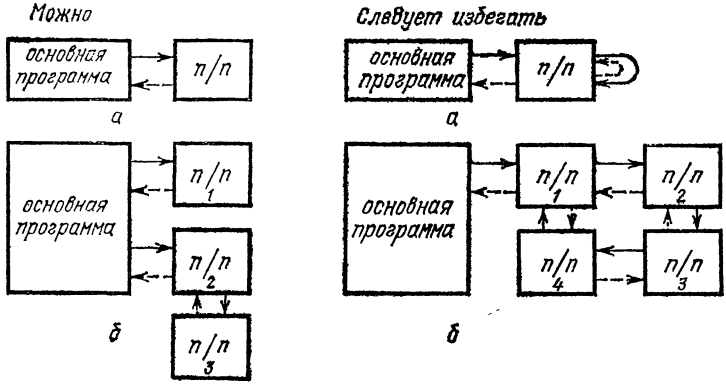
---

*) В принципе, в программировании при определенных обстоятельствах допускается организация обращения подпрограммы самой к себе (рекурсивное обращение), однако этим следует пользоваться весьма осторожно и при наличии определенного опыта программирования.

5. Требования к вызывающей программе (какие должны быть описаны массивы, какие функции и подпрограммы, какие необходимо сделать присвоения перед обращением к данной подпрограмме и т. д.).

6. Возможные изменения подпрограммы.

Для следующих примеров использования подпрограмм и функций нам потребуется знание *метода половинного деления* решения




уравнений с одним неизвестным. Этот метод заключается в следующем. Пусть дано уравнение $f(x)=0$, где функция $f$ непрерывна на отрезке $[a, b]$ и на концах отрезка имеет разные знаки (т. е. $f(a)\times f(b)<0$). Если непрерывная функция на отрезке меняет знак, то она на этом отрезке имеет по крайней мере один ноль. Пусть отрезок $[a, b]$ выделен так, что на нем имеется только один корень уравнения. Найдем этот корень методом половинного деления. Для этого разделим отрезок $[a, b]$ пополам и положим $x_1=(a+b)/2$. Если $f(x_1)=0$, то $x_1$ является корнем уравнения. Если $f(x)\neq 0$, то выбираем тот из отрезков $[a, x_1]$ или $[x_1, b]$, на концах которого $f(x)$ имеет противоположные знаки. Полученный отрезок снова делим пополам и проводим те же рассуждения. Процесс продолжаем до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданного $\varepsilon$ (точность). Как только длина отрезка станет меньше $\varepsilon$, любую точку отрезка можно с точностью $\varepsilon$ принять за корень уравнения $f(x)=0$. Схема алгоритма приведена на рис. 4.10.

Пояснения к схеме алгоритма. Алгоритм решения уравнения методом половинного деления представляет самостоятельный интерес, поэтому его имеет смысл оформить в виде подпрограммы. Это нашло отражение в схеме. Входом в указанную подпрограмму являются концы отрезка ($a$ и $b$) и требуемая точ-

ность ε. Далее, значения концов отрезка в процессе работы алгоритма будут меняться (отрезок «стягивается»), поэтому, чтобы не портить входные переменные, в схеме предусмотрены вспомогательные переменные $a_1$ и $b_1$. Значение ε в процессе работы не меняется. Чтобы подпрограмма не зависела от конкретного уравнения (т. е.

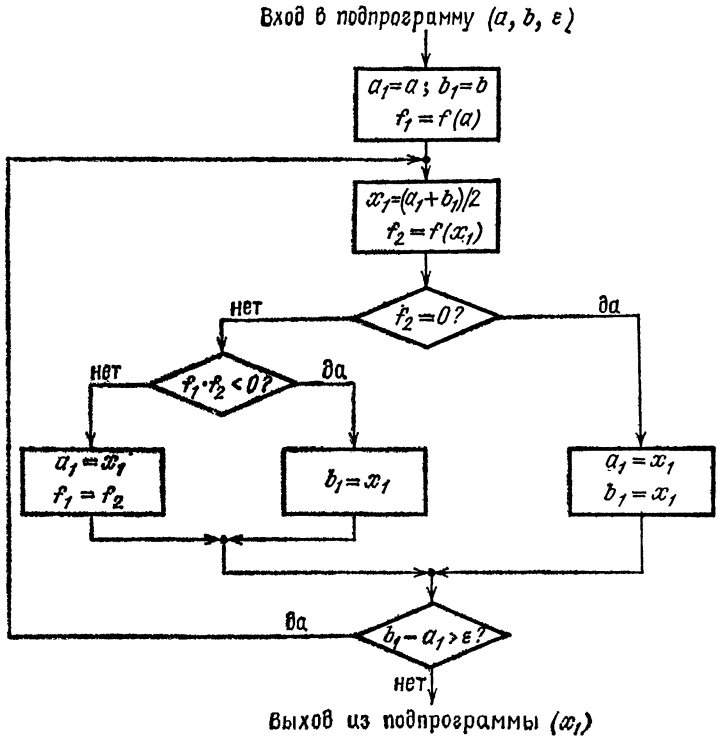


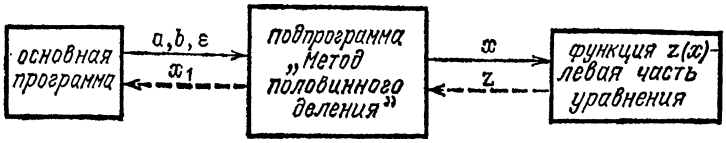

Рис. 4.10

от вида $f(x)$), оформим $f(x)$ в основной программе как функцию, а в подпрограмме будем вызывать эту функцию по мере необходимости.

Пример 1. Методом половинного деления найти корень уравнения $\cos\frac{2}{x}-2\sin\frac{1}{x}+\frac{1}{x}=0$ на отрезке [1; 2] с точностью $\varepsilon=10^{-4}$.

Составим программу, которая будет использовать подпрограмму «Метод половинного деления», которая в свою очередь будет

вызывать функцию $z(x)=\cos\frac{2}{x}-2\sin\frac{1}{x}+\frac{1}{x}$. Схема взаимосвязи основной программы, подпрограммы и функции будет следующей:




Далее следует программа решения задачи (программа 4.34).

*Список переменных, используемых в основной программе.* Исходные данные: A, B — концы отрезка [*a*, *b*], E — заданная точность ε.

Результат: X1 — вычисленное значение корня.

Вспомогательные переменные: Z — имя функции $z=f(x)$ *), X — формальный аргумент функции *z*.

Инструкция к подпрограмме «Метод половинного деления».

*Назначение*: решение уравнения вида $f(x)=0$ на отрезке [*a*, *b*] с точностью ε.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: A, B — концы отрезка, E — заданная точность.

Выходные: X1 — вычисленное значение корня.

Внутренние: A1, B1 — концы отрезка [*a*, *b*] — служат для того, чтобы подпрограмма не портила входные A и B; F1 — значение $f(x)$ в A1; F2 — значение $f(x)$ в B1.

*Используемые подпрограммы и функции*: функция $z(x)=f(x)$.

*Требования к вызывающей программе*: в основной программе (или подпрограмме, обращающейся к данной подпрограмме) необходимо:

— до обращения к данной подпрограмме описать функцию $z(x)=f(x)$;

— перед обращением к данной подпрограмме в A и B занести значения концов отрезка *a* и *b* соответственно, в E занести значение ε;

— не использовать F1, F2, A1 и B1 для обозначения переменных, которые не должны быть испорчены в процессе работы данной подпрограммы.

---

*) Хотя формально Z не является переменной, но фактически под вычисляемое значение функции выделяется ячейка памяти, обращение к которой осуществляется по имени Z. Поэтому здесь и далее имена используемых в программе функций указываются в разделе «Вспомогательные переменные».

**Программа 4.34**

```
10 REM ПРОГРАММА "РЕШЕНИЕ УРАВНЕНИЯ
20 REM МЕТОДОМ ПОЛОВИННОГО ДЕЛЕНИЯ"
30 REM
40 REM ИСХОДНЫЕ ДАННЫЕ: A,B,E
50 REM РЕЗУЛЬТАТ: X1
60 REM ВСПОМОГАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ: Z,X
70 REM
80 DEF FNZ(X)=COS(2/X)-2*SIN(1/X)+1/X
90 PRINT
100 PRINT TAB(5);"ВВЕДИТЕ A,B,E"
110 INPUT A,B,E
120 GOSUB 1000
130 PRINT
140 PRINT TAB(5);"КОРЕНЬ УРАВНЕНИЯ = ";X1
150 STOP
1000 REM ПОДПРОГРАММА "МЕТОД ПОЛОВИННОГО
1001 REM ДЕЛЕНИЯ"
1002 REM
1003 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: A,B,E
1004 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: X1
1005 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: F1,F2,A1,B1
1006 REM ИСПОЛЬЗУЕТСЯ ФУНКЦИЯ Z(X)
1007 REM
1008 A1=A \ B1=B \ F1=FNZ(A1)
1009 X1=(A1+B1)/2 \ F2=FNZ(X1)
1010 IF F2=0 GO TO 1014
1011 IF F1*F2<0 GO TO 1013
1012 A1=X1 \ F1=F2 \ GO TO 1015
1013 B1=X1 \ GO TO 1015
1014 A1=X1 \ B1=X1
1015 IF B1-A1>E GO TO 1009
1016 RETURN

RUNNH

     ВВЕДИТЕ A,B,E

? 1,2,0.0001

     КОРЕНЬ УРАВНЕНИЯ =  1.87567

STOP AT LINE 150
```

*Возможные изменения подпрограммы*: при перенумерации строк подпрограммы необходимо соответственно изменить номера строк в операторах перехода в строках 1010, 1011, 1012, 1013, 1015.

З а м е ч а н и е. Если необходимо решить какое-либо другое уравнение с использованием приведенной программы, то в основной программе надо в строке 80 записать функцию, фигурирующую в уравнении, и ввести необходимые A, B, E.

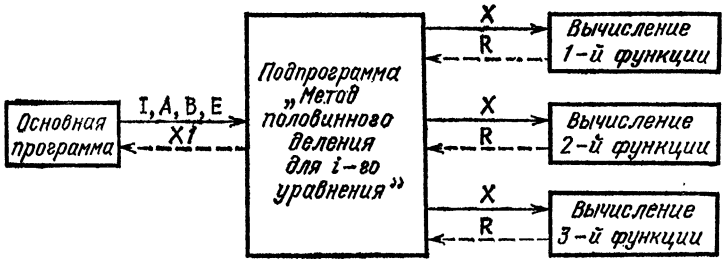
Пример 2. Решить методом половинного деления уравнения $\cos\frac{2}{x}-2\sin\frac{1}{x}+\frac{1}{x}=0$ на отрезке [1; 2] с точностью $10^{-4}$;

$\sin(\ln x)-\cos(\ln x)+2\ln x=0$ на отрезке [1; 3] с точностью $10^{-3}$;

$e^x+\sqrt{1+e^{2x}}-2=0$ на отрезке [—1; 0] с точностью $10^{-4}$.

В данной задаче мы можем воспользоваться непосредственно предыдущей программой, выполнив ее три раза, каждый раз меняя строку 80 и вводя новые A, B и E (см. замечание). Но можно несколько модифицировать подпрограмму «Метод половинного деления» так, чтобы она «умела» решать несколько уравнений поочередно. Соответственно, несколько изменится основная программа. В подпрограмму, во-первых, необходимо ввести еще один входной параметр I — номер решаемого уравнения. Во-вторых, поскольку в программе нельзя использовать различные функции с одним и тем же именем, необходимо решить проблему вызова нужной (I-й) функции. Эту проблему можно решить, например, так: вычисление необходимых функций оформить в виде подпрограмм и обращение к ним осуществлять оператором ON I GOSUB. При этом, естественно, перед обращением к очередной подпрограмме необходимо присвоить нужное значение входной переменной X (см. программу 4.35).

Схема взаимосвязи основной программы и подпрограмм для данной задачи будет следующей:




*Список переменных, используемых в основной программе.* Исходные данные: N — количество решаемых уравнений, A, B — концы очередного отрезка, E — заданная точность решения очередного уравнения.

Результат: X1 — вычисленный корень очередного уравнения.

Вспомогательные переменные: I — номер очередного уравнения.

Инструкция к подпрограмме «Метод половинного деления для *i*-го уравнения».

*Назначение*: решение $i$-го уравнения из группы уравнений вида $f(x)=0$ методом половинного деления.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: I — номер решаемого уравнения — служит параметром оператора ON GOSUB в строках 1009, 1011; A, B — концы отрезка существования корня очередного уравнения; E — точность, с которой надо решать очередное уравнение.

Выходные: X1 — вычисленное значение корня очередного уравнения.

Внутренние: A1, B1 — концы отрезка — служат для того, чтобы подпрограмма не портила входные A и B; F1 — значение $f(x)$ в A1; F2 — значение $f(x)$ в B1; X — значение, при котором вычисляется левая часть уравнения,— служит входным параметром подпрограммы вычисления левой части уравнения; R — значение левой части уравнения, вычисленное при X,— является результатом работы подпрограммы вычисления левой части уравнения.

*Используемые подпрограммы и функции*: подпрограммы вычисления значений левых частей уравнений располагаются в строках, начиная с 2006, по одной в строке.

*Требования к вызывающей программе*: в вызывающей программе необходимо:

— описать подпрограммы вычисления значений левых частей уравнений. Подпрограмм должно быть столько, сколько уравнений. Подпрограммы должны располагаться в строках 2006, 2007, . . ., по одной в строке;

— перед обращением к данной подпрограмме переменной I присвоить значение номера очередного уравнения, в A и B занести значения концов очередного отрезка, в E занести значение очередного $\varepsilon$;

— не использовать F1, F2, A1, B1, R, X для обозначения переменных, которые не должны быть испорчены в процессе работы данной подпрограммы.

*Возможные изменения подпрограммы*: если необходимо решать большее количество уравнений, то в строках 1009 и 1011 в операторе ON GOSUB надо добавить номера строк, с которых начинаются подпрограммы вычисления значений левых частей. При перенумерации строк подпрограммы необходимо соответственно изменить номера строк в операторах перехода в строках 1012, 1013, 1014, 1015, 1017.

Инструкция к подпрограмме вычисления левой части уравнения (для всех подпрограмм инструкция тождественна).

*Назначение*: вычисление левой части уравнения вида $f(x)=0$.

*Переменные, используемые подпрограммой.* Входные: X — аргумент функции $f(x)$.

Выходные: R — вычисленное значение $f(x)$.

Внутренние: нет.

*Используемые подпрограммы и функции*: нет.

*Требования к вызывающей программе*: в вызывающей программе необходимо присвоить переменной X значение аргумента, при котором надо вычислить левую часть уравнения.

*Возможные изменения подпрограммы*: нет.

**Программа 4.35**

```
10 REM ПРОГРАММА "РЕШЕНИЕ УРАВНЕНИЙ
20 REM МЕТОДОМ ПОЛОВИННОГО ДЕЛЕНИЯ"
30 REM
40 REM ИСХОДНЫЕ ДАННЫЕ: N,A,B,E
50 REM РЕЗУЛЬТАТЫ: X1
60 REM ВСПОМОГАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ: I
70 REM
80 PRINT
90 PRINT TAB(2);"ВВЕДИТЕ N"
100 INPUT N
110 FOR I=1 TO N
120 PRINT
130 PRINT TAB(2);"ВВЕДИТЕ A,B,E"
140 INPUT A,B,E
150 GOSUB 1000
160 PRINT "КОРЕНЬ ";I;"УРАВНЕНИЯ = ";X1
170 NEXT I
180 STOP
1000 REM ПОДПРОГРАММА "МЕТОД ПОЛОВИННОГО
1001 REM ДЕЛЕНИЯ ДЛЯ I-ГО УРАВНЕНИЯ"
1002 REM
1003 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: I,A,B,E
1004 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: X1
1005 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: R,F1,F2,A1,B1,X
1006 REM ИСПОЛЬЗУЮТСЯ ПОДПРОГРАММЫ: СМ. 2000
1007 REM
1008 A1=A \ B1=B \ X=A1
1009 ON I GOSUB 2006,2007,2008 \ F1=R
1010 X1=(A1+B1)/2 \ X=X1
1011 ON I GOSUB 2006,2007,2008 \ F2=R
1012 IF F2=0 GO TO 1016
1013 IF F1*F2<0 GO TO 1015
1014 A1=X1 \ F1=F2 \ GO TO 1017
1015 B1=X1 \ GO TO 1017
1016 A1=X1 \ B1=X1
1017 IF B1-A1>E GO TO 1010
1018 RETURN
```

```
2000 REM ПОДПРОГРАММЫ - ЛЕВЫЕ ЧАСТИ УРАВНЕНИЙ
2001 REM
2002 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: X
2003 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: R
2004 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: НЕТ
2005 REM
2006 R=COS(2/X)-2*SIN(1/X)+1/X \ RETURN
2007 R=SIN(LOG(X))-COS(LOG(X))+2*LOG(X) \ RETURN
2008 R=EXP(X)+SQR(1+EXP(2*X))-2 \ RETURN

RUNNH

  ВВЕДИТЕ N
? 3

  ВВЕДИТЕ A,B,E
? 1,2,0.0001
КОРЕНЬ  1 УРАВНЕНИЯ =  1.87567

  ВВЕДИТЕ A,B,E
? 1,3,0.001
КОРЕНЬ  2 УРАВНЕНИЯ =  1.37402

  ВВЕДИТЕ A,B,E
? -1,0,0.0001
КОРЕНЬ  3 УРАВНЕНИЯ = -.287659

STOP AT LINE 180
```

Задание I уровня. Сформулировать, если нужно, задачу математически. Составить программу с использованием необходимых подпрограмм. Для составленных подпрограмм выписать список входных и выходных параметров и внутренних переменных подпрограммы. Подпрограммы составить так, чтобы они не портили входные параметры.

*Варианты вадач I уровня.*

1. Чего больше: всех возможных трехзначных чисел, записываемых цифрами 1, 2, 3, 4, 5; всех двузначных чисел, записываемых цифрами 2, 4, 6, 8; всех четырехзначных чисел, записываемых цифрами 1, 3, 7, 8, 9? Подсчет количества соответствующих чисел оформить в виде подпрограммы.

2. Сколькими способами можно отобрать команду в составе 5 человек из 8 кандидатов; из 10 кандидатов; из 11 кандидатов? Подсчет количества способов отбора оформить в виде подпрограммы.

3. В порт в среднем приходит 3 корабля в день. Какова вероятность того, что в порт в день придет 2 корабля; 4 корабля? Вычисление вероятности оформить в виде подпрограммы.

4. Два спортсмена одновременно начинают движение из одной точки. Первый спортсмен начинает движение со скоростью

10 км/час и равномерно (линейно) за каждый следующий час увеличивает скорость на 1 км. Второй начинает движение со скоростью 9 км/час и равномерно за каждый следующий час увеличивает скорость на 1,6 км/час. Выяснить, какой спортсмен преодолеет больший путь через 1 час; через 4 часа. Вычисление путей оформить с помощью функций.

5. В задаче 4 определить, когда второй спортсмен догонит первого. Решение квадратного уравнения оформить в виде подпрограммы.

6. В партии, состоящей из $k$ изделий, имеется $l$ дефектных. Из партии выбирается для контроля $r$ изделий. Найти вероятность того, что из них ровно $s$ изделий будут дефектными. Решить задачу для $k=10$, $l=5$, $r=4$, $s=2$ и для $k=10$, $l=4$, $r=5$, $s=3$. Вычисление $C_n^m$ оформить в виде подпрограммы.

7. Имеется квадратный лист бумаги со стороной $a$. Из листа делается коробка следующим образом: по углам листа вырезается четыре квадрата и коробка склеивается по швам (рис. 4.11). Какова должна быть сторона вырезаемого квадрата, чтобы коробка имела наибольшую вместимость? Решить задачу при $a=6$ см и $a=18$ см. Решение квадратного уравнения оформить в виде подпрограммы.

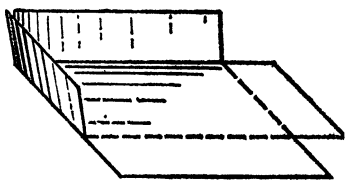


Рис. 4.11

8. Траектория снаряда, вылетающего из орудия под углом $\alpha$ с начальной скоростью $v_0$, описывается уравнениями

$$x = v_0 t \cos\alpha,$$

$$y = v_0 t \sin\alpha - \frac{gt^2}{2}.$$

С точностью $\Delta x=2$ км определить точку, в которой снаряд «уйдет под землю». Задачу решить при $\alpha=2\pi/6$, $v_0=35$ км/мин и при $\alpha=\pi/4$, $v_0=30$ км/мин. При решении задачи использовать функцию.

9. Два треугольника заданы своими сторонами $a$, $b$ и $c$ (т. е. заданы длины сторон $a$, $b$ и $c$). Вычислить площади треугольников по формуле Герона и определить, какой треугольник имеет бо́льшую площадь. При решении задачи взять следующие данные: для первого треугольника $a=3$, $b=4$, $c=5$; для второго треугольника $a=2$, $b=\sqrt{37}$, $c=\sqrt{37}$. Вычисление площади треугольника по формуле Герона оформить в виде функции.

10. Два треугольника заданы координатами своих вершин $A$, $B$ и $C$. Вычислить площади треугольников с помощью формулы Герона и определить, какой треугольник имеет бо́льшую площадь. При решении задачи использовать следующие данные: для первого

треугольника $A(1; 1)$, $B(4; 2)$, $C(2; 3{,}5)$; для второго треугольника $A(1; 2)$, $B(4; 1)$, $C(3; 3{,}5)$. Вычисление длин сторон треугольника и его площади по формуле Герона оформить в одной подпрограмме.

11. Два треугольника заданы координатами своих вершин $A$, $B$ и $C$. Вычислить площади треугольников, не используя формулу Герона, и определить, какой треугольник имеет бóльшую площадь. При решении задачи использовать следующие данные: для первого треугольника $A(1; 1)$, $B(5; 2)$, $C(3; 3)$; для второго треугольника $A(2; 5)$, $B(4; 3)$, $C(6; 4)$. Вычисление площади треугольника оформить в виде функции.

12. Футболист ударом ноги посылает мяч вертикально вверх с высоты 1 м с начальной скоростью 20 м/с. На какой высоте мяч будет через 1 с; 3 с; 4 с? Вычисление высоты оформить с помощью функции.

13. В задаче 12 определить, когда мяч будет на высоте 5 м; 10 м. Решение квадратного уравнения оформить в виде подпрограммы.

14. Используя схему Горнера, разделить многочлен $P_1(x)=x^4+2x^3-4x+3$ на двучлен $x+3$ и вычислить $P_1(-3)$; разделить $P_2(x)=x^5+6x^3-7x+2$ на $x-2$ и вычислить $P_2(2)$. Вычисление многочлена по схеме Горнера оформить в виде подпрограммы.

15. Вычислить коэффициенты третьей производной многочлена $x^7+6x^6+3x^4-5x^3$ и четвертой производной многочлена $x^9-7x^5+3x^4-2x^2+23$. Вычисление коэффициентов $r$-й производной многочлена $n$-й степени оформить в виде подпрограммы.

Указания к решению задач I уровня.

1. Количество $k$-значных чисел, составляемых из $n$ различных цифр (кроме 0), равно $A_n^k=\frac{n!}{(n-k)!}$, т. е. равно числу размещений из $n$ по $k$. Составить подпрограмму вычисления $A_n^k$. Вычисление $A_n^k$ можно производить как непосредственно по приведенной формуле, так и по формуле $A_n^k=\prod_{i=n-k+1}^{n} i$. Вторая формула предпочтительней (объясните, почему).

2. Выбрать $k$ человек из $n$ кандидатур можно $C_n^k$ способами, где $C_n^k=\frac{n!}{k!\,(n-k)!}$ — число сочетаний из $n$ по $k$. Составить подпрограмму вычисления $C_n^k$ по формуле $C_n^k=\frac{\prod_{i=n-k+1}^{n} i}{k!}$. Объясните, почему вторая формула предпочтительней первой.

3. Вероятность того, что в указанный порт в день придет $k$ кораблей, равна $P(k)=\frac{3^k e^{-3}}{k!}$. Составить подпрограмму вычисления вероятности по приведенной формуле.

4. Движение первого спортсмена описывается зависимостью $S=10t+0{,}5t^2$, движение второго описывается зависимостью $S=9t+0{,}8t^2$, где $S$ — пройденный путь, $t$ — время. В программе описать две функции $f_1(t)=10t+0{,}5t^2$ и $f_2(t)=9t+0{,}8t^2$ и сравнивать значения функций при соответствующих $t$. Предусмотреть печать необходимой текстовой информации.

5. Приравниваем пройденные пути (см. указание к задаче 4) и получаем уравнение $0{,}3t^2-t=0$. В программе предусмотреть анализ корней.

6. Искомая вероятность $P$ вычисляется по формуле $P=\dfrac{C_l^s\cdot C_{k-l}^{r-s}}{C_k^r}$. При вычислении $C_n^m$ см. указания к задаче 2.

7. Если обозначить через $x$ сторону вырезаемых квадратиков, то получающаяся коробка будет иметь квадратное основание со стороной $a-2x$ и высоту $x$. Следовательно, вместимость коробки (обозначим ее $f(x)$) будет равна

$$f(x)=(a-2x)^2\,x=4x^3-4ax^2+a^2x.$$

Далее, из необходимого условия максимума ($f'(x)=0$) получаем уравнение $12x^2-8ax+a^2=0$. Решаем полученное уравнение при соответствующих $a$ и анализируем корни. В программе предусмотреть анализ корней.

8. Исключив $t$, найдем уравнение траектории в виде

$$y=y(x)=x\operatorname{tg}\alpha-\frac{gx^2}{2v_0^2\cos^2\alpha}.$$

Теперь, начиная с $x=0$, с шагом $\Delta x=2$ вычисляем значения $y(x)$ и проверяем условие $y(x)<0$. Точка, в которой выполнится указанное условие, и есть решение. В задаче принять $g=9{,}8$ м/с$^2$. Обратить внимание на соответствие единиц измерения! Объясните, почему при решении данной задачи на ЭВМ предпочтительней пользоваться км и мин, чем м и с. Оформить $y(x)$ в виде функции.

9. По формуле Герона площадь треугольника со сторонами $a$, $b$, $c$ равна

$$S=\sqrt{p(p-a)(p-b)(p-c)},$$

где $p$ — полупериметр треугольника, т. е. $p=(a+b+c)/2$. В программе предусмотреть печать необходимой текстовой информации.

10. Длина отрезка, соединяющего точки $P(x_1, y_1)$ и $Q(x_2, y_2)$, вычисляется по формуле

$$|\overline{PQ}|=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}.$$

Далее см. указание к задаче 9.

11. Площадь треугольника, заданного вершинами $A(x_1, y_1)$, $B(x_2, y_2)$ и $C(x_3, y_3)$, вычисляется по формуле

$$S=\frac{1}{2}|(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)|.$$

Предусмотреть печать необходимой текстовой информации.

12. Движение мяча описывается зависимостью $y=y(t)=y_0+v_0t-gt^2/2$ ($x=\text{const}$), где $y_0$ — начальная высота, $v_0$ — начальная скорость, $t$ — время, $y(t)$ — высота в момент $t$. В задаче принять $g=9{,}8$ м/с$^2$. Оформить $y(t)$ в виде функции. В программе предусмотреть печать необходимой текстовой информации.

13. Время, за которое мяч окажется на высоте $h$, определяется из квадратного уравнения $4{,}9t^2-20t+(h-1)$. В программе предусмотреть анализ корней. Объясните, в чем смысл двух корней.

14. Согласно схеме Горнера при делении многочлена $n$-й степени $a_0x^n+a_1x^{n-1}+\ldots+a_n$ на линейный двучлен $x-c$ коэффициенты частного $b_0x^{n-1}+b_1x^{n-2}+\ldots+b_{n-1}$ и остаток $r$ вычисляются по формулам $b_0=a_0$, $b_k=cb_{k-1}+a_k$, $k=1, 2, \ldots, n-1$, и $r=cb_{n-1}+a_n$. При этом значение исходного многочлена при $x=c$ равно остатку $r$. В программе необходимо предусмотреть ввод коэффициентов исходного многочлена в виде массива и передачу этого массива в подпрограмму. Результатом работы должен быть массив коэффициентов частного и остаток. Обратите внимание на то, что в массиве коэффициентов должны содержаться и нулевые коэффициенты.

15. Если дан многочлен $n$-й степени $a_0x^n+a_1x^{n-1}+\ldots+a_n$, то коэффициенты его $r$-й производной $b_0x^{n-r}+b_1x^{n-r-1}+\ldots+b_{n-r}$ вычисляются через коэффициенты исходного многочлена по формулам $b_k=a_k\prod_{i=0}^{r-1}(n-k-i)$, $k=0, 1, \ldots, n-r$. В программе необходимо предусмотреть ввод коэффициентов исходного многочлена в виде массива и передачу этого массива в подпрограмму. Результатом работы должен быть массив коэффициентов производной.

Задание II уровня. Задачи второго уровня требуют использования двух подпрограмм или подпрограммы и функции. При решении задачи необходимо выполнить все требования I уровня. Для составленных подпрограмм нарисовать схему их взаимосвязи и их связи с основной программой.

*Варианты задач II уровня.*

1. Определить вероятности того, что среди пяти детей одной семьи нет ни одной девочки; одна девочка; две девочки; три девочки; четыре девочки; пять девочек. Выясните, вероятность скольких девочек будет максимальной. Вероятности рождения мальчика и девочки предполагаются одинаковыми. Вычисление соответствующей

вероятности оформить в виде подпрограммы. Поиск максимального элемента массива оформить в виде подпрограммы.

2. Определить вероятность того, что в семье, имеющей 6 детей, не больше четырех девочек. Вероятности рождения мальчика и девочки предполагаются одинаковыми. Вычисление вероятности того, что из $n$ детей $m$ девочек ($m \leqslant n$), оформить в виде подпрограммы. Суммирование элементов массива длины $l$ оформить в виде подпрограммы.

3. Разделить третью производную многочлена $x^8+2x^7+3x^6-7x^4-5x^3-27$ на двучлен $x-3$ и вычислить значение этой производной при $x=3$. Разделить вторую производную многочлена $x^7-x^6+2x^5-2x^4-13$ на $x+4$ и вычислить значение этой производной при $x=-4$. Использовать схему Горнера. Вычисление коэффициентов производной оформить в виде подпрограммы. Вычисление многочлена по схеме Горнера оформить в виде подпрограммы.

4. В задаче 12 уровня I определить с точностью до 0,25 с, когда и на какой максимальной высоте окажется мяч в течение 4 с. Вычисление высоты оформить с помощью функции. Поиск максимального элемента массива оформить в виде подпрограммы.

5. В задаче 12 уровня I определить с точностью до 0,25 с, когда, на какой высоте и какой будет максимальная скорость мяча в течение 4 с. Вычисление высоты оформить с помощью функции. Поиск максимального по модулю элемента массива оформить в виде подпрограммы.

6. Круг задан координатами центра $Q$ и координатами одной из точек окружности (точка $Z$). Внутри круга содержится треугольник, заданный координатами своих вершин $A$, $B$, $C$. Произвольно выбирается точка внутри круга. Найти вероятность того, что эта точка попадет в треугольник. Предполагается, что вероятность попадания точки в часть круга пропорциональна площади этой части и не зависит от расположения внутри круга. Задачу решить для $Q(4, 5)$, $Z(7, 5)$, $A(2, 4)$, $B(5, 5)$, $C(3, 6)$ и $Q(5, 4)$, $Z(5, 7)$, $A(3, 3)$, $B(7, 5)$, $C(5, 6)$. Вычисление площади круга и площади треугольника оформить в виде подпрограмм.

7. Вычислить приближенно площадь фигуры, ограниченной осью $x$, прямыми $x=1$ и $x=3$ и кривой $y=y(x)=1/x+5$, разбивая интервал изменения $x$ на 10 частей и суммируя площади десяти прямоугольников с основанием 0,2 и высотой, равной значению функции в середине каждого интервала. Вычисление $y(x)$ оформить с помощью функции. Вычисление площади оформить в виде подпрограммы.

8. В задаче 4 уровня I определить с точностью до 0,25 часа, когда и каким окажется максимальное расстояние между спортсменами в течение 5 часов. Вычисление расстояния оформить с помощью

функции. Поиск максимального по модулю элемента массива оформить в виде подпрограммы.

9. В задаче 6 уровня I найти вероятность того, что партия будет забракована (если среди контрольных изделий окажется более $s$ дефектных, бракуется вся партия). Решить задачу для $k=10$, $l=5$, $r=4$, $s=3$. Вычисление вероятности появления среди выбранных изделий ровно $i$ дефектных оформить в виде подпрограммы. Суммирование элементов массива оформить в виде подпрограммы.

10. Вычислить приближенно площадь фигуры, ограниченной осью $x$, прямыми $x=1$ и $x=3$ и кривой $y=y(x)=x^2/(1+x^2)$, разбивая интервал изменения $x$ на 10 частей и суммируя площади десяти прямоугольников с основанием 0,2 и высотой, равной значению функции на левой границе его основания. Вычисление $y(x)$ оформить с помощью функции. Вычисление площади оформить в виде подпрограммы.

11. Круг задан координатами центра $Q$ и координатами одной из точек окружности (точка $Z$). Внутри круга содержится квадрат, заданный координатами трех вершин $A$, $B$, $C$. Произвольно выбрана точка внутри круга. Найти вероятность того, что эта точка попадет в квадрат. Предполагается, что вероятность попадания точки в часть круга пропорциональна площади этой части и не зависит от расположения внутри круга. Задачу решить для $Q(4, 5)$, $Z(7, 5)$, $A(2, 4)$, $B(4, 6)$, $C(2, 6)$ и $Q(5, 4)$, $Z(5, 7)$, $A(5, 3)$, $B(3{,}5; 4{,}5)$, $C(5, 6)$. Вычисление площади круга оформить в виде подпрограммы. Вычисление площади квадрата оформить в виде подпрограммы.

12. В задаче 8 уровня I определить с точностью до 5 с, когда и на какой максимальной высоте окажется снаряд в течение 1 мин. Задачу решить при $\alpha=\pi/4$ и $v_0=30$ км/мин. Вычисление высоты оформить с помощью функции. Поиск максимального элемента массива оформить в виде подпрограммы.

13. Стрелок производит по мишени 5 выстрелов. Вероятность попадания в мишень при каждом выстреле 0,6. Вычислить вероятности того, что стрелок не попадет в мишень ни разу; попадет 1 раз; 2 раза; 3 раза; 4 раза; 5 раз. Выяснить, вероятность скольких попаданий будет максимальной. Вычисление вероятности и поиск максимального элемента массива оформить в виде подпрограмм.

14. В задаче 13 вычислить вероятность того, что стрелок попадет в мишень не более 3 раз. Вычисление вероятности $m$ попаданий при $n$ выстрелах оформить в виде подпрограммы. Суммирование элементов массива оформить в виде подпрограммы.

15. Треугольник задан координатами своих вершин $A$, $B$ и $C$. С точностью $10^{-4}$ проверить равенство площадей треугольников $ABP$, $ACP$ и $BCP$, где $P$— точка пересечения медиан треугольника $ABC$. Проверку осуществить для треугольника $A(1,1)$, $B(4,2)$, $C(2,3)$ и треугольника $A(3,2)$, $B(4,1)$, $C(5,3)$. Вычисление

координат точки пересечения медиан оформить в виде подпрограммы. Вычисление площади треугольника оформить в виде подпрограммы.

Указания к решению задач II уровня.

1. Вероятность рождения девочки $p=0,5$, мальчика — $q=1-p$. Вероятность того, что в семье, имеющей $n$ детей, будет $m$ девочек, равна $C_n^m p^m q^{n-m}$. Необходимо составить подпрограмму вычисления приведенной формулы. Далее, найденные с помощью подпрограммы вероятности необходимо поместить в массив и составить подпрограмму поиска максимального элемента массива. Алгоритм поиска максимального элемента массива см. в работе 4, но, в отличие от алгоритма работы 4, здесь необходимо найти все максимальные элементы массива (если их несколько). При вычислении $C_n^m$ см. указание к задаче 2 уровня I. В программе предусмотреть печать необходимой текстовой информации.

2. См. указания к задаче 1. Алгоритм суммирования элементов массива см. в работе 4. В данной задаче для того, чтобы определить вероятность рождения не более $k$ девочек, необходимо вычислить вероятности того, что в семье не будет ни одной девочки, будет 1 девочка, 2 девочки и т. д. до $k$ девочек включительно, и затем просуммировать вычисленные вероятности. В программе предусмотреть печать необходимой текстовой информации.

3. См. указания к задачам 14 и 15 уровня I.

4. Оформив вычисление высоты $y(t)$ с использованием функции (см. указания к задаче 12 уровня I), получаем массив значений $y(t)$ на отрезке [0; 4] с шагом 0,25. Затем в полученном массиве ищем максимальный элемент, это и будет максимальная высота. Для определения нужного момента времени необходимо из подпрограммы поиска максимального элемента получить еще и номер $i$ найденного элемента, тогда искомый момент времени равен $0,25 \cdot i$. Алгоритм поиска максимального элемента массива приведен в работе 4. В программе предусмотреть печать необходимой текстовой информации.

5. Скорость мяча равна $v_0 - gt$ (см. указания к задаче 12 уровня I). В программе необходимо построить массив значений скорости (со знаком) на отрезке [0; 4] с шагом 0,25. Затем в полученном массиве найти максимальный по модулю элемент и его номер $i$ (это должна делать подпрограмма). Далее, искомый момент времени $t^*$ определяется как $t^*=0,25 \cdot i$, а искомая высота определяется как $y(t^*)$. В программе предусмотреть печать необходимой текстовой информации.

6. Для определения площади круга необходимо вычислить его радиус, это расстояние между точками $Q$ и $Z$, т. е. вычислить расстояние между двумя точками (см. указания к задаче 10 уровня I). Вычислить площадь треугольника (см. указания к задаче 11 уров-

ня I). Далее, искомая вероятность равна отношению площади треугольника к площади круга.

7. Подпрограмму вычисления площади фигуры необходимо составить в общем виде, т. е. для любого интервала изменения $x$ и любого количества частей разбиения. Подпрограмма должна зависеть только от интервала изменения $x$ и количества частей разбиения и не зависеть от вида конкретной функции.

8. Расстояние между спортсменами равно разности пройденных ими путей, т. е. равно $y(t)=t-0{,}3t^2$ (см. указания к задаче 4 уровня I). Объясните смысл знака расстояния. Далее, задача решается аналогично задаче 5 (см. указания).

9. Искомая вероятность $P$ равна

$$P=\sum_{i=s+1}^{r} p_i=1-\sum_{i=0}^{s} p_i,$$

где $p_i=\dfrac{C_l^i\cdot C_{k-l}^{r-i}}{C_k^r}$ — вероятность появления среди выбранных изделий ровно $i$ дефектных (см. указание к задаче 6 уровня I). Вычисление $p_i$ и суммирование $p_i$ от $i=s+1$ до $i=r$ либо от $i=0$ до $i=s$ оформить в виде подпрограмм.

10. См. указания к задаче 7.

11. Искомая вероятность равна отношению площади квадрата к площади круга. Площадь квадрата равна удвоенной площади треугольника, построенного по трем заданным вершинам. Далее см. указания к задаче 6.

12. Высота есть координата $y$ и определяется по формуле $y(t)=v_0t\sin\alpha-gt^2/2$. Далее см. указания к задаче 4.

13. Вероятность попадания в мишень при одном выстреле равна 0,6, тогда вероятность промаха $q=0{,}4$. Вероятность того, что стрелок при $n$ выстрелах попадет $m$ раз в мишень, равна $C_n^m p^m q^{n-m}$. Далее см. указание к задаче 1.

14. См. указания к задаче 13 и задаче 2.

15. Если треугольник задан координатами своих вершин $M_1(x_1, y_1)$, $M_2(x_2, y_2)$ и $M_3(x_3, y_3)$, то координаты точки пересечения медиан треугольника (точка $M(x_M, y_M)$) определяются по формулам

$$x_M=\frac{x_1+x_2+x_3}{3},$$

$$y_M=\frac{y_1+y_2+y_3}{3}.$$

См. также указания к задаче 11 уровня I.

Задание III уровня. Задачи III уровня требуют использования трех или более подпрограмм и (или) функций. Для задач III уровня взаимосвязи подпрограмм и их связи с основной

программой приобретают более сложный характер и требуют для их корректной реализации элементов творческого подхода. При решении задач необходимо выполнить все требования задания II уровня. Для составленных подпрограмм написать инструкции по их использованию. Во всех программах III уровня предусмотреть печать необходимой текстовой информации.

*Варианты вадач III уровня.*

1. Пять спортсменов стартуют одновременно из одной точки с начальными скоростями 10 км/час, 9,5 км/час, 9,25 км/час, 9 км/час, 8,5 км/час и равномерно (линейно) за каждый следующий час увеличивают свою скорость на 1 км, 1,2 км, 1,4 км, 1,6 км и 1,8 км соответственно. Выяснить, какие спортсмены будут друг от друга на максимальном и какие на минимальном расстоянии через 2,25 часа; через 4 часа. Вычисление матрицы расстояний между спортсменами оформить в виде подпрограммы. Поиск максимального расстояния оформить в виде подпрограммы. Поиск минимального расстояния оформить в виде подпрограммы.

2. В задаче 1 выяснить, у каких спортсменов будет максимальная и у каких будет минимальная разность скоростей через 2 часа; через 4 часа. Вычисление матрицы разности скоростей оформить в виде подпрограммы. Поиск максимальной разности скоростей оформить в виде подпрограммы. Поиск минимальной разности скоростей оформить в виде подпрограммы.

3. Решить задачу 6 уровня II, осуществляя проверку на принадлежность треугольника кругу. Задачу решить для $Q(4,5)$, $Z(7,5)$, $A(2,4)$, $B(6,5)$, $C(3,6)$ и $Q(5,4)$, $Z(5,7)$, $A(3,3)$, $B(2,1)$, $C(1,2)$. В случае, если треугольник не принадлежит полностью кругу, выдавать на печать соответствующее сообщение. Проверку на принадлежность треугольника кругу оформить в виде подпрограммы. Вычисление площади круга оформить в виде подпрограммы. Вычисление площади треугольника оформить в виде подпрограммы.

4. Решить задачу 11 уровня II, осуществляя проверку на принадлежность квадрата кругу. Задачу решить для $Q(4,5)$, $Z(7,5)$, $A(2,4)$, $B(4,6)$, $C(2,6)$ и $Q(5,4)$, $Z(5,7)$, $A(5,5; 5)$, $B(7,5; 3)$, $C(6,5; 6)$. Вычисление координат четвертой вершины квадрата оформить в виде подпрограммы. Проверку на принадлежность квадрата кругу оформить в виде подпрограммы. Вычисление площади круга оформить в виде подпрограммы. Вычисление площади треугольника оформить в виде подпрограммы.

5. На плоскости заданы координаты пяти точек $A$, $B$, $C$, $D$ и $E$. Выяснить, какие точки находятся на максимальном и минимальном расстояниях друг от друга, и вычислить сумму всех расстояний между точками. Задачу решить для $A(1,1)$, $B(2,3)$, $C(4,3)$, $D(5,6)$, $E(6,4)$ и $A(1,2)$, $B(2,4)$, $C(3,3)$, $D(5,1)$, $E(5,5)$. Вычисление

матрицы расстояний между точками оформить в виде подпрограммы. Поиск максимального расстояния оформить в виде подпрограммы. Поиск минимального расстояния оформить в виде подпрограммы. Суммирование расстояний оформить в виде подпрограммы.

6. Пять орудий дают залп из одной точки. Стволы орудий расположены под углами $\alpha_i$ ($i=1, 2, 3, 4, 5$) к горизонту. Снаряды вылетают из стволов орудий с начальными скоростями $v_0^i$ ($i=1, 2, 3, 4, 5$). Выяснить, какие снаряды при падении на землю будут на максимальном и какие на минимальном расстоянии друг от друга. Задачу решить при $\alpha_1=\pi/6$, $\alpha_2=\pi/5$, $\alpha_3=\pi/7$, $\alpha_4=\pi/8$, $\alpha_5=2\pi/15$, $v_0^1=30$ км/мин, $v_0^2=32$ км/мин, $v_0^3=29$ км/мин, $v_0^4=31$ км/мин, $v_0^5=28$ км/мин и при $\alpha_1=\pi/4$, $\alpha_2=3\pi/10$, $\alpha_3=\pi/5$, $\alpha_4=7\pi/20$, $\alpha_5=3\pi/20$, $v_0^1=35$ км/мин, $v_0^2=33$ км/мин, $v_0^3=34$ км/мин, $v_0^4=36$ км/мин, $v_0^5=37$ км/мин. Вычисление матрицы расстояний между снарядами оформить в виде подпрограммы. Поиск максимального расстояния оформить в виде подпрограммы. Поиск минимального расстояния оформить в виде подпрограммы.

7. Выпуклый четырехугольник задан координатами своих вершин $A$, $B$, $C$ и $D$ в порядке их обхода. Середины соседних сторон четырехугольника соединяются отрезками. Проверить, является ли полученный четырехугольник параллелограммом с площадью, равной половине исходного четырехугольника. Проверку осуществить с точностью $10^{-4}$ для четырехугольника $A$ (1,1), $B$ (2,3), $C$ (5,4), $D$ (6,2) и четырехугольника $A$ (1, 2), $B$ (3, 4), $C$ (5, 3), $D$ (4, 1). Вычисление координат середин сторон четырехугольника оформить в виде подпрограммы. Проверку того, является ли четырехугольник параллелограммом, оформить в виде подпрограммы. Вычисление площади четырехугольника оформить в виде подпрограммы.

В задачах 8—13 для вычисления значений многочленов использовать схему Горнера, которую оформить в виде подпрограммы.

8. При $x$, изменяющемся от 0 до 3 с шагом 0,5, найти максимальное значение третьей производной многочлена $x^7+6x^6-5x^5+x^3-x+7$ и минимальное значение четвертой производной того же многочлена. Вычисление коэффициентов производной оформить в виде подпрограммы. Поиск максимального элемента массива оформить в виде подпрограммы. Поиск минимального элемента оформить в виде подпрограммы.

9. При $x$, изменяющемся от $-2$ до 3 с шагом 0,5, найти максимальное значение суммы многочлена $x^8-7x^7+5x^5-3x^4+27$ и его третьей производной. Вычисление коэффициентов $r$-й производной многочлена степени $n$ оформить в виде подпрограммы. Вычисление коэффициентов суммы двух многочленов оформить в виде подпрограммы. Поиск максимального элемента массива оформить в виде подпрограммы.

10. При $x$, изменяющемся от —3 до 2 с шагом 0,5, вычислить значения суммы многочлена $x^6-5x^5+4x^4-x^2+1$ и его второй производной. Затем просуммировать полученные значения. Вычисление коэффициентов $r$-й производной многочлена степени $n$ оформить в виде подпрограммы. Вычисление коэффициентов суммы двух многочленов оформить в виде подпрограммы. Суммирование элементов массива оформить в виде подпрограммы.

11. При $x$, изменяющемся от —2 до 2 с шагом 0,5, вычислить значения четвертой производной многочлена $x^8-7x^7+6x^6-x^2+25$ и просуммировать полученные значения. Вычисление коэффициентов $r$-й производной многочлена степени $n$ оформить в виде подпрограммы. Суммирование элементов массива оформить в виде подпрограммы.

12. При $x$, изменяющемся от —3 до 3 с шагом 0,5, найти максимальное значение третьей производной от суммы многочленов $x^7+2x^6-5x^5-3x^2+27$ и $x^5-4x^4+x^3-2x+3$. Вычисление коэффициентов суммы двух многочленов оформить в виде подпрограммы. Вычисление коэффициентов $r$-й производной многочлена степени $n$ оформить в виде подпрограммы. Поиск максимального элемента массива оформить в виде подпрограммы.

13. При $x$, изменяющемся от —2 до 2 с шагом 0,5, найти сумму значений четвертой производной от суммы многочленов $x^6-5x^5+x^4-27$ и $x^7-3x^6+7x^4-2x^2+1$. Вычисление коэффициентов суммы двух многочленов оформить в виде подпрограммы. Вычисление коэффициентов $r$-й производной многочлена степени $n$ оформить в виде подпрограммы. Суммирование элементов массива оформить в виде подпрограммы.

В задачах 14—15 для решения уравнений использовать метод половинного деления, который оформить в виде подпрограммы.

14. Вычислить приближенно площадь фигуры, ограниченной частью кривой $y=y(x)=x+\sqrt{x}+\sqrt[3]{x}-2{,}5$, лежащей в верхней полуплоскости, осью абсцисс и прямой $x=2$. Для определения левого конца интервала изменения $x$ найти корень уравнения $x+\sqrt{x}+\sqrt[3]{x}-2{,}5=0$ на отрезке [0,5; 1] с точностью $10^{-4}$. Для вычисления площади интервал изменения $x$ разделить на 10 частей, на полученных отрезках построить прямоугольники с высотой, равной значению $y(x)$ в середине каждого отрезка, и просуммировать площади прямоугольников. Вычисление $y(x)$ производить с помощью функции. Вычисление площади оформить в виде подпрограммы.

15. Вычислить приближенно площадь фигуры, ограниченной частью кривой $y=y(x)=0{,}6\cdot 3^x-2{,}3\cdot x-3$, лежащей в четвертой четверти, и осями координат. Для определения правого конца интервала изменения $x$ найти корень уравнения $0{,}6\cdot 3^x-2{,}3x-3=0$ на отрезке [2; 3] с точностью $10^{-4}$. Для вычисления площади интервал

изменения $x$ разделить на 10 частей, на полученных отрезках построить прямоугольники с высотой, равной значению $|y(x)|$ в левом конце каждого отрезка, и просуммировать площади прямоугольников. Вычисление $y(x)$ производить с помощью функции. Вычисление площади оформить в виде подпрограммы.

Указания к решению задач III уровня.

1. Движения спортсменов описываются зависимостями $s_1=10t+0{,}5t^2$, $s_2=9{,}5t+0{,}6t^2$, $s_3=9{,}25t+0{,}7t^2$, $s_4=9t+0{,}8t^2$ и $s_5=8{,}5t+0{,}9t^2$, где $s_i$ — путь, пройденный $i$-м спортсменом за время $t$. В программе предусмотреть два массива V и W. В массив V поместить коэффициенты при $t$ (т. е. $v_i$ есть начальная скорость $i$-го спортсмена), в массив W поместить коэффициенты при $t^2$ (т. е. $w_i$ есть половина ускорения $i$-го спортсмена). Теперь расстояние между $i$-м и $j$-м спортсменами через время $t$ будет равно $s_i-s_j=(v_i-v_j)t+(w_i-w_j)t^2$. Используя эту формулу, сформируем матрицу расстояний R между спортсменами в момент $t$, т. е. $R_{ij}$ есть расстояние между $i$-м и $j$-м спортсменами. Указанную матрицу необходимо формировать с учетом зависимости от $t$ в подпрограмме, далее, для заданных $t$ (2,25 часа и 4 часа) найти максимальный и минимальный по модулю элементы матрицы R (см. работу 4). Учесть, что в матрице R элементы $R_{ii}=0$ и $R_{ij}=-R_{ji}$. Объясните смысл знака $R_{ij}$. Индексы найденных максимального и минимального элементов и есть искомый ответ.

2. Задача решается аналогично задаче 1 с учетом того, что разность скоростей $i$-го и $j$-го спортсменов в момент $t$ равна $v_i-v_j+2(w_i-w_j)t$.

3. Треугольник принадлежит кругу, если все три вершины принадлежат кругу. Точка принадлежит кругу, если расстояние от нее до центра круга не больше радиуса круга. См. также указания к задаче 10 уровня I и к задаче 6 уровня II.

4. Координаты четвертой вершины квадрата можно вычислить следующим образом. Из трех заданных вершин $A(x_A, y_A)$, $B(x_B, y_B)$ и $C(x_C, y_C)$ выявляем две, лежащие на одной диагонали квадрата, т. е. принадлежащие наибольшей из сторон треугольника $ABC$. Пусть это оказались вершины $B$ и $C$. Далее определяем координаты $(x_0, y_0)$ середины диагонали по формулам $x_0=\frac{x_B+x_C}{2}$ и $y_0=\frac{y_B+y_C}{2}$. Теперь определяем координаты $(x_D, y_D)$ четвертой вершины $D$ по формулам $x_D=2x_0-x_A$ и $y_D=2y_0-y_A$. Или сразу имеем $x_D=x_B+x_C-x_A$ и $y_D=y_B+y_C-y_A$. Далее, квадрат принадлежит кругу, если все его четыре вершины принадлежат кругу. Далее см. указания к задаче 3 и задаче 11 уровня II.

5. Формируем матрицу R расстояний между точками по формуле $R_{ij}=\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}$, где $(x_i, y_i)$ — координаты $i$-й точки. Для формирования матрицы $R_{ij}$ предусмотреть в программе

массивы X и Y (в массиве X содержатся абсциссы точек, в Y — ординаты). Алгоритмы поиска минимального и максимального элементов, суммирования элементов массива приведены в работе 4. Учесть, что в матрице R элементы $R_{ii}=0$ и $R_{ij}=R_{ji}$.

6. Точка падения $i$-го снаряда $x_i$ определяется по формуле $x_i=\frac{(v_0^i)^2 \sin 2\alpha_i}{g}$ (см. указания к задаче 8 уровня I). Теперь формируем матрицу R расстояний между снарядами по формуле $R_{ij}=|x_j-x_i|$. Алгоритм поиска минимального (максимального) элемента массива приведен в работе 4. Учесть, что в матрице R элементы $R_{ii}=0$ и $R_{ij}=R_{ji}$.

7. Координаты $(x_c, y_c)$ середины отрезка, соединяющего точки $M(x_M, y_M)$ и $N(x_N, y_N)$, вычисляются по формулам $x_c=\frac{x_M+x_N}{2}$, $y_c=\frac{y_M+y_N}{2}$. Чтобы проверить, является ли четырехугольник параллелограммом, достаточно проверить равенство противоположных сторон. Площадь четырехугольника равна сумме площадей двух треугольников, на которые диагональ делит четырехугольник. Площадь треугольника вычислить в соответствии с указанием к задаче 11 уровня I.

8. См. указания к задачам 14 и 15 уровня I и работу 4.

9. См. указания к задачам 14 и 15 уровня I и работу 4. При составлении подпрограммы вычисления коэффициентов суммы двух многочленов предусмотреть выявление многочлена большей степени. Обратите внимание на суммирование двух массивов разной длины.

10. См. указания к задаче 9.

11. См. указания к задачам 14 и 15 уровня I и работу 4.

12. См. указания к задаче 9.

13. См. указания к задаче 9.

14. См. пример и указания к задаче 7 уровня II.

15. См. пример и указания к задаче 7 уровня II. При вычислении площади обратите внимание на знак функции.

Вопросы для самопроверки.

1. В каких случаях целесообразно использовать подпрограммы?
2. Оформление подпрограммы. Обращение к подпрограмме. Операторы GOSUB и RETURN.
3. Каким образом организовать программу, чтобы не допустить выполнение оператора RETURN без обращения к подпрограмме?
4. Как осуществляется обмен информацией между основной программой и подпрограммой? Входные и выходные переменные.
5. Что такое внутренние и прочие переменные?
6. Оператор ON GOSUB. Особенности его выполнения.
7. Взаимосвязь основной программы и подпрограммы. Как осуществляется взаимодействие между различными подпрограммами в рамках одной программы?

8. Цель и содержание инструкции к подпрограммам.
9. В каких случаях можно и целесообразно использовать функцию?
10. Определение функции в программе. Оператор DEF FN. Обращение к функции.
11. Метод половинного деления. Условие прекращения процесса.
12. В чем смысл операторов в строке 1014 в программе 4.34?
13. Как организовать выбор необходимого уравнения в программе 4.35?

## *Работа 6.* ИСПОЛЬЗОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ МЕТОДОВ

Теоретическое введение. При разработке алгоритмов решения конкретных инженерных задач часто приходится прибегать к методам численного анализа. Рассмотрим наиболее простые численные методы решения некоторых классов задач, которые используются в данной работе. Для рассматриваемых методов приводятся реализующие их подпрограммы с необходимыми инструкциями. Использование некоторых методов иллюстрируется примерами. При этом с целью достижения большей наглядности применения самого метода в программах не предусматривается контроль вводимой информации. В работе также не рассматриваются в полном объеме специальные вопросы численного анализа (устойчивость, обусловленность, сходимость и т. д.).

*1. Интерполирование функций.*

На практике часто приходится иметь дело с функциями, заданными таблично, когда для значений аргумента $x_1, x_2, \ldots, x_n$ известны значения функции $y_1=f(x_1)$, $y_2=f(x_2)$, $\ldots$, $y_n=f(x_n)$. Для того чтобы определить значение функции $f$ в какой-либо точке $x$, отличной от заданных $x_1, x_2, \ldots, x_n$, поступают следующим образом: строят функцию $F$, которая в заданных точках $x_1, x_2, \ldots, x_n$ совпадает с заданными значениями $y_1, y_2, \ldots, y_n$, т. е.

$$F(x_i)=y_i,\ i=1,\ 2,\ \ldots,\ n,$$

а при остальных $x$ приближенно представляет функцию $f$. При этом функция $F$ называется интерполирующей, а точки $x_1, x_2, \ldots, x_n$ называются узлами интерполяции.

Чаще всего функцию $F$ задают в виде многочлена, называемого *интерполяционным многочленом Лагранжа* (обозначается $L_n(x)$). Интерполяционный многочлен Лагранжа, построенный по таблице $(x_1, y_1)$, $(x_2, y_2)$, $\ldots$, $(x_n, y_n)$, имеет вид

$$L_n(x)=\sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}. \qquad (1)$$

Часто интерполяционный многочлен строится не по всем заданным узлам $x_1, x_2, \ldots, x_n$, а по некоторым $m$ ($m<n$) выбранным узлам $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$. В этом случае многочлен строится по формуле (1) при $n=m$. При этом, как правило, в качестве узлов интерполяции выбирается $m$ ближайших к $x$ узлов, где $x$ — точка, в которой отыскивается приближенное значение функции $f$.

Далее рассматриваются случаи $m=2$ (линейное интерполирование) и $m=3$ (квадратичное интерполирование).

1.1. *Линейное интерполирование.* Пусть задана таблица $(x_1, y_1)$, $(x_2, y_2)$, . . ., $(x_n, y_n)$, в которой все $x_i$ различны. Тогда приближенное значение функции $f$ в точке $x$ ($y \approx f(x)$) вычисляется по формуле

$$y = y_{i_1}\frac{x-x_{i_2}}{x_{i_1}-x_{i_2}} + y_{i_2}\frac{x-x_{i_1}}{x_{i_2}-x_{i_1}} = y_{i_1} + (y_{i_2}-y_{i_1})\frac{x-x_{i_1}}{x_{i_2}-x_{i_1}}, \qquad (2)$$

где $x_{i_1}$ и $x_{i_2}$ — узлы, определяемые либо по правилу 1), либо по правилу 2).

Правило 1): $x_{i_1}$ и $x_{i_2}$ — ближайшие к $x$ узлы из набора $x_1, x_2, \ldots, x_n$.

Правило 2): а) если $x \leqslant \min\{x_1, x_2, \ldots, x_n\}$ или $x \geqslant \max\{x_1, x_2, \ldots, x_n\}$, то $x_{i_1}$ и $x_{i_2}$ — ближайшие к $x$ узлы из набора $x_1, x_2, \ldots, x_n$; б) если $\min\{x_1, x_2, \ldots, x_n\} < x < \max\{x_1, x_2, \ldots, x_n\}$, то $x_{i_1}$ — ближайший к $x$ узел такой, что $x_{i_1} \leqslant x$, а $x_{i_2}$ — ближайший к $x$ узел такой, что $x_{i_2} \geqslant x$ ($x_{i_1}$ и $x_{i_2}$ берутся из набора $x_1, x_2, \ldots, x_n$).

Замечание. В случае, когда узлы $x_1, x_2, \ldots, x_n$ расположены равномерно (т. е. с постоянным шагом), оба правила дают одинаковый результат.

Далее приводится подпрограмма линейного интерполирования (программа 4.36). При этом до обращения к подпрограмме таблица $(x_1, y_1)$, $(x_2, y_2)$, . . ., $(x_n, y_n)$ должна быть перестроена так, чтобы $x_1 = x_{i_1}$, $y_1 = y_{i_1}$, $x_2 = x_{i_2}$, $y_2 = y_{i_2}$, где $x_{i_1}$ и $x_{i_2}$ определены либо по правилу 1), либо по правилу 2).

**Программа 4.36**

```
1000 REM ПОДПРОГРАММА "ЛИНЕЙНАЯ ИНТЕРПОЛЯЦИЯ"
1001 REM
1002 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: X0,X(1),X(2),Y(1),Y(2)
1003 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: L,Y0
1004 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: НЕТ
1005 REM
1006 REM ПРОВЕРКА ТАБЛИЦЫ
1007 L=0
1008 IF X(1)<>X(2) GO TO 1010
1009 L=1 \ Y0=Y(1) \ GO TO 1012
1010 REM ВЫЧИСЛЕНИЕ ИНТЕРПОЛЯЦИОННОГО ЗНАЧЕНИЯ
1011 Y0=Y(1)+(Y(2)-Y(1))*(X0-X(1))/(X(2)-X(1))
1012 RETURN
```

Инструкция к подпрограмме «Линейная интерполяция».

*Назначение*: вычисление интерполяционного значения $y$ в точке $x$ с использованием линейного интерполирования.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: X0 — значение $x$, при котором вычисляется интерполяционное значение $y$; X(1), X(2) — табличные значения аргумента, используемые при интерполяции; Y(1), Y(2) — табличные значения функции, используемые при интерполяции.

Выходные: L — код ошибки, $L=\begin{cases} 0, \text{ если } x_{i_1} \neq x_{i_2}, \\ 1, \text{ если } x_{i_1}=x_{i_2}; \end{cases}$ Y0 — интерполяционное значение в точке $x$, вычисленное по формуле (2), если L=0, и равное $y_{i_1}$, если L=1.

Внутренние: нет.

*Используемые подпрограммы и функции*: нет.

*Требования к вызывающей программе*: в основной программе (или подпрограмме, обращающейся к данной подпрограмме) перед обращением к подпрограмме необходимо:

— описать массивы X и Y (оба длины не менее 2);

— в X0 занести значение $x$, при котором необходимо вычислить интерполяционное значение $y$;

— в X(1) и X(2) получить $x_{i_1}$ и $x_{i_2}$, определенные либо по правилу 1), либо по правилу 2);

— в Y(1) и Y(2) занести значения $y_{i_1}$ и $y_{i_2}$, соответствующие $x_{i_1}$ и $x_{i_2}$.

Замечание 1. Два последних требования желательно удовлетворить, составив отдельную подпрограмму для определения значений $x_{i_1}$ и $x_{i_2}$ и соответствующих им $y_{i_1}$ и $y_{i_2}$. Такая подпрограмма представляет самостоятельный интерес.

Замечание 2. В случае, если после работы подпрограммы L=1, т. е. $x_{i_1}=x_{i_2}$, проверьте правильность заполнения исходной таблицы.

*Возможные изменения подпрограммы*: при перенумерации строк подпрограммы необходимо соответственно изменить номера строк в операторах перехода в строках 1008 и 1009.

*1.2. Квадратичное интерполирование.* Пусть задана таблица $(x_1, y_1)$, $(x_2, y_2)$, . . ., $(x_n, y_n)$, в которой все $x_i$ различны. Тогда приближенное значение функции $f$ в точке $x$ ($y \approx f(x)$) вычисляется по формуле

$$y = y_{i_1}\frac{(x-x_{i_2})(x-x_{i_3})}{(x_{i_1}-x_{i_2})(x_{i_1}-x_{i_3})} + y_{i_2}\frac{(x-x_{i_1})(x-x_{i_3})}{(x_{i_2}-x_{i_1})(x_{i_2}-x_{i_3})} +$$
$$+ y_{i_3}\frac{(x-x_{i_1})(x-x_{i_2})}{(x_{i_3}-x_{i_1})(x_{i_3}-x_{i_2})} = y_{i_1} + (y_{i_1}-y_{i_2})\frac{x-x_{i_1}}{x_{i_1}-x_{i_2}} +$$
$$+ (y_{i_1}-y_{i_2})\frac{(x-x_{i_2})(x-x_{i_1})}{(x_{i_1}-x_{i_2})(x_{i_1}-x_{i_3})} - (y_{i_2}-y_{i_3})\frac{(x-x_{i_2})(x-x_{i_1})}{(x_{i_2}-x_{i_3})(x_{i_1}-x_{i_3})}, \quad (3)$$

где $x_{i_1}$, $x_{i_2}$ и $x_{i_3}$ — узлы, определяемые либо по правилу 3), либо по правилу 4).

Правило 3): $x_{i_1}$, $x_{i_2}$ и $x_{i_3}$ — ближайшие к $x$ узлы из набора $x_1, x_2, \ldots, x_n$.

Правило 4): пусть $x_{\min_1}$ — наименьшее из значений $x_1, x_2, \ldots, x_n$; пусть $x_{\min_2}$ — следующее по величине за $x_{\min_1}$ значение из набора $x_1, x_2, \ldots, x_n$; пусть $x_{\max_1}$ — наибольшее из значений $x_1, x_2, \ldots, x_n$; пусть $x_{\max_2}$ — предыдущее по величине перед $x_{\max_1}$ значение из набора $x_1, x_2, \ldots, x_n$; тогда

а) если $x \leqslant \frac{x_{\min_2}+x_{\min_1}}{2}$ или $x \geqslant \frac{x_{\max_2}+x_{\max_1}}{2}$, то $x_{i_1}$, $x_{i_2}$ и $x_{i_3}$ — ближайшие к $x$ узлы из набора $x_1, x_2, \ldots, x_n$;

б) если $\frac{x_{\min_2}+x_{\min_1}}{2} < x < \frac{x_{\max_2}+x_{\max_1}}{2}$, то $x_{i_1}$, $x_{i_2}$ и $x_{i_3}$ — ближайшие к $x$ узлы такие, что выполнено $\frac{x_{i_1}+x_{i_2}}{2} \leqslant x < \frac{x_{i_2}+x_{i_3}}{2}$.

Подпрограмма квадратичного интерполирования составляется аналогично предыдущей подпрограмме. Необходимо только внести следующие изменения:

— изменить комментарии в строках 1000 и 1002 с учетом изменения названия подпрограммы и с учетом того, что входными переменными теперь будут X0, X(1), X(2), X(3), Y(1), Y(2), Y(3);

— строки 1007, 1008, 1009, 1011 и 1012 заменить строками

```
1007 L=0 \ IF X(1)=X(2) GO TO 1009
1008 IF X(1)<>X(3) THEN IF X(2)<>X(3) GO TO 1010
1009 L=1 \ Y0=Y(1) \ GO TO 1014
1011 Y0=-(Y(2)-Y(3))*(X0-X(2))*(X0-X(1))/((X(2)-X(3))*(X(1)-X(3)))
1012 Y0=Y0+(Y(1)-Y(2))*(X0-X(2))*(X0-X(1))/((X(1)-X(2))*(X(1)-X(3)))
1013 Y0=Y0+Y(1)+(Y(1)-Y(2))*(X0-X(1))/(X(1)-X(2))
1014 RETURN
```

Соответствующим образом изменится инструкция к подпрограмме.

*2. Решение алгебраических и трансцендентных уравнений.* Решение конкретных задач довольно часто сводится к нахождению корней уравнения вида

$$f(x)=0, \tag{4}$$

где функция $f(x)$ определена и непрерывна на некотором интервале.

Если функция $f(x)$ представляет собой многочлен, то уравнение (4) называется алгебраическим; если же в функцию $f(x)$ входят трансцендентные (тригонометрические, логарифмические, показательные и т. п.) функции, то уравнение (4) называется трансцендентным.

Решение уравнения (4) разбивается на два этапа:

1) отделение корней, т. е. отыскание достаточно малых областей, в каждой из которых заключен один и только один корень уравнения;

2) вычисление выделенного корня с заданной точностью.

Проблема отделения корней представляет собой более сложную задачу, чем вычисление уже выделенного корня, и здесь рассматриваться не будет. Отметим только, что при отделении корней большую помощь может оказать приближенное построение графика функции с сопутствующим анализом на монотонность, смену знака, выпуклость функции и т. д.

Для вычисления выделенного (изолированного) корня существует множество методов, наиболее употребимыми из которых являются *метод половинного деления, метод итераций и метод Ньютона.*

Метод половинного деления подробно рассмотрен в работе 5. Рассмотрим здесь метод итераций и метод Ньютона. При изложении этих методов будем считать, что нам известен отрезок $[a, b]$, внутри которого существует один, и только один, из корней уравнения (4).

2.1. *Метод итераций.* Уравнение (4) представим в виде

$$x = \varphi(x), \tag{5}$$

что всегда можно сделать, и притом многими способами. Например, можно умножить левую и правую части уравнения (4) на произвольную константу $\lambda \neq 0$ и затем прибавить $x$ к левой и правой частям, т. е. представить уравнение (4) в виде

$$x = x + \lambda f(x). \tag{6}$$

Выберем теперь на отрезке $[a, b]$ произвольную точку $x_0$ и последовательно будем вычислять

$$\begin{aligned} x_1 &= \varphi(x_0), \\ x_2 &= \varphi(x_1), \\ &\cdots\cdots \\ x_k &= \varphi(x_{k-1}), \\ &\cdots\cdots \end{aligned} \tag{7}$$

Процесс последовательного вычисления значений $x_k$ ($k = 1, 2, \ldots$) по формулам (7) называется *итерационным процессом* (процессом последовательных приближений).

Если на отрезке $[a, b]$ выполнено условие

$$|\varphi'(x)| \leqslant q < 1, \tag{8}$$

то итерационный процесс (т. е. последовательность $x_0, x_1, x_2, \ldots$) сходится к корню уравнения (5). При этом, если необходимо вычислить корень уравнения (5) с точностью $\varepsilon$, то процесс итераций следует продолжать до тех пор, пока для двух последовательных

приближений $x_n$ и $x_{n-1}$ не будет выполнено

$$|x_n - x_{n-1}| \leqslant \varepsilon_1 = \frac{1-q}{q}\varepsilon, \qquad (9)$$

при этом всегда будет выполнено

$$|x^* - x_n| \leqslant \varepsilon,$$

где $\varepsilon$ — заданная предельная абсолютная погрешность корня $x^*$.

Если $q \leqslant 0{,}5$, то вместо (9) можно пользоваться соотношением

$$|x_n - x_{n-1}| \leqslant \varepsilon_1 = \varepsilon. \qquad (10)$$

Замечание. При использовании метода итераций необходимо стремиться представить уравнение (4) в виде (5) так, чтобы условие (8) выполнялось с наименьшим значением $q$. Часто этого можно добиться, используя представление (6) с удачно подобранным коэффициентом $\lambda$.

Далее приводится подпрограмма, реализующая метод итераций (программа 4.37).

**Программа 4.37**

```
1000 REM ПОДПРОГРАММА "МЕТОД ИТЕРАЦИЙ"
1001 REM
1002 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: X0,E
1003 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: X1
1004 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: Y,D
1005 REM ИСПОЛЬЗУЕТСЯ ФУНКЦИЯ Z(X)
1006 REM
1007 X1=X0
1008 Y=FNZ(X1)
1009 D=ABS(Y-X1) \ X1=Y
1010 IF D>E GO TO 1008
1011 RETURN
```

Инструкция к подпрограмме «Метод итераций».

*Назначение*: решение уравнения вида $x=\varphi(x)$ методом итераций с точностью $\varepsilon$ при начальном приближении $x_0$.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: **X0** — начальное приближение $x_0$; E — погрешность $\varepsilon_1$ (см. описание метода).

Выходные: X1 — вычисленное значение корня.

Внутренние: Y — очередное вычисляемое приближение; D — абсолютная величина разности двух последовательных приближений.

*Используемые подпрограммы и функции*: функция Z(X)=$\varphi(x)$.

*Требования к вызывающей программе*: в вызывающей программе необходимо:

— до обращения к данной подпрограмме описать функцию $Z(X)=\varphi(x)$;

— перед обращением к данной подпрограмме в X0 занести начальное приближение $x_0$, в E занести значение $\varepsilon_1$;

— не использовать Y и D для обозначения переменных, которые не должны быть испорчены в процессе работы данной подпрограммы.

З а м е ч а н и е. Подпрограмма гарантированно работает лишь при выполнении условия сходимости метода (см. описание работы). В противном случае подпрограмма может «зациклиться».

*Возможные изменения подпрограммы*: при перенумерации строк подпрограммы необходимо соответственно изменить номер строки в операторе перехода в строке 1010.

*2.2. Метод Ньютона.* Пусть задано уравнение (4), причем $f'(x)$ и $f''(x)$ определены, непрерывны и сохраняют постоянные знаки на отрезке $[a, b]$. Тогда, выбрав на отрезке $[a, b]$ некоторую точку $x_0$, последовательно вычисляем

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)}, \\ x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)}, \\ &\cdots\cdots\cdots \\ x_k &= x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}, \\ &\cdots\cdots\cdots \end{aligned} \tag{11}$$

Процесс последовательного вычисления значений $x_k$ ($k=1, 2, \ldots$) по формулам (11) называется процессом последовательных приближений по методу Ньютона.

Если начальное приближение $x_0$ выбрано так, что выполнено условие

$$f(x_0)\cdot f''(x_0) > 0, \tag{12}$$

то сходимость метода Ньютона (т. е. сходимость последовательности $x_0, x_1, x_2, \ldots$) к корню уравнения гарантирована. Если корень уравнения необходимо вычислить с точностью $\varepsilon$, то процесс вычисления приближений следует прекратить, когда для двух последовательных приближений $x_n$ и $x_{n-1}$ будет выполняться условие

$$|x_n - x_{n-1}| \leqslant \varepsilon_1 = \sqrt{\frac{2m_1\varepsilon}{M_2}}, \tag{13}$$

где $m_1$ — наименьшее значение $|f'(x)|$ на отрезке $[a, b]$; $M_2$ — наибольшее значение $|f''(x)|$ на отрезке $[a, b]$. При этом выполняется

$$|x^* - x_n| \leqslant \varepsilon,$$

где $\varepsilon$ — заданная предельная абсолютная погрешность корня $x^*$.

Если $\frac{2m_1}{M_2} \geqslant 10^{-2}$, то вместо (13) можно пользоваться соотношением

$$|x_n - x_{n-1}| \leqslant \varepsilon_1 = 10^{-1}\sqrt{\varepsilon}. \qquad (14)$$

Подпрограмма, реализующая метод Ньютона, составляется аналогично подпрограмме «Метод итераций». При этом, если в вызывающей программе описать функцию $Z(X)=x-f(x)/f'(x)$, то подпрограмма «Метод Ньютона» полностью совпадет с подпрограммой «Метод итераций», за исключением комментария, задающего название подпрограммы.

Однако подпрограмму «Метод Ньютона» имеет смысл составить с использованием функций $Z(X)=f(x)$ и $P(X)=f'(x)$. Это имеет следующие преимущества. Во-первых, выражение $f(x)/f'(x)$ для достаточно сложной $f(x)$ может оказаться слишком длинным и не уместиться в одной строке. В этом случае основная программа и подпрограмма несколько усложняются. Использование же двух функций существенно расширяет возможности записи сложных выражений. Во-вторых, подпрограмму «Метод Ньютона» можно дополнить анализом на равенство или близость к нулю значений $f'(x)$ — знаменателя выражения $f(x)/f'(x)$. При этом требуется отдельно вычислять значения $f'(x)$.

По сравнению с подпрограммой «Метод итераций» в подпрограмме «Метод Ньютона» с использованием двух функций, кроме соответствующего изменения комментария в строке 1000, необходимо строку 1008 заменить строкой

```
1008 Y=X1-FNZ(X1)/FNP(X1)
```

Соответствующим образом изменится инструкция к подпрограмме.

*3. Вычисление определенных интегралов.*

В научно-технических задачах нередко возникает необходимость вычисления определенного интеграла вида

$$\int_a^b f(x)\,dx. \qquad (15)$$

При этом подынтегральная функция может быть задана как аналитически, так и таблично. Для приближенного вычисления интеграла (15) существует много численных методов, из которых рассмотрим два: *метод трапеций* и *метод Симпсона.*

*3.1. Метод трапеций* (для аналитически заданной подынтегральной функции). Согласно методу трапеций отрезок интегрирования $[a, b]$ разбивается на $n$ равных частей длины $h=(b-a)/n$. Обозначим точки разбиения $x_0=a$, $x_1=x_0+h$, . . ., $x_i=x_0+ih$, . . .

..., $x_n=b$. Значения функции $f$ в точках $x_i$ обозначим $y_i$, т. е. $y_i=f(x_i)$. Тогда согласно методу трапеций

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2n}\left(y_0+y_n+2\sum_{i=1}^{n-1} y_i\right). \qquad (16)$$

Далее приводится подпрограмма, реализующая метод трапеций (программа 4.38).

**Программа 4.38**

```
1000 REM ПОДПРОГРАММА "МЕТОД ТРАПЕЦИЙ"
1001 REM
1002 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: A,B,N
1003 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: S
1004 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: I,H,X1
1005 REM ИСПОЛЬЗУЕТСЯ ФУНКЦИЯ Y(X)
1006 REM
1007 H=(B-A)/N \ S=0 \ X1=A
1008 FOR I=1 TO N-1
1009 X1=X1+H \ S=S+FNY(X1)
1010 NEXT I
1011 S=H*(FNY(A)+FNY(B)+2*S)/2
1012 RETURN
```

Инструкция к подпрограмме «Метод трапеций».

*Назначение*: вычисление интеграла вида $\int_a^b f(x)dx$ с аналитически заданной подынтегральной функцией методом трапеций.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: A, B — концы отрезка $[a, b]$; N — число частей разбиения отрезка $[a, b]$.

Выходные: S — вычисленное значение интеграла.

Внутренние: I — управляющая переменная цикла; H — шаг; X1 — текущее значение переменной интегрирования.

*Используемые подпрограммы и функции*: функция $Y(X)=f(x)$.

*Требования к вызывающей программе*: в вызывающей программе необходимо:

— до обращения к данной подпрограмме описать функцию $Y(X)=f(x)$;

— перед обращением к данной подпрограмме в A и B занести значения концов отрезка $[a, b]$, в N занести число частей разбиения отрезка $[a, b]$;

— не использовать I, H и X1 для обозначения переменных, которые не должны быть испорчены в процессе работы данной подпрограммы.

*Возможные изменения подпрограммы*: нет.

*3.2. Метод Симпсона* (для аналитически заданной подынтегральной функции). Разобьем отрезок интегрирования $[a, b]$ на $2n$ равных частей длины $h=(b-a)/2n$. Обозначим точки разбиения $x_0=a$, $x_1=x_0+h$, . . ., $x_i=x_0+ih$, . . ., $x_{2n}=b$. Значения функции $f$ в точках $x_i$ обозначим $y_i$, т. е. $y_i=f(x_i)$. Тогда согласно методу Симпсона

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6n}(y_0+4y_1+2y_2+\ldots+4y_{2n-1}+y_{2n}) =$$

$$=\frac{b-a}{6n}\left(y_0+y_{2n}+\sum_{i=1}^{2n-1}(3+(-1)^{i-1})\,y_i\right),$$

или, обозначив $N=2n$, имеем

$$\int_a^b f(x)\,dx \approx \frac{b-a}{3N}\left(y_0+y_N+\sum_{i=1}^{N-1}(3+(-1)^{i-1})\,y_i\right). \qquad (17)$$

Подпрограмма, реализующая метод Симпсона, составляется аналогично подпрограмме «Метод трапеций» со следующими изменениями. Во-первых, введем внутреннюю переменную C, которая будет равна 1 при нечетных $i$ и —1 при четных $i$. Так мы, экономя вычисления, реализуем коэффициент $(-1)^{i-1}$ в формуле (17). Далее необходимо соответствующим образом изменить комментарии в строках 1000 и 1004 и строки 1007, 1009 и 1011 заменить следующими:

```
1007 H=(B-A)/N \ S=0 \ X1=A \ C=1
1009 X1=X1+H \ S=S+(3+C)*FNY(X1) \ C=-C
1011 S=H*(FNY(A)+FNY(B)+S)/3
```

Соответственно несколько изменится инструкция к подпрограмме.

*4. Решение систем линейных уравнений.*

Из всего множества методов решения систем линейных уравнений рассмотрим метод Гаусса и некоторые его модификации.

*4.1. Метод Гаусса.* Пусть дана система

$$\begin{aligned}
&a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n=b_1,\\
&a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n=b_2,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n=b_n,
\end{aligned} \qquad (18)$$

или, в матричной форме,

$$Ax=b, \qquad (19)$$

где

$$A=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, \quad x=\begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{pmatrix}, \quad b=\begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_n \end{pmatrix},$$

$A$ — матрица системы, $x$ — столбец неизвестных, $b$ — столбец свободных членов.

Суть метода Гаусса состоит в следующем. Система (18) путем последовательного исключения неизвестных приводится к системе с треугольной матрицей, из которой затем определяются значения неизвестных. Рассмотрим процесс исключения неизвестных. Пусть $a_{11}\neq 0$. Разделим первое уравнение на $a_{11}$. Затем вычтем из каждого $i$-го ($i\geqslant 2$) уравнения полученное после деления первое, умноженное на $a_{i1}$. Получим систему

$$\begin{aligned} &x_1+\tilde{a}_{12}x_2+\dots+\tilde{a}_{1n}x_n=\tilde{b}_1, \\ &0+a_{22}^{(1)}x_2+\dots+a_{2n}^{(1)}x_n=b_2^{(1)}, \\ &\dots\dots\dots\dots\dots\dots \\ &0+a_{n2}^{(1)}x_2+\dots+a_{nn}^{(1)}x_n=b_n^{(1)}, \end{aligned} \tag{20}$$

где

$$\begin{aligned} &\tilde{a}_{1j}=\frac{a_{ij}}{a_{11}}, \quad j=1, \dots, n, \quad \tilde{b}_1=\frac{b_1}{a_{11}}, \\ &a_{ij}^{(1)}=a_{ij}-a_{i1}\tilde{a}_{1j}, \quad b_i^{(1)}=b_i-a_{i1}\tilde{b}_1, \\ &i=2, \dots, n; \quad j=1, \dots, n. \end{aligned} \tag{21}$$

В результате этих преобразований неизвестное $x_1$ оказалось исключенным из всех уравнений, кроме первого. Рассмотрим теперь систему (20). Пусть $a_{22}^{(1)}\neq 0$. Разделим второе уравнение на $a_{22}^{(1)}$. Далее, из каждого $i$-го ($i\geqslant 3$) уравнения вычтем полученное второе, умноженное на $a_{i2}^{(1)}$. Получим систему, в которой неизвестное $x_2$ будет исключено из всех уравнений, кроме первого и второго. Продолжая аналогичным образом процесс, а итоге получим систему вида

$$\begin{aligned} &x_1+\tilde{a}_{12}x_2+\tilde{a}_{13}x_3+\dots+\tilde{a}_{1n}x_n=\tilde{b}_1, \\ &\quad x_2+\tilde{a}_{23}^{(1)}x_3+\dots+\tilde{a}_{2n}^{(1)}x_n=\tilde{b}_2^{(1)}, \\ &\quad\quad x_3+\dots+\tilde{a}_{3n}^{(2)}x_n=\tilde{b}_3^{(2)}, \\ &\quad\quad\quad\dots\dots\dots\dots\dots \\ &\quad\quad\quad\quad x_n=\tilde{b}_n^{(n-1)}. \end{aligned} \tag{22}$$

Коэффициенты системы (22) можно представить в виде треугольной матрицы с единичной главной диагональю. Из системы (22) легко определяются значения неизвестных. Из последнего уравнения сразу определяем $x_n$. Подставляя $x_n$ в предпоследнее уравнение,

находим $x_{n-1}$ и т. д. Общие формулы определения неизвестных имеют вид

$$x_i = \tilde{b}_i^{(i-1)} - \sum_{j=i+1}^{n} \tilde{a}_{ij}^{(i-1)} x_j, \quad i = n, n-1, \ldots, 1. \tag{23}$$

Приведение системы (18) к виду (22) называется *прямым ходом метода Гаусса.* При этом процесс исключения $k$-го неизвестного называется $k$-м *шагом прямого хода.* Элементы $a_{11}, a_{22}^{(1)}, \ldots, a_{nn}^{(n-1)}$ называются *ведущими.* Общие формулы пересчета коэффициентов системы на $k$-м шаге имеют вид

$$\tilde{a}_{kj}^{(k-1)} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad j = k, \ldots, n, \quad \tilde{b}_k^{(k-1)} = \frac{b_k^{(k-1)}}{a_{kk}^{(k-1)}},$$
$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot \tilde{a}_{kj}^{(k-1)}, \tag{24}$$
$$b_i^{(k)} = b_i^{(k-1)} - a_{ik}^{(k-1)} \cdot \tilde{b}_k^{(k-1)},$$
$$i = k+1, \ldots, n; \quad j = k, \ldots, n.$$

Определение значений неизвестных из системы (22) по формулам (23) называется *обратным ходом метода Гаусса.*

Если на каком-то $k$-м шаге на главной диагонали окажется нулевой элемент $a_{kk}^{(k-1)} = 0$, то среди элементов $a_{ik}^{(k-1)}$ $(i = k+1, \ldots, n)$ следует найти ненулевой элемент и перестановкой строк переместить его на главную диагональ, а затем продолжить вычисления. Если такого ненулевого элемента не найдется, то определитель системы равен нулю и система либо не имеет решений, либо решений бесконечно много.

*4.1.1. Вычисление определителя методом Гаусса.* Определитель матрицы $A$ будет равен произведению ведущих элементов с учетом того, что при перестановке строк определитель меняет знак. То есть

$$\det A = \pm \prod_{k=1}^{n} a_{kk}^{(k-1)}, \tag{25}$$

где знак «+» получится, если суммарная перестановка строк была четной, а «—», если суммарная перестановка строк была нечетной.

З а м е ч а н и е. В методе Гаусса происходит деление строк на соответствующие ведущие элементы. При этом, если некоторый ведущий элемент мал, то происходит деление на маленькое число и либо может произойти переполнение, либо могут возникнуть значительные ошибки округления при последующих вычитаниях. С целью устранения этих недостатков разработаны следующие модификации метода Гаусса.

*4.1.2. Метод главных элементов.* На каждом $k$-м шаге сначала ищем элемент, равный $\max\limits_{\substack{k \leqslant i \leqslant n \\ k \leqslant j \leqslant n}} |a_{ij}^{(k-1)}|$. Пусть это будет элемент

$a_{pq}^{(k-1)}$. Тогда меняем местами $k$-е и $p$-е уравнения и переобозначим неизвестные $x_k=x_q$ и $x_q=x_k$, т. е. переставим $k$-й и $q$-й столбцы матрицы $A$. Далее производим $k$-й шаг метода Гаусса. В методе главных элементов необходимо запоминать перестановки столбцов (т. е. переобозначения переменных).

*4.1.3. Метод Гаусса с выбором главного элемента в строке.* На каждом $k$-м шаге сначала выбираем элемент, равный $\max\limits_{k\leqslant j\leqslant n}|a_{kj}^{(k-1)}|$. Пусть это будет элемент $a_{kq}^{(k-1)}$. Переобозначим неизвестные $x_k=x_q$ и $x_q=x_k$, т. е. переставим $k$-й и $q$-й столбцы матрицы $A$. Далее производим $k$-й шаг метода Гаусса. Здесь также необходимо запоминать перестановки столбцов (т. е. переобозначения переменных).

*4.1.4. Метод Гаусса с выбором главного элемента в столбце.* На каждом $k$-м шаге сначала ищем элемент, равный $\max\limits_{k\leqslant i\leqslant n}|a_{ik}^{(k-1)}|$. Пусть это будет элемент $a_{pk}^{(k-1)}$. Меняем местами $k$-е и $p$-е уравнения и производим $k$-й шаг метода Гаусса.

Далее следует подпрограмма, реализующая метод Гаусса с выбором главного элемента в столбце (программа 4.39).

Инструкция к подпрограмме «Метод Гаусса с выбором главного элемента в столбце».

*Назначение*: решение системы $n$ линейных уравнений с $n$ неизвестными методом Гаусса с выбором главного элемента в столбце.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: A — двумерный массив, матрица коэффициентов системы; B — одномерный массив, столбец свободных членов; N — число уравнений (неизвестных) в системе — длина массива B, число строк (столбцов) в массиве A.

Выходные: X — одномерный массив длины N, решение системы; L — код ошибки, L=0 — система решена, L=1 — определитель системы равен нулю.

Внутренние: A1 — двумерный массив размером N×N — служит для промежуточной работы подпрограммы с коэффициентами системы; I, J, K — индексы массивов и управляющие переменные циклов; R служит для промежуточного хранения отдельных элементов массивов.

*Используемые подпрограммы и функции*: нет.

*Требования к вызывающей программе*: в вызывающей программе необходимо:

— до обращения к данной подпрограмме описать двумерные массивы A и A1 размером N×N и описать одномерные массивы B и X размером N (все описания даются без учета нулевых значений индексов);

**Программа 4.39**

```
1000 REM ПОДПРОГРАММА "МЕТОД ГАУССА С ВЫБОРОМ
1001 REM ГЛАВНОГО ЭЛЕМЕНТА В СТОЛБЦЕ"
1002 REM
1003 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: A(I,J),B(I),N
1004 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: X(I),L
1005 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: I,J,K,R,A1(I,J)
1006 REM
1007 FOR I=1 TO N
1008 FOR J=1 TO N
1009 A1(I,J)=A(I,J)
1010 NEXT J
1011 X(I)=B(I)
1012 NEXT I \ L=0
1013 REM ПРЯМОЙ ХОД
1014 FOR I=1 TO N
1015 REM ПОИСК ГЛАВНОГО ЭЛЕМЕНТА В I-М СТОЛБЦЕ
1016 K=I \ R=ABS(A1(I,I))
1017 FOR J=I+1 TO N
1018 IF ABS(A1(J,I))<=R GO TO 1020
1019 K=J \ R=ABS(A1(J,I))
1020 NEXT J
1021 IF R=0 GO TO 1038 \ IF K=I GO TO 1027
1022 REM ПЕРЕСТАНОВКА I-ГО И K-ГО УРАВНЕНИЙ
1023 R=X(K) \ X(K)=X(I) \ X(I)=R
1024 FOR J=I TO N
1025 R=A1(K,J) \ A1(K,J)=A1(I,J) \ A1(I,J)=R
1026 NEXT J
1027 REM ИСКЛЮЧЕНИЕ I-ГО НЕИЗВЕСТНОГО
1028 R=A1(I,I) \ X(I)=X(I)/R
1029 FOR J=I TO N
1030 A1(I,J)=A1(I,J)/R
1031 NEXT J
1032 FOR K=I+1 TO N
1033 R=A1(K,I) \ X(K)=X(K)-R*X(I)
1034 FOR J=I TO N
1035 A1(K,J)=A1(K,J)-R*A1(I,J)
1036 NEXT J
1037 NEXT K \ GO TO 1040
1038 REM ОПРЕДЕЛИТЕЛЬ СИСТЕМЫ РАВЕН НУЛЮ
1039 L=1 \ I=N+1
1040 NEXT I \ IF L=1 GO TO 1047
1041 REM ОБРАТНЫЙ ХОД
1042 FOR I=N-1 TO 1 STEP -1
1043 FOR J=I+1 TO N
1044 X(I)=X(I)-A1(I,J)*X(J)
1045 NEXT J
1046 NEXT I
1047 RETURN
```

— перед обращением к данной подпрограмме занести в A матрицу коэффициентов исходной системы (по строкам) и занести в B столбец свободных членов;

— не использовать I, J, K, R для обозначения переменных и A1 для обозначения массивов, которые не должны быть испорчены в процессе работы данной подпрограммы.

*Возможные изменения подпрограммы*: при перенумерации строк подпрограммы необходимо соответственно изменить номера строк в операторах перехода в строках 1018, 1021, 1037, 1040.

*4.2. Метод Гаусса — Жордана.* Отличие метода Гаусса — Жордана от метода Гаусса заключается в следующем. На каждом $k$-м шаге $k$-е неизвестное исключается не только из уравнений с номерами $i=k+1, \ldots, n$, но и из уравнений с номерами $i=1, \ldots, k-1$ (см. формулы (24)). Таким образом, в методе Гаусса — Жордана система (18) приводится к системе с единичной матрицей, и отпадает необходимость в обратном ходе.

Замечание. Метод Гаусса — Жордана имеет те же модификации, что и метод Гаусса (с выбором главного элемента, главного элемента по строке, главного элемента по столбцу), причем соответствующий элемент выбирается из той же части матрицы $A$, как и в методе Гаусса.

*5. Решение дифференциальных уравнений.*

Математическое описание многих явлений часто сводится к системе дифференциальных уравнений вида

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \ldots, y_n), \\ y_2' &= f_2(x, y_1, y_2, \ldots, y_n), \\ &\cdots\cdots\cdots\cdots \\ y_n' &= f_n(x, y_1, y_2, \ldots, y_n) \end{aligned} \tag{26}$$

с дополнительными (начальными) условиями

$$\begin{aligned} y_1(x_0) &= y_{1_0}, \\ y_2(x_0) &= y_{2_0}, \\ &\cdots \\ y_n(x_0) &= y_{n_0}, \end{aligned} \tag{27}$$

где $x$ — независимая переменная, $y_1(x)$, $y_2(x)$, $\ldots$, $y_n(x)$ — неизвестные (искомые) функции.

Наиболее простым методом численного решения системы (26) с начальными условиями (27) является *метод Эйлера*, который мы и рассмотрим. Пусть нам известно решение системы (26) в точке $x$, и необходимо получить решение в точке $x+h$, тогда согласно методу Эйлера

$$\begin{aligned} y_1(x+h) &\approx y_1(x) + hf_1(x, y_1(x), y_2(x), \ldots, y_n(x)), \\ y_2(x+h) &\approx y_2(x) + hf_2(x, y_1(x), y_2(x), \ldots, y_n(x)), \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ y_n(x+h) &\approx y_n(x) + hf_n(x, y_1(x), y_2(x), \ldots, y_n(x)). \end{aligned} \tag{28}$$

Итак, отправляясь от значения решения в точке $x_0$ (условия (27)), мы можем последовательно получить приближенные значения решения в точках $x_1=x_0+h_1$, $x_2=x_1+h_2$, $x_3=x_2+h_3$, . . .

Замечание. Дифференциальное уравнение $n$-го порядка вида

$$y^{(n)}=f(x,\ y,\ y',\ y'',\ \ldots,\ y^{(n-1)}) \tag{29}$$

с начальными условиями

$$y(x_0)=y_0,\quad y'(x_0)=y_0',\ \ldots,\ y^{(n-1)}(x_0)=y_0^{(n-1)} \tag{30}$$

сводится к системе (26) с условиями (27) заменой $y'$ на неизвестную функцию $p_1(x)$, $y''$ на $p_2(x)$, . . ., $y^{(n-1)}$ на $p_{n-1}(x)$. То есть получается система

$$\begin{aligned} &y'=p_1,\\ &p_1'=p_2,\\ &p_2'=p_3,\\ &\ldots\ldots\\ &p_{n-1}'=f(x,\ y,\ p_1,\ p_2,\ \ldots,\ p_{n-1}) \end{aligned} \tag{31}$$

с начальными условиями

$$y(x_0)=y_0,\quad p_1(x_0)=y_0',\quad p_2(x_0)=y_0'',\ \ldots,\ p_{n-1}(x_0)=y_0^{(n-1)}. \tag{32}$$

Далее следует подпрограмма, реализующая метод Эйлера (программа 4.40).

**Программа 4.40**

```
1000 REM ПОДПРОГРАММА "МЕТОД ЭЙЛЕРА"
1001 REM
1002 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: N,H,X,Y0(I)
1003 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: Y(I),Y0(I),Y1(I)
1004 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: I
1005 REM ИСПОЛЬЗУЕТСЯ ПОДПРОГРАММА ВЫЧИСЛЕНИЯ
1006 REM ЗНАЧЕНИЙ ПРАВЫХ ЧАСТЕЙ УРАВНЕНИЙ
1007 REM
1008 FOR I=1 TO N \ Y(I)=Y0(I) \ NEXT I
1009 GOSUB 2000
1010 FOR I=1 TO N
1011 Y0(I)=Y0(I)+H*Y1(I)
1012 NEXT I
1013 RETURN
```

Инструкция к подпрограмме «Метод Эйлера».

*Назначение*: решение системы $n$ дифференциальных уравнений первого порядка, разрешенных относительно производных, методом Эйлера.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: N — число уравнений в системе; H — шаг; X — значение независимой

переменной; Y0 — массив длины N — значения решения в точке X, при первом обращении к подпрограмме — начальные условия.

Выходные: Y0 — массив длины N — значения решения в точке X+H; Y — массив длины N — значения решения в точке X; Y1 — массив длины N — значения производной решения в точке X.

Внутренние: I — индекс массива, управляющая переменная цикла.

*Используемые подпрограммы и функции*: подпрограмма вычисления значений правых частей уравнений с входными переменными N, X, Y (массив длины N) и выходной переменной Y1 (массив длины N); перед обращением к подпрограмме вычисления значений правых частей уравнений в N заносится число уравнений, в X заносится текущее значение аргумента, в массив Y заносятся значения решения в точке X; после работы подпрограммы в массиве Y1 получаются значения правых частей уравнений в точке X; подпрограмма располагается в строках, начиная с 2000-й.

*Требования к вызывающей программе*: в вызывающей программе необходимо:

— до обращения к данной подпрограмме описать массивы Y, Y0 и Y1 длины N;

— перед обращением к данной подпрограмме в X занести текущее значение аргумента $x$, в H занести значение шага $h$, в N занести число уравнений, в массив Y0 занести значения решения в точке X (перед первым обращением — начальные условия);

— записать подпрограмму вычисления значений правых частей уравнений в строках, начиная с 2000-й;

— не использовать I для обозначения переменных, которые не должны быть испорчены в процессе работы данной подпрограммы.

Замечание. Подпрограмма работает без контроля точности, поэтому необходимую точность следует обеспечивать за счет выбора шага $h$.

*Возможные изменения подпрограммы*: если подпрограмма вычисления значений правых частей уравнений располагается с $k$-й строки ($k \neq 2000$), то в строке 1009 данной подпрограммы необходимо заменить 2000 на $k$.

*6. Обеспечение необходимой точности.*

Пусть имеется метод приближенного вычисления некоторой величины $A$ с использованием шага $h$ (например, вычисление интеграла, решение дифференциальных уравнений и т. д.). На практике часто применяется следующий прием вычисления величины $A$ с заданной точностью $\varepsilon$. Выбираются шаги $h_1$ и $h_2$ ($h_2 < h_1$), и с использованием выбранных шагов вычисляются приближения $A_{h_1}$ и $A_{h_2}$ к величине $A$. Затем по некоторому критерию близости сравниваются $A_{h_1}$ и $A_{h_2}$. Если величины $A_{h_1}$ и $A_{h_2}$ достаточно близки,

то величина $A_{h_2}$ принимается за приближенное значение $A$; если нет, то выбирается шаг $h_3$ ($h_3<h_2$), вычисляется величина $A_{h_3}$, сравнивается с $A_{h_2}$ и т. д.

На практике, как правило, шаги $h_i$ выбираются так, что $h_{i+1}=\frac{h_i}{2}$ ($i=1, 2, \ldots$) и величина $A_{h_{i+1}}$ принимается за приближенное значение $A$, если выполнено

$$|A_{h_{i+1}}-A_{h_i}|\leqslant\varepsilon \quad \text{при} \quad |A_{h_{i+1}}|\leqslant 1,$$

либо

$$\left|\frac{A_{h_{i+1}}-A_{h_i}}{A_{h_{i+1}}}\right|\leqslant\varepsilon \quad \text{при} \quad |A_{h_{i+1}}|>1. \tag{33}$$

Если ни одно из этих условий не выполнено, то процесс продолжается.

Рассмотрим вычисление определенного интеграла методом трапеций с заданной точностью с использованием описанного приема. Итак, пусть требуется вычислить $\int_a^b f(x)\,dx$ с точностью $\varepsilon$. Выберем начальный шаг $h_1=(b-a)/N$, где $N$ — начальное число разбиений отрезка $[a, b]$. Вычислим методом трапеций с шагом $h_1$ величину $J_1\approx\int_a^b f(x)\,dx$. Теперь возьмем шаг $h_2=h_1/2$ и вычислим методом трапеций с шагом $h_2$ величину $J_2\approx\int_a^b f(x)\,dx$. Сравним $J_1$ и $J_2$ по критерию (33), и если одно из условий выполнено, то $J_2$ примем за приближенное значение интеграла; если нет, то выберем $h_3=h_2/2$ и продолжим процесс.

Далее приводится подпрограмма, реализующая метод трапеций с обеспечением заданной точности (программа 4.41).

**Программа 4.41**

```
2000 REM ПОДПРОГРАММА "МЕТОД ТРАПЕЦИЙ С
2001 REM ОБЕСПЕЧЕНИЕМ НЕОБХОДИМОЙ ТОЧНОСТИ"
2002 REM
2003 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: A,B,N,E
2004 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: S
2005 REM ВНУТРЕННИЕ ПЕРЕМЕННЫЕ: J1,J2
2006 REM ИСПОЛЬЗУЕТСЯ П/П "МЕТОД ТРАПЕЦИЙ"
2007 REM
2008 GOSUB 1000 \ J1=S
2009 N=2*N \ GOSUB 1000 \ J2=S
2010 IF ABS(J2)>1 GO TO 2013
2011 IF ABS(J2-J1)<=E GO TO 2014
2012 J1=J2 \ GO TO 2009
2013 IF ABS((J2-J1)/J2)>E GO TO 2012
2014 RETURN
```

Инструкция к подпрограмме «Метод трапеций с обеспечением необходимой точности».

*Назначение*: вычисление интеграла вида $\int_a^b f(x)\,dx$ с аналитически заданной подынтегральной функцией методом трапеций с заданной точностью ε.

*Используемый метод*: см. описание работы.

*Переменные, используемые подпрограммой.* Входные: A, B — концы отрезка $[a, b]$; N — начальное число частей разбиения отрезка $[a, b]$; E — заданная погрешность ε.

Выходные: S — значение интеграла, вычисленное с заданной точностью.

Внутренние: J1 и J2 — два последовательных приближения к значению интеграла.

*Используемые подпрограммы и функции*: подпрограмма «Метод трапеций», расположенная с 1000-й строки, с входными переменными A, B, N и выходной переменной S.

*Требования к вызывающей программе*: в вызывающей программе необходимо:

— до обращения к данной подпрограмме описать функцию $Y(X)=f(x)$ (необходима для работы подпрограммы «Метод трапеций»);

— перед обращением к данной подпрограмме в A и B занести значения концов отрезка $[a, b]$, в N занести начальное число частей разбиения отрезка $[a, b]$, в E занести значение ε;

— не использовать J1 и J2, а также I, H, X1 (используются в подпрограмме «Метод трапеций») для обозначения переменных, которые не должны быть испорчены в процессе работы данной подпрограммы.

Замечание. Подпрограмма работает без ограничения на количество делений начального шага пополам.

*Возможные изменения подпрограммы*:

— при перенумерации строк данной подпрограммы необходимо соответственно изменить номера строк в операторах перехода в строках 2010, 2011, 2012, 2013;

— если подпрограмма «Метод трапеций» располагается с $k$-й строки ($k \neq 1000$), то в строках 2008 и 2009 данной подпрограммы необходимо заменить 1000 на $k$.

Пример 1. Методом итераций найти корень уравнения

$$\arcsin(2x+1)-x^2=0 \qquad (34)$$

на отрезке $[-0{,}5;\ 0]$ с абсолютной погрешностью $\varepsilon=10^{-4}$.

Уравнение (34) преобразуем к виду (5) следующим образом:

$$\arcsin(2x+1)=x^2,$$
$$\sin(\arcsin(2x+1))=\sin x^2,$$
$$2x+1=\sin x^2,$$
$$x=\underbrace{0{,}5(\sin x^2-1)}_{\varphi(x)}.$$

Теперь имеем $|\varphi'(x)|=|x\cos x^2|\leqslant 0{,}5$ для всех $-0{,}5\leqslant x\leqslant 0$ То есть метод итераций сходится при любом $-0{,}5\leqslant x_0\leqslant 0$, и при этом для определения конца вычислений можно пользоваться соотношением (10). В качестве начального приближения выберем, например, $x_0=-0{,}4$.

Далее приводятся основная программа решения задачи (программа 4.42) и результаты ее работы. Подпрограмма «Метод итераций», к которой происходит обращение в строке 120, приведена ранее.

*Список переменных, используемых в основной программе.* Исходные данные: X0 — начальное приближение; E — предельная абсолютная погрешность вычисления корня.

Результат: X1 — приближенное значение корня уравнения.

Вспомогательные переменные: Z — имя функции $Z(X)=\varphi(x)$; X — формальный аргумент функции Z.

**Программа 4.42**

```
10 REM ПРОГРАММА "РЕШЕНИЕ УРАВНЕНИЯ
20 REM МЕТОДОМ ИТЕРАЦИЙ"
30 REM
40 REM ИСХОДНЫЕ ДАННЫЕ: X0,E
50 REM РЕЗУЛЬТАТ: X1
60 REM ВСПОМОГАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ: Z,X
70 REM
80 DEF FNZ(X)=.5*(SIN(X*X)-1)
90 PRINT
100 PRINT TAB(5);"ВВЕДИТЕ X0,E"
110 INPUT X0,E
120 GOSUB 1000
130 PRINT
140 PRINT TAB(5);"КОРЕНЬ УРАВНЕНИЯ = ";X1
150 STOP

RUNNH

     ВВЕДИТЕ X0,E
? -0.4,0.0001

     КОРЕНЬ УРАВНЕНИЯ = -.414539

STOP AT LINE 150
```

Пример 2. Методом Ньютона найти корень уравнения

$$\sin x - x + 0{,}15 = 0$$

на отрезке [0,5; 1] с абсолютной погрешностью $\varepsilon = 10^{-4}$.

Имеем

$$f'(x) = \cos x - 1 < 0 \quad \text{при} \quad 0{,}5 \leqslant x \leqslant 1,$$
$$f''(x) = -\sin x < 0 \quad \text{при} \quad 0{,}5 \leqslant x \leqslant 1.$$

Выбираем начальное приближение $x_0$ из условия (12). Так как $f''(x) < 0$ на [0,5; 1], то $x_0$ необходимо выбрать так, чтобы $f(x_0) < 0$, этому условию удовлетворяет, например, $x_0 = 1$. Далее имеем $m_1 = \min\limits_{[0{,}5;\ 1]} |\cos x - 1| = |\cos 0{,}5 - 1| \approx |0{,}88 - 1| = 0{,}12$ и $M_2 = \max\limits_{[0{,}5;\ 1]} |-\sin x| = |-\sin 1| \approx 0{,}84$. Таким образом, имеем $2m_1/M_2 \approx 0{,}29 > 10^{-2}$, и мы можем воспользоваться условием (14).

Программа решения этой задачи составляется аналогично программе предыдущего примера.

Пример 3. На отрезке [0; 0,3] составить таблицу значений приближенного решения дифференциального уравнения

$$y'' = 3y' - 2y + 2x - 3, \tag{35}$$

удовлетворяющего начальным условиям $y(0) = 1$, $y'(0) = 2$, выбрав шаг интегрирования $h = 0{,}1$.

Приведем уравнение (35) к системе вида (26) следующим образом. Обозначим $y$ через $y_1$ и $y'$ через $y_2$. Тогда получим систему

$$y_1' = y_2,$$
$$y_2' = 3y_2 - 2y_1 + 2x - 3 \tag{36}$$

с начальными условиями $y_1(0) = 1$, $y_2(0) = 2$.

Далее приводится основная программа решения задачи (программа 4.43) с подпрограммой вычисления правых частей уравнений. Подпрограмма «Метод Эйлера», к которой происходит обращение в строке 170, приведена ранее.

*Список переменных, используемых в основной программе.* Исходные данные: N — число уравнений в системе; A, B — концы отрезка интегрирования; H — шаг интегрирования; Y0 — массив начальных условий.

Результаты: X — очередное значение аргумента; Y0 — массив вычисленных значений решения системы (36) при очередном значении аргумента.

Вспомогательные переменные: I — индекс — управляющая переменная цикла; Y, Y1 — массивы, необходимые для подпрограммы «Метод Эйлера».

Замечания к программе. 1. Программа составлена в достаточно общем виде и пригодна для решения систем, содержащих до 20 уравнений, при соответствующих изменениях печати и подпрограммы «Вычисление правых частей уравнений».

Программа 4.43

```
 10 REM ПРОГРАММА "РЕШЕНИЕ ДИФФЕРЕНЦИАЛЬНЫХ
 20 REM УРАВНЕНИЙ МЕТОДОМ ЭЙЛЕРА"
 30 REM
 40 REM ИСХОДНЫЕ ДАННЫЕ: N,A,B,H,Y0(I)
 50 REM РЕЗУЛЬТАТЫ: X,Y0(I)
 60 REM ВСПОМОГАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ: I,Y(I),Y1(I)
 70 REM
 80 DIM Y(20),Y0(20),Y1(20)
 90 PRINT TAB(2)!"ВВЕДИТЕ N,A,B,H"
100 INPUT N,A,B,H
110 FOR I=1 TO N
120 PRINT TAB(2)!"ВВЕДИТЕ Y0("!I!")"
130 INPUT Y0(I)
140 NEXT I \ PRINT
150 PRINT "X","Y(X)" \ X=A
160 PRINT X,Y0(1)
170 GOSUB 1000 \ X=X+H
180 PRINT X,Y0(1)
190 IF X<B GO TO 170
200 STOP
2000 REM ПОДПРОГРАММА "ВЫЧИСЛЕНИЕ ПРАВЫХ
2001 REM ЧАСТЕЙ УРАВНЕНИЙ"
2002 REM
2003 REM ВХОДНЫЕ ПЕРЕМЕННЫЕ: X,Y(I)
2004 REM ВЫХОДНЫЕ ПЕРЕМЕННЫЕ: Y1(I)
2005 REM
2006 Y1(1)=Y(2)
2007 Y1(2)=3*Y(2)-2*Y(1)+2*X-3
2008 RETURN

RUNNH

  ВВЕДИТЕ N,A,B,H
? 2,0,0.3,0.1
  ВВЕДИТЕ Y0( 1 )
? 1
  ВВЕДИТЕ Y0( 2 )
? 2

X             Y(X)
 0             1
 .1            1.2
 .2            1.41
 .3            1.631

STOP AT LINE 200
```

2. В программе один и тот же массив Y0 используется в качестве входного и выходного. Это обусловлено работой подпрограммы «Метод Эйлера» и является типичным приемом программирования. До работы подпрограммы «Метод Эйлера» в Y0 содержатся значения решения в точке $x$, после работы подпрограммы в Y0 получаются

значения решения в точке $x+h$, а значения решения в точке $x$ сохраняются в массиве Y, так как они в общем случае могут понадобиться для каких-либо сранений.

3. В программе не используются все возможности подпрограммы «Метод Эйлера», так как это не требуется по условию задачи. Тем не менее после работы подпрограммы «Метод Эйлера» в качестве результата, кроме значения $y$ в точке $x+h$, можно использовать еще и значения $y$ в точке $x$ (Y (1)), $y'$ в точке $x$ (Y (2) либо Y1 (1)), $y''$ в точке $x$ (Y1 (2)) и $y'$ в точке $x+h$ (Y0 (2)).

4. Подпрограмму «Вычисление правых частей уравнений» можно составлять различными способами. В частности, при решении систем вида (31) однотипные вычисления $y_k'=y_{k+1}$ можно организовать в цикле. При этом понадобится входной параметр N, который в нашем случае не используется и поэтому отсутствует.

Задание I уровня. Составить программу с использованием необходимых подпрограмм. В тех случаях, когда это нужно, проверить условие сходимости. В программе предусмотреть печать поясняющей текстовой информации.

*Варианты задач I уровня.*

1. Задана таблица (1,1611;1,4462), (2,611;10,70319), (2,0203; 5,68038), (1,0537; 1,13792), (2,074; 6,06063), (1,1074; 1,28657). Используя подпрограмму «Линейная интерполяция», вычислить интерполяционное значение при $x=1{,}2148$ и $x=2{,}3425$. При определении узлов, по которым необходимо проводить интерполирование, пользоваться правилом 1). Определение необходимых узлов оформить в виде подпрограммы. Выдать в качестве результатов вычисленные интерполяционные значения и номера узлов, по которым проводилась интерполяция.

2. Решить задачу 1, используя подпрограмму «Квадратичная интерполяция». Использовать правило 3).

3. Методом половинного деления найти корень уравнения $x+\sqrt{x}+\sqrt[3]{x}-2{,}5=0$ на отрезке [0,4;1] с точностью $10^{-4}$ и корень уравнения $\cos x-e^{-\frac{x^2}{2}}+x-1=0$ на отрезке [1; 2] с точностью $10^{-3}$. Использовать соответствующую подпрограмму.

4. С использованием подпрограммы «Метод итераций» найти корень уравнения $x-2+\sin\frac{1}{x}=0$ на отрезке [1,2;2] с точностью $10^{-4}$.

5. С использованием подпрограммы «Метод Ньютона» найти корень уравнения $e^x+\ln x-10x=0$ на отрезке [3;4] с точностью $10^{-4}$.

6. С использованием подпрограммы «Метод трапеций» вычислить $\int\limits_1^4 \frac{\ln^2 x}{x}\,dx$, разбивая отрезок интегрирования на 52 части.

7. С использованием подпрограммы «Метод Симпсона» вычислить $\int_0^2 \frac{e^{3x}+1}{e^x+1}\,dx$, разбивая отрезок интегрирования на 240 частей.

8. Вычислить интеграл из задачи 6 с использованием подпрограммы «Метод трапеций с обеспечением необходимой точности» с точностью $10^{-4}$.

9. Вычислить интеграл из задачи 7 с использованием подпрограммы «Метод Симпсона с обеспечением необходимой точности» с точностью $10^{-4}$.

10. С использованием подпрограммы «Метод Гаусса с выбором главного элемента в столбце» решить систему

$$\begin{aligned} x_1+2x_2+3x_3&=4,\\ 2x_1+3x_2+4x_3&=5,\\ x_1-x_2+x_3&=6. \end{aligned}$$

11. С использованием подпрограммы «Метод Гаусса с выбором главного элемента в строке» решить систему

$$\begin{aligned} 8x_1-x_2-2x_3 &=2{,}3,\\ 10x_2+x_3+2x_4&=-0{,}5,\\ -x_1+6x_3+2x_4&=-1{,}2,\\ 3x_1-x_2+2x_3+12x_4&=3{,}7. \end{aligned}$$

12. С использованием подпрограммы «Метод Эйлера» решить уравнение $y'=y+x$ при $y(0)=1$ на отрезке $[0;0{,}5]$ с шагом $h=0{,}05$.

13. С использованием подпрограммы «Метод Эйлера» решить уравнение $y''+4y'+4y=0$ при $y(0)=1$, $y'(0)=-1$ на отрезке $[0;0{,}5]$ с шагом $h=0{,}05$.

14. С использованием подпрограммы «Метод Эйлера» решить систему

$$\begin{aligned} y'&=y+z+x,\\ z'&=-4y-3z+2x \end{aligned}$$

при $y(0)=1$, $z(0)=0$ на отрезке $[0;0{,}3]$ с шагом $h=0{,}05$.

15. Решить задачу 1, используя подпрограмму «Линейная интерполяция». Использовать правило 2).

Указания к решению задач I уровня.

1. При определении необходимых узлов можно упорядочить по возрастанию таблицу значений узлов либо таблицу расстояний от исследуемой точки до узлов и затем определять необходимые узлы.

2. См. указания к задаче 1.

3. См. работу 5.

4—8. См. теоретическое введение.

9. Подпрограмма «Метод Симпсона» с обеспечением необходимой точности» полностью совпадает с аналогичной подпрограммой

для метода трапеций, за исключением обращения в ней соответственно к подпрограмме «Метод Симпсона».

10—14. См. теоретическое введение.

15. См. указания к задаче 1.

Задание II уровня. В задачах II уровня необходимо составить подпрограмму метода, модифицируя соответствующую подпрограмму из теоретического введения. К составленным подпрограммам написать инструкции. С использованием составленных подпрограмм решить указанные задачи.

*Варианты задач II уровня.*

1. Составить подпрограмму, реализующую метод итераций с ограничением на число проводимых итераций. Ограничение задавать во входной переменной. Предусмотреть в подпрограмме код ошибки, равный нулю, если решение с необходимой точностью найдено за заданное количество итераций, и единице в противном случае. Решить уравнение из задачи 4 уровня I.

2. Составить подпрограмму вычисления определителя с использованием метода Гаусса с выбором главного элемента в столбце. Вычислить определитель матрицы системы из задачи 11 уровня I.

3. Составить подпрограмму вычисления определителя с использованием метода Гаусса с выбором главного элемента в строке. Вычислить определитель матрицы системы из задачи 11 уровня I.

4. Составить подпрограмму вычисления определителя с использованием метода главных элементов. Вычислить определитель матрицы системы из задачи 10 уровня I.

5. Составить подпрограмму, реализующую квадратичную интерполяцию с проверкой всей таблицы на присутствие одинаковых узлов. Предусмотреть в подпрограмме код ошибки, равный нулю, если в таблице нет одинаковых узлов, и единице в противном случае. Решить задачу 1 уровня I, используя правило 4).

6. Составить подпрограмму, реализующую метод Гаусса — Жордана с выбором главного элемента в столбце. Предусмотреть в подпрограмме код ошибки, равный нулю, если система решена, и единице, если определитель системы равен нулю. Решить задачу 11 уровня I.

7. Составить подпрограмму, реализующую метод Ньютона с ограничением на число последовательных приближений. Ограничение задавать во входной переменной. Предусмотреть в подпрограмме код ошибки, равный нулю, если решение с необходимой точностью найдено за заданное количество приближений, и единице в противном случае. Решить уравнение из задачи 5 уровня I.

8. Составить подпрограмму, реализующую метод Ньютона с проверкой на допустимую близость к нулю значений производной. Допустимую меру близости к нулю задавать во входной переменной. Предусмотреть в подпрограмме код ошибки, равный нулю, если

значения производной не становятся слишком маленькими и решение найдено с необходимой точностью. Если значения производной становятся недопустимо близки к нулю, предусмотреть выход из подпрограммы со значением кода ошибки, равным единице. Решить уравнение из задачи 5 уровня I.

9. Составить подпрограмму, реализующую метод Ньютона с проверкой на равенство нулю значений производной. Предусмотреть в подпрограмме код ошибки, равный нулю, если решение найдено с необходимой точностью. Если значение производной становится равным нулю, предусмотреть выход из подпрограммы со значением кода ошибки, равным единице. Решить уравнение из задачи 5 уровня I.

10. Составить подпрограмму, реализующую метод трапеций с обеспечением необходимой точности при ограничении на количество делений начального шага пополам. Ограничение на количество делений шага пополам задавать во входной переменной. Предусмотреть в подпрограмме код ошибки, равный нулю, если необходимая точность достигнута при выполнении ограничения на количество делений начального шага пополам, и единице в противном случае. Вычислить интеграл из задачи 6 уровня I.

11. Для метода Симпсона решить задачу, аналогичную задаче 10. Вычислить интеграл из задачи 7 уровня I.

12. Составить подпрограмму, реализующую линейное интерполирование с проверкой всей таблицы на присутствие одинаковых узлов. Предусмотреть в подпрограмме код ошибки, равный нулю, если в таблице нет одинаковых узлов, и единице в противном случае. Решить задачу 1 уровня I, используя правило 2).

13. Составить подпрограмму, реализующую вычисление определителя с использованием метода Гаусса — Жордана с выбором главного элемента в столбце. Вычислить определитель матрицы системы из задачи 11 уровня I.

14. Составить подпрограмму вычисления определителя с использованием метода Гаусса — Жордана с выбором главного элемента в строке. Вычислить определитель матрицы системы из задачи 11 уровня I.

15. Составить подпрограмму вычисления определителя с использованием метода Гаусса — Жордана с выбором главного элемента в матрице. Вычислить определитель матрицы системы из задачи 10 уровня I.

Указания к решению задач II уровня.

1—2. См. теоретическое введение.

3—4. При вычислении определителя нет необходимости запоминать перестановки столбцов, надо только при каждой перестановке умножать определитель на —1.

5. См. указание к задаче 1 уровня I.

6—11. См. теоретическое введение.

12. См. указание к задаче 1 уровня I.

13. См. теоретическое введение.

14—15. См. указание к задачам 3—4.

Задание III уровня. Задачи III уровня требуют либо составления более сложных подпрограмм, либо комбинирования подпрограмм и предполагают элемент творчества. Для составленных подпрограмм написать инструкцию.

*Варианты задач III уровня.*

1. Составить подпрограмму, реализующую решение системы линейных уравнений и вычисление определителя матрицы системы методом Гаусса с выбором главного элемента в строке. Предусмотреть в подпрограмме код ошибки, равный нулю, если система решена, и единице в противном случае. Решить задачу 11 уровня I.

2. Составить подпрограмму, реализующую решение системы линейных уравнений и вычисление определителя методом главных элементов. Предусмотреть в подпрограмме код ошибки, равный нулю, если система решена, и единице в противном случае. Решить задачу 10 уровня I.

3. Используя подпрограмму «Метод трапеций с обеспечением необходимой точности», построить таблицу функции $F(x)=\int_1^x \frac{\ln^3 t}{t}\,dt$ на отрезке [1; 4] с шагом $h=0{,}5$. Используя подпрограмму «Линейная интерполяция», вычислить $F(2{,}761)$.

4. Решить задачу 3, используя подпрограмму «Квадратичная интерполяция».

5. Решить задачу 12 уровня I. Используя подпрограмму «Линейная интерполяция», вычислить $y(0{,}327)$.

6. Решить задачу 5, используя подпрограмму «Квадратичная интерполяция».

7. Составить подпрограмму, реализующую решение системы линейных уравнений и вычисление определителя методом Гаусса — Жордана с выбором главного элемента в матрице. Предусмотреть в подпрограмме код ошибки, равный нулю, если система решена, и единице в противном случае. Решить задачу 10 уровня I.

8. Решить задачу 13 уровня I. Используя подпрограмму «Линейная интерполяция», вычислить $y(0{,}176)$ и $y'(0{,}241)$.

9. Решить задачу 8, используя подпрограмму «Квадратичная интерполяция».

10. Решить задачу 14 уровня I. Используя подпрограмму «Линейная интерполяция», вычислить $y(0{,}11)$ и $z(0{,}22)$.

11. Решить задачу 10, используя подпрограмму «Квадратичная интерполяция».

12. Составить подпрограмму, реализующую решение системы

линейных уравнений и вычисление определителя методом Гаусса — Жордана с выбором главного элемента в строке. Предусмотреть в подпрограмме код ошибки, равный нулю, если система решена, и единице в противном случае. Решить задачу 11 уровня I.

13. Используя подпрограмму «Метод Симпсона с обеспечением необходимой точности», построить таблицу функции $F(x) = \int_0^x \frac{e^{3t}+1}{e^t+1}\,dt$ на отрезке [0;2] с шагом $h=0{,}25$. Используя подпрограмму «Линейная интерполяция», вычислить $F(0{,}737)$.

14. Решить задачу 13, используя подпрограмму «Квадратичная интерполяция».

15. Составить подпрограмму, реализующую метод Ньютона с ограничением на число последовательных приближений и проверкой на равенство нулю значений производной. Ограничение задавать во входной переменной. Предусмотреть в подпрограмме код ошибки, равный нулю, если решение с необходимой точностью найдено за заданное количество приближений, и единице в противном случае. Решить уравнение из задачи 5 уровня I.

Указания к решению задач III уровня.

1,2,7,12 и 15. См. теоретическое введение.

3,4,13 и 14. Для убыстрения вычислений воспользоваться тем, что $\int_a^b f(x)\,dx = \int_a^c f(x)\,dx + \int_c^b f(x)\,dx$. Таблицу соответствующей функции запомнить в массиве для последующей интерполяции. Интегрирование предусмотреть с точностью $10^{-3}$. См. указание к задаче 1 уровня I.

5,6,8—11. Таблицу значений решения запомнить в массиве для последующей интерполяции. См. указание к задаче 1 уровня 1. В задачах 8 и 9 см. замечания к программе «Решение дифференциальных уравнений методом Эйлера».

Вопросы для самопроверки.

1. В чем состоит задача интерполяции? Многочлен Лагранжа.
2. Что такое линейное интерполирование? Проверьте равенство в формуле (2).
3. Что такое квадратичное интерполирование? Проверьте равенство в формуле (3).
4. Изобразите геометрически узлы интерполяции, выбранные по правилам 1), 2), 3), 4).
5. Метод половинного деления. Сходимость метода.
6. Метод итераций. Условие сходимости. Условие прекращения процесса.
7. Метод Ньютона. Условие сходимости. Условие окончания вычислений.
8. Сравнение методов решения уравнений.

9. Метод трапеций. Как организовать вычисление интеграла с заданной точностью?
10. Метод Симпсона.
11. Метод Гаусса. Недостатки метода.
12. Методы главных элементов. Сравнение методов.
13. Вычисление определителя.
14. Метод Гаусса — Жордана.
15. Метод Эйлера. Как организовать решение дифференциального уравнения с заданной точностью?

## *Работа 7.* ВЫВОД РЕЗУЛЬТАТОВ В ВИДЕ ГРАФИКОВ *), ТАБЛИЦ И ГИСТОГРАММ. ПОСТРОЕНИЕ ГЕОМЕТРИЧЕСКИХ ФИГУР

Теоретическое введение. При решении задач на ЭВМ важное значение имеет наглядность и удобство дальнейшего использования выводимых результатов. Основным средством наглядного представления информации в языке бейсик является стандартная функция TAB (см. п. 5.6 главы 3), позволяющая строить графики, выводить данные в виде таблиц, получать различные геометрические фигуры. Одним из способов наглядного представления результатов является построение гистограмм, когда для каждого данного на экране строится линия, длина которой пропорциональна числовому значению данного. Ниже рассмотрена реализация на языке бейсик следующих способов вывода результатов:

— в виде графиков;
— в виде таблиц;
— в виде гистограмм,

а также методы построения геометрических фигур.

*1. Построение графиков с использованием функции* TAB. Для построения графика функции $y=f(x)$ в заданном интервале изменения $x$ с заданным шагом для каждой новой точки будем печатать в новой строке какой-нибудь символ, например *, в позиции, определяемой вычисленным значением функции $f(x)$, используя для этого функцию TAB.

Рассмотрим программу 4.44 (здесь и далее справа — результат выполнения программы).

Когда программа выполняется, то при X=0 * появляется в нулевой позиции первой строки, при X=1 (при втором выполнении оператора PRINT) * появляется в первой позиции следующей строки, при X=2 * появляется во второй позиции третьей строки и т. д. Строки отстоят друг от друга на равном расстоянии,

*) Средства для работы с графопостроителем не рассматриваются.

**Программа 4.44**

```
10 FOR X=0 TO 15
20 PRINT TAB(X);"*"
30 NEXT X
40 STOP
```

```
RUNNH
*
 *
  *
   *
    *
     *
      *
       *
        *
         *
          *
           *
            *
             *
STOP AT LINE 40
```

и этому расстоянию может быть поставлен в соответствие постоянный шаг изменения X, равный в рассматриваемой программе 1. Таким образом, совокупность звездочек, появившаяся на экране в результате выполнения программы 4.44, может рассматриваться как график функции $Y=X$ в интервале изменения X от 0 до 15. При этом вертикальная ось экрана является осью абсцисс (осью $x$), а горизонтальная — осью ординат (осью $y$), т. е. график оказывается повернутым относительно привычного представления на 90°.

Рассмотрим еще одну программу (см. программу 4.45).

**Программа 4.45**

```
10 X=-5
20 PRINT TAB(X*X);"*"
30 X=X+1
40 IF X<=5 GO TO 20
50 STOP
```

```
RUNNH
*
 *
  *
   *
    *
     *
      *
       *
        *
         *
          *
STOP AT LINE 50
```

При каждом выполнении оператора PRINT на следующей строке экрана появляется * в позиции, определяемой значением X*X, где X меняется от —5 до 5 с шагом 1. Если считать вертикальную ось экрана осью $x$, а горизонтальную — осью $y$, то после

выполнения этой программы на экране будет вычерчен график функции $y=x^2$, содержащий 11 точек, обозначенных звездочками.

Приведенную программу можно модифицировать таким образом, чтобы рядом с * на экране появлялось значение функции (см. программу 4.45′).

**Программа 4.45′**

```
10 X=-5                          RUNNH
20 H=X*X
30 PRINT TAB(H);"*";H                                    * 25
40 X=X+1                                        * 16
50 IF X<=5 GO TO 20                        * 9
60 STOP                               * 4
                                    * 1
                                  * 0
                                    * 1
                                       * 4
                                            * 9
                                                 * 16
                                                      * 25

                                 STOP AT LINE 60
```

Значение X*X в этой программе вычисляется один раз (оператор 20) и присваивается переменной H, которая используется далее в операторе 30.

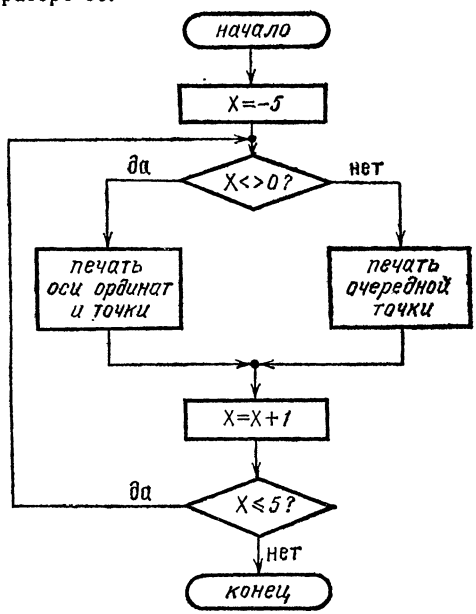



Рис. 4.12

При желании можно точки графика печатать при помощи значений функции в каждой точке. Для этого нужно заменить оператор 30 следующим образом:

```
30 PRINT TAB(H); H
```

Модифицируем программу так, чтобы на экране вычерчивались также оси абсцисс и ординат. Для этого добавим еще один член в список вывода оператора PRINT, с тем чтобы на каждой строке в первой позиции печатался символ «!» (из этих символов и будет состоять ось абсцисс). Ось ординат начертим символами — «—», которые должны для этого быть выведены во всю строку при $x=0$ (рис. 4.12 и программа 4.46).

**Программа 4.46**

```
10 X=-5
20 IF X<>0 GO TO 90
30 PRINT "*";
40 FOR I=1 TO 30
50 PRINT "-";
60 NEXT I
70 PRINT
80 GO TO 100
90 PRINT "!";TAB(X*X);"*"
100 X=X+1
110 IF X<=5 GO TO 20
120 STOP
```

```
RUNNH
!                              *
!                    *
!          *
!   *
!*
*-------------------------------
!*
!   *
!
!                    *
!                              *

STOP AT LINE 120
```

Операторы 40—60 выводят при $x=0$ в одну строку символы «—», образующие ось ординат.

При составлении программы для получения графика функции нужно иметь в виду, что аргумент функции TAB может изменяться в диапазоне от 0 до 71, и, следовательно, если значения функции $y=f(x)$ лежат вне этого диапазона или изменяются незначительно для отдельных точек графика, то следует вводить *масштаб* и *сдвиг*. Масштаб выбирается, исходя из тех соображений, чтобы минимальное и максимальное значения функции изображались точками, отстоящими друг от друга не более чем на ширину экрана. Сдвиг выбирается так, чтобы минимальное значение выводилось не менее чем в 0-ю позицию.

Если $y_{max}$ и $y_{min}$ — максимальное и минимальное значения функции $y=f(x)$ при изменении $x$ в заданных пределах, то масштаб $M$ и сдвиг $S$ можно определить, используя соотношения

$$My_{max}+S=A_{max},$$
$$My_{min}+S=A_{min},$$

где $A_{max}$ и $A_{min}$ — максимальное и минимальное значения аргумента функции TAB (выбираются в пределах от 0 до 71).

Аргумент функции TAB задается далее выражением $S+Mf(x)$.

Пример. Построить график функции $y=\sin x$ в интервале от 0 до $2\pi$ с шагом $\pi/8$ с вычерчиванием оси абсцисс.

Значения функции $\sin x$ изменяются в пределах от —1 до 1, и значения масштаба и сдвига можно, следовательно, определить из соотношений

$$M+S=50,$$
$$-M+S=0$$

(график будет располагаться между 0-й и 50-й позициями экрана). Отсюда $M=25$, $S=25$.

При вычерчивании оси абсцисс нужно иметь в виду, что в первом полупериоде символ, изображающий ось абсцисс (!), должен печататься до символа, изображающего функцию, во втором — наоборот. Если же функция равна 0, т. е. пересекает в этой точке ось абсцисс, то в эту позицию должен выводиться только один символ (*). Поэтому в программе должны быть три различных оператора PRINT (см. программу 4.47).

Пояснения к программе. В программе использованы вспомогательные переменные P и H, чтобы не вычислять многократно одни и те же выражения.

Проверка P на совпадение с нулем в строке 50 осуществляется с точностью $2\cdot10^{-3}$. Точное равенство нулю значения $\sin x$ практически никогда не достигается, так как вычисление $\sin x$ в ЭВМ осуществляется приближенно суммированием ряда.

Для улучшения восприятия на осях ординат и абсцисс можно указать также некоторые точки, определяющие выбранный масштаб построения, а также напечатать обозначение осей. Программа 4.48 выполняет построение графика функции $\cos x$ с обозначением на оси ординат точек —1, 0, 1 и обозначением осей X и Y.

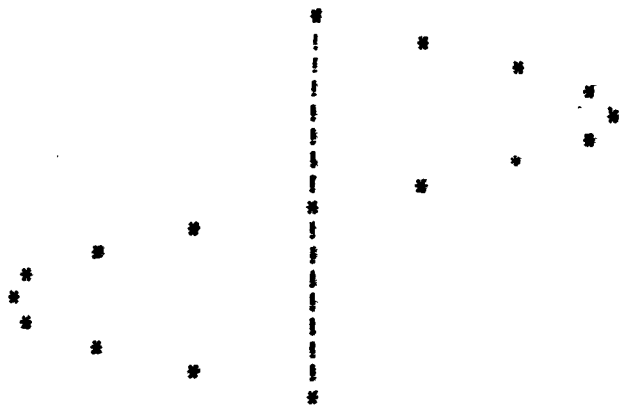
График располагается между 10-й и 60-й позициями экрана. Для этого используется масштаб $M=25$, сдвиг $S=35$. Разметка оси ординат осуществляется оператором PRINT (строка 10). Помимо этого потребовалось прерывание горизонтальной линии, обозначающей ось ординат, символом «.», выведенным в 10-ю позицию (при X=—1) оператором PRINT в строке 30, символом «!», выведенным в 30-ю позицию (при X=0) оператором PRINT в строке 50, и точкой графика *, выведенной в 60-ю позицию (при X=1) оператором PRINT в строке 70.

*2. Построение графиков с использованием массива.* Другой способ вывода результатов в виде графика — создание его копии в символьном массиве и вывод, далее, этого массива на экран или на печатающее устройство. Символьный массив при этом рассматривается как плоскость: координата по оси абсцисс — номер столбца, координата по оси ординат — номер строки.

**Программа 4.47**

```
10 I=0
20 X=0
30 H=PI/8
40 P=25*SIN(X)
50 IF ABS(P)<=2.00000E-03 GO TO 90
60 IF P>0 GO TO 110
70 PRINT TAB(P+25);"*";TAB(25);"!"
80 GO TO 120
90 PRINT TAB(25);"*"
100 GO TO 120
110 PRINT TAB(25);"!";TAB(P+25);"*"
120 X=X+H
130 I=I+1
140 IF I<=16 GO TO 40
150 STOP
```

```
RUNNH
```



```
STOP AT LINE 150
```

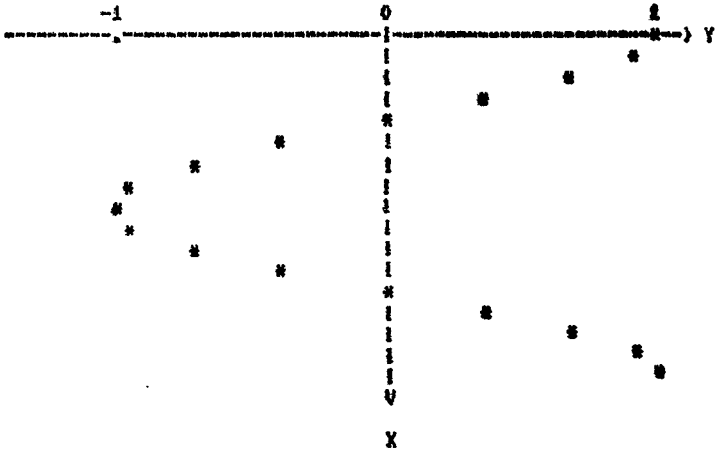
Символьный массив вначале должен быть заполнен пробелами, а далее, в зависимости от значения аргумента и вычисленного значения функции в соответствующий элемент массива вносится какой-либо символ, например *. При этом номера столбцов могут изменяться от 1 до 80 по ширине экрана. Число строк определяется сохранением наглядности выполняемого построения. Нужно иметь в виду, что экран дисплея имеет фиксированное число строк. Если массив будет содержать большее число строк, то изображение не поместится полностью на экране, однако может быть выведено на печатающее устройство.

При формировании в массиве график может быть ориентирован привычным для нас способом (ось ординат — вертикальная). Мас-

**Программа 4.48**

```
10 PRINT TAB(9);"-1";TAB(35);"0";TAB(60);"1"
20 FOR I=1 TO 10 \ PRINT "-"; \ NEXT I
30 PRINT ".";
40 FOR I=1 TO 24 \ PRINT "-"; \ NEXT I
50 PRINT "!";
60 FOR I=1 TO 24 \ PRINT "-"; \ NEXT I
70 PRINT "*";
80 FOR I=1 TO 2 \ PRINT "-"; \ NEXT I
90 PRINT ") Y"
100 H=PI/8 \ X=H
110 FOR I=1 TO 16
120 A=25*COS(X)
130 IF ABS(A)<1.00000E-04 GO TO 170
140 IF A>0 GO TO 180
150 PRINT TAB(A+35);"*";TAB(35);"!"
160 GO TO 190
170 PRINT TAB(35);"*" \ GO TO 190
180 PRINT TAB(35);"!";TAB(A+35);"*"
190 X=X+H
200 NEXT I
210 PRINT TAB(35);"V" \ PRINT  \ PRINT TAB(35);"X"

RUNNH
```




штаб и сдвиг при этом необходимо вводить как для функции, так и для аргумента.

Если $x_{min}$, $x_{max}$ — границы изменения аргумента, $y_{min}$, $y_{max}$ — границы изменения функции, то масштаб и сдвиг для аргумента и функции могут быть определены из соотношений

$$M_x x_{max} + S_x = J_{max},$$
$$M_x x_{min} + S_x = J_{min},$$
$$M_y y_{max} + S_y = I_{max},$$
$$M_y y_{min} + S_y = I_{min},$$

где $J_{min}$, $J_{max}$, $I_{min}$, $I_{max}$ определяют границы номеров столбцов и строк в массиве, в пределах которых будет располагаться график.

Для значений аргумента $x$ и функции $y$ номер строки $I$ и номер столбца $J$ определяются далее по формулам

$$I = S_y + M_y \cdot y,$$
$$J = S_x + M_x \cdot x.$$

Следует отметить, что порядок, в котором заполняются элементы массива, значения не имеет.

Формировать график можно и в числовом массиве (что существенно уменьшает необходимый объем памяти), кодируя соответствующие символы цифрами, например, 0 — пробел, 1 — точка графика *, 5 — ось ординат «!», 6 — ось абсцисс «—». При выводе на печать коды заменяются соответствующими символами. Такой способ применен в приведенном ниже примере.

Пример. Построить график функции $y=x^2$ при $x$, изменяющемся в пределах от —5 до 5 с шагом 1, с вычерчиванием осей координат.

Решение. Формирование графика будем осуществлять в массиве A (26, 60). Точку, соответствующую минимальному значению $x$, поместим в 5-й столбец ($x_{min}=-5$, $J_{min}=5$), максимальному значению $x$ — в 55-й столбец ($x_{max}=5$, $J_{max}=55$). Отсюда $S_x=30$, $M_x=5$. Учитывая порядок вывода строк массива на экран, точку (точки) с минимальным значением функции поместим в 26-ю строку ($y_{min}=0$, $I_{min}=26$), с максимальным значением в 1-ю ($y_{max}=25$, $I_{max}=1$). Следовательно, $S_y=26$, $M=-1$, т. е. номер строки и номер столбца для значений аргумента X и функции Y будут определяться соотношениями

$$J = 30 + 5X,$$
$$I = 26 - Y.$$

Ниже приводится программа (см. программу 4.49), осуществляющая формирование и печать графика функции $y=x^2$, и результат ее выполнения.

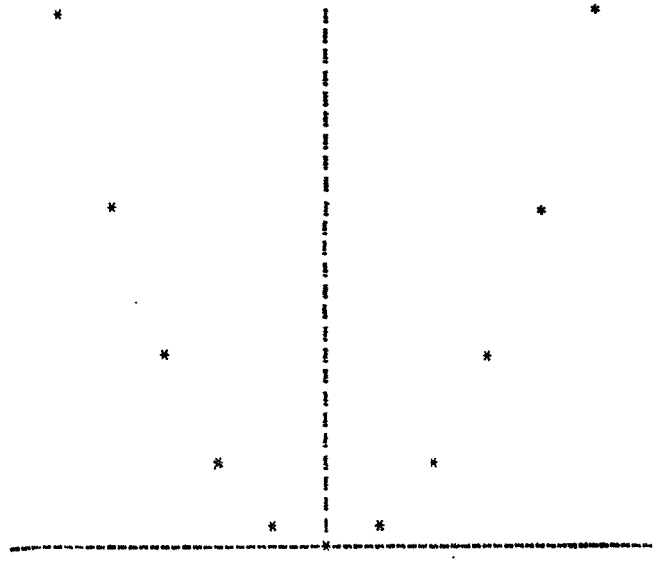
Пояснения к программе. Печать графика осуществляется по строкам. Для каждого значения I (I=1, . . ., 26) формируется строка в виде символьной переменной P$, которая оператором 250 выводится на экран. При формировании строки для печати в цикле по J (J=1, . . ., 60) к переменной P$ присоединяется по одному символу, значение которого определяется значением соответствующего элемента массива A (строки 190—240).

*3. Построение таблиц.* Чтобы вывести результаты в виде таблицы, нужно уметь напечатать (или вывести на экран) горизонтальные линии, вертикальные линии, разделяющие графы, напечатать шапку таблицы, т. е. содержание граф.

**Программа 4.49**

```
 10 DIM A(26,60)
 20 REM ЗАПОЛНЕНИЕ МАТРИЦЫ ПРОБЕЛАМИ. КОД ПРОБЕЛА - 0
 30 FOR I=1 TO 26
 40 FOR J=1 TO 60
 50 A(I,J)=0
 60 NEXT J
 70 NEXT I
 80 REM ЗАНОСИМ В МАТРИЦУ ОСИ КООРДИНАТ.КОДЫ ОСЕЙ: X - 6, Y - 5
 90 FOR I=1 TO 26 \ A(I,30)=5 \ NEXT I
100 FOR J=1 TO 60 \ A(26,J)=6 \ NEXT J
110 REM ПОСТРОЕНИЕ ГРАФИКА. КОД ГРАФИКА - 1
120 FOR X=-5 TO 5
130 Y=X*X
140 A(26-Y,30+5*X)=1
150 NEXT X
160 REM ПЕЧАТЬ ГРАФИКА
170 FOR I=1 TO 26
180 P$=""
190 FOR J=1 TO 60
200 IF A(I,J)=1 THEN P$=P$+"*" \ GO TO 240
210 IF A(I,J)=5 THEN P$=P$+"!" \ GO TO 240
220 IF A(I,J)=6 THEN P$=P$+"-" \ GO TO 240
230 P$=P$+" "
240 NEXT J
250 PRINT P$
260 NEXT I
```

RUNNH



Вывод горизонтальной линии в позиции от 1 до N обеспечивается операторами

```
FOR I=1 TO N  \ PRINT "-"; \ NEXT I \ PRINT
```

где последний оператор PRINT, не содержащий списка вывода, обеспечивает переход к следующей строке.

Печать вертикальных разделяющих линий обеспечивается выводом в одни и те же позиции каждой строки подходящего символа (например, I или !). Печать шапки таблицы осуществляется до входа в цикл выводом названий граф в одну или две строки в позиции, предназначенные для соответствующей информации.

После получения в цикле очередных результатов они печатаются в одни и те же позиции (графы таблицы) в последовательных строках. Средством, обеспечивающим вывод каждого получающегося результата в одни и те же позиции последовательных строк (друг под другом), является функция TAB с аргументом, заданным константой.

Пример. Напечатать таблицу функции $y=\mathrm{ch}\,x=(e^x+e^{-x})/2$ в интервале изменения $x$ от 0 до 2 с шагом 0, 2 (программа 4.50).

Пояснения к программе. Печать горизонтальной линии оформлена как подпрограмма (строки 140, 150), которая выполняется три раза при обращении к ней в строках 30, 50, 120.

Оператор в строке 10 печатает заголовок таблицы, начиная с 4-й позиции.

Оператор в строке 40 печатает шапку таблицы.

В цикле (строки 70—110) осуществляется вычисление функции CH(X) при одиннадцати значениях X (0; 0,2; 0,4; . . .; 2) и вывод в последовательные строки значений аргумента и функции, а также символов I, вычерчивающих вертикальные линии таблицы.

**Программа 4.50**

```
10 PRINT TAB(4);"ТАБЛИЦА ФУНКЦИИ CH(X)"
20 PRINT
30 GOSUB 140
40 PRINT "I";TAB(5);"X";TAB(10);"I";TAB(14);"CH(X)";TAB(23);"I"
50 GOSUB 140
60 X=0
70 FOR I=1 TO 11
80 Y=(EXP(X)+EXP(-X))/2
90 PRINT "I";TAB(2);X;TAB(10);"I";TAB(12);Y;TAB(23);"I"
100 X=X+.2
110 NEXT I
120 GOSUB 140
130 STOP
140 REM ПОДПРОГРАММА "ГОРИЗОНТАЛЬНАЯ ЛИНИЯ"
150 FOR J=1 TO 24 \ PRINT "-"; \ NEXT J \ PRINT  \ RETURN
```

```
RUNNH
   ТАБЛИЦА ФУНКЦИИ CH(X)
-----------------------
I    X    I   CH(X)    I
-----------------------
I  0      I  1         I
I  .2     I  1.02007   I
I  .4     I  1.08107   I
I  .6     I  1.18547   I
I  .8     I  1.33743   I
I  1      I  1.54308   I
I  1.2    I  1.81066   I
I  1.4    I  2.1509    I
I  1.6    I  2.57746   I
I  1.8    I  3.10747   I
I  2      I  3.7622    I
-----------------------
```

Одним из средств улучшения восприятия выводимых результатов является отсечение при печати незначащих цифр, появляющихся в процессе преобразования исходной информации (в процессе вычислений). Это обеспечивается использованием оператора PRINT USING (см. п. 5.5 главы 3). Использование этого оператора часто необходимо при выводе результатов в виде таблиц, когда под каждое данное отводится ограниченное число позиций. Например, если при печати таблицы функции CH(X) значение функции необходимо выводить с тремя цифрами после точки, то оператор PRINT в строке 90 нужно заменить (убрав строку 80) на

```
90 PRINT USING "I   #.#   I   ##.###   I";X;(EXP(X)+EXP(-X))/2
```

В результате выполнения модифицированной таким образом программы на экран будет выведено

```
   ТАБЛИЦА ФУНКЦИИ CH(X)

-----------------------
I    X    I   CH(X)    I
-----------------------
I  0.0    I    1.000   I
I  0.2    I    1.020   I
I  0.4    I    1.081   I
I  0.6    I    1.185   I
I  0.8    I    1.337   I
I  1.0    I    1.543   I
I  1.2    I    1.811   I
I  1.4    I    2.151   I
I  1.6    I    2.577   I
I  1.8    I    3.107   I
I  2.0    I    3.762   I
-----------------------
```

Для дальнейшего использования результатов вычислений их вывод нужно осуществлять на печатающем устройстве (ПУ) (см. п. 13.1 главы 3). Для этого необходимо добавить в программу операторы открытия и закрытия каналов связи с ПУ, а также заменить операторы PRINT на PRINT # N (N — номер канала связи от 1 до 6). В качестве примера модифицируем программу 4.50 для вывода результатов на ПУ (программа 4.51).

**Программа 4.51**

```
10 OPEN "LPt" FOR OUTPUT AS FILE #1
20 PRINT #1,TAB(4);"ТАБЛИЦА ФУНКЦИИ CH(X)"
30 PRINT #1
40 GOSUB 160
50 PRINT #1,"I";TAB(5);"X";TAB(10);"I";TAB(14);"CH(X)";TAB(23);"I"
60 GOSUB 160
70 X=0
80 FOR I=1 TO 11
90 Y=(EXP(X)+EXP(-X))/2
100 PRINT #1,"I";TAB(2);X;TAB(10);"I";TAB(12);Y;TAB(23);"I"
110 X=X+.2
120 NEXT I
130 GOSUB 160
140 CLOSE #1
150 STOP
160 REM ПОДПРОГРАММА "ГОРИЗОНТАЛЬНАЯ ЛИНИЯ"
170 FOR J=1 TO 24 \ PRINT #1,"-"; \ NEXT J \ PRINT #1 \ RETURN
```

*4. Построение гистограмм.* Одним из способов наглядного представления числовых величин, дающим возможность их качественного сравнения, является построение аналога гистограмм.

Каждое числовое значение будем представлять горизонтальной линией, построенной из каких-либо символов (например, *), выводимых с начала строки до позиции, номер которой зависит от выводимого числового значения. Последовательные значения будем выводить в последовательные строки экрана. Справа от линии можно напечатать числовое значение, представленное этой линией.

При организации вывода в виде гистограмм следует помнить, что число выводимых данных не должно превышать количества строк экрана (во избежании потери наглядности). При выводе на ПУ это ограничение теряет силу.

Для построения гистограммы нужно иметь информацию о диапазоне представляемых значений, а также выбрать номера позиций строк, в которых должны заканчиваться линии, соответствующие минимальному и максимальному из выводимых значений. На основании этого может быть выбран масштаб $M$ — приращение значения на одну позицию строки.

Для построения одной линии гистограммы нужно:

— вывести символы * в начало строки до позиции, соответствующей минимальному значению;

— организовать цикл, в котором увеличивать некоторую вспомогательную переменную от минимального значения до значения, соответствующего формируемой линии гистограммы, с шагом $M$, и на каждом шаге выводить в строку один символ *. Таким образом, количество выведенных в цикле символов * будет соответствовать (в принятом масштабе) превышению выводимого значения над минимальным.

Пример. Построить гистограмму урожайности поля за 10 лет с 1975 до 1984. Последней линией гистограммы вывести среднее значение урожайности поля за просмотренные годы.

Известно, что минимальное значение урожайности — 20 ц/га, максимальное — 50 ц/га (в общем случае минимальное и максимальное значения после ввода данных можно определить в программе).

Решение. Линию, соответствующую минимальному значению (20 ц/га), будем заканчивать в 7-й позиции, максимальному — в 67-й. Таким образом, масштаб $M=0{,}5$ ц/га на одну позицию (одну *). В начале каждой линии будем печатать год, которому она соответствует (в позициях 1—4 и ** в позициях 6—7, т. е. фактически выводится часть гистограммы выше уровня 19 ц/га. Если выводимое значение больше минимального, то количество выводимых в строку звездочек будем увеличивать в соответствии с выводимым значением. Данный алгоритм реализован в программе 4.52.

*Список используемых переменных.* Исходные данные: A — год, Y — урожайность.

Результат: гистограмма, построенная на экране дисплея.

Вспомогательные переменные: S — переменная для определения значения средней урожайности поля, N — счетчик лет, C — управляющая переменная цикла построения линии гистограммы.

Исходные данные для построения линий гистограммы заданы в операторах DATA и при выполнении программы последовательно считываются оператором READ.

Можно было бы ввести исходные данные в массив и затем для построения гистограммы организовать цикл по индексу. Программа 4.53 составлена для получения гистограммы урожайности поля с использованием массива Y1 (урожайность) размером 10.

Пояснения к программе. В качестве счетчика лет использован индекс I.

Перед обращением к подпрограмме «Линия гистограммы» параметру подпрограммы Y присваивается значение очередного (I-го) элемента массива Y1.

*5. Построение фигур.* Используя функцию TAB, на экране дисплея можно строить различные фигуры, что позволяет во многом

Программа 4.52

```
10 S=0 \ N=0
11 DATA 1975,25,1976,30,1977,28,1978,35,1979,40
12 DATA 1980,33,1981,29,1982,34,1983,30,1984,36
13 DATA 0
20 READ A
30 IF A=0 GO TO 70
31 READ Y
40 PRINT A;"**";
50 GOSUB 2000
60 S=S+Y \ N=N+1 \ GO TO 20
70 S=S/N
80 PRINT " СР.  **";
90 Y=S \ GOSUB 2000
100 STOP
2000 REM ПОДПРОГРАММА "ЛИНИЯ ГИСТОГРАММЫ"
2010 C=20
2020 IF C)=Y GO TO 2060
2030 C=C+.5
2040 PRINT "*";
2050 GO TO 2020
2060 PRINT TAB(55);Y
2070 RETURN

RUNNH

 1975 ************                                          25
 1976 ************************                              30
 1977 ******************                                    28
 1978 ************************************                  35
 1979 ****************************************************  40
 1980 ****************************                          33
 1981 **********************                                29
 1982 ********************************                      34
 1983 **********************                                30
 1984 ************************************                  36
 СР.  ************************                              32

STOP AT LINE 100
```

улучшить наглядность представления результатов, а также использовать ЭВМ для решения ряда нечисленных задач, в том числе и для программирования игр. Методы построения фигур рассмотрим на примерах.

Пример 1. Построение квадрата.

Требуется построить на экране дисплея два квадрата со сторонами из A символов, отстоящих на A символов друг от друга, и заполнить их символами *.

Для решения задачи нужно вывести одинаковый рисунок звездочек в A строк. Этот рисунок содержит по условию задачи две последовательности по A звездочек, разделенных A пробелами.

**Программа 4.53**

```
10 DIM Y1(10)
20 FOR I=1 TO 10
30 INPUT Y1(I)
40 NEXT I
50 A=1974 \ S=0
60 FOR I=1 TO 10
70 PRINT A+I;"**"; \ Y=Y1(I)
80 GOSUB 2000
90 S=S+Y
100 NEXT I
110 S=S/10
120 PRINT " СР.  **";
130 Y=S \ GOSUB 2000
140 STOP
2000 REM ПОДПРОГРАММА "ЛИНИЯ ГИСТОГРАММЫ"
2010 C=20
2020 IF C)=Y GO TO 2060
2030 C=C+.5
2040 PRINT "*";
2050 GO TO 2020
2060 PRINT TAB(55);Y
2070 RETURN
```

Печать одной строки можно осуществить двумя выполняемыми последовательно циклами. Первый выводит звездочки в первые A позиций, начиная с N=1, второй — в A позиций, начиная с (2A+1)-й позиции. Вывод одной последовательности из A звездочек выполняется аналогично печати горизонтальной линии таблицы и оформлен в программе 4.54 как подпрограмма.

Выполнение указанной последовательности действий A раз обеспечит вывод A строк, т. е. построение двух квадратов с длинами сторон, равными A.

На значение A наложены ограничения, связанные с количеством позиций в одной строке экрана. В связи с этим в программе 4.54 предусмотрен контроль вводимого значения A (оператор 30).

Фигуры, построенные в результате выполнения программы 4.54, являются квадратами лишь в том смысле, что каждая их сторона содержит одинаковое количество звездочек.

Пример 2. Построение треугольника.

Требуется построить прямоугольный треугольник с основанием A символов и высотой A символов.

Для решения задачи нужно в A последовательных строк выводить * так, что в первую строку выводится одна *, а в каждую следующую строку на одну звездочку больше, чем в предыдущую, т. е. количество выводимых в строку звездочек совпадает с номером строки.

Программа 4.54

```
10 PRINT "ВВЕДИТЕ ДЛИНУ СТОРОНЫ КВАДРАТА "
20 INPUT A
30 IF A>23 GO TO 10
40 FOR I=1 TO A
50 N=1 \ GOSUB 1000
60 N=2*A+1 \ GOSUB 1000
70 PRINT
80 NEXT I
90 STOP
1000 REM ПОДПРОГРАММА "ЛИНИЯ"
1010 PRINT TAB(N);"*";
1020 FOR J=2 TO A
1030 PRINT "*";
1040 NEXT J
1050 RETURN

RUNNH

ВВЕДИТЕ ДЛИНУ СТОРОНЫ КВАДРАТА
? 10
 **********          **********
 **********          **********
 **********          **********
 **********          **********
 **********          **********
 **********          **********
 **********          **********
 **********          **********
 **********          **********
 **********          **********

STOP AT LINE 90
```

В программе должен быть организован внешний цикл по номеру строки I. Для каждого I должен выполняться внутренний цикл (по J), обеспечивающий печать * в позициях с 1-й по I-ю (см. программу 4.55).

Задание I уровня. Для заданной функции определить масштаб и сдвиг. Составить схему и программу для построения графика функции в заданном интервале с заданным шагом. Предусмотреть вывод осей координат. Составить программу для построения таблицы значений аргумента и функции. Предусмотреть вывод заголовка и шапки таблицы. Проверить работу программ на ЭВМ.

Задание II уровня. Для вариантов задач, приведенных в задании I уровня составить программу для построения графика с использованием массива. Предусмотреть печать осей координат. Составить программу для построения таблицы значений аргумента и функции с отсечением части цифр. Использовать опе-

**Программа 4.55**

```
10 PRINT "ВВЕДИТЕ ОСНОВАНИЕ ТРЕУГОЛЬНИКА"
20 INPUT A
30 FOR I=1 TO A
40 PRINT TAB(1);"*";
50 FOR J=2 TO I
60 PRINT "*";
70 NEXT J
80 PRINT
90 NEXT I
100 STOP

RUNNH

ВВЕДИТЕ ОСНОВАНИЕ ТРЕУГОЛЬНИКА
? 10
 *
 **
 ***
 ****
 *****
 ******
 *******
 ********
 *********
 **********

STOP AT LINE 100
```

*Варианты задач I уровня.*

| № п/п | Функция | Интервал | Шаг |
|---|---|---|---|
| 1 | $y = e^x \sin x$ | $[0;\ 2\pi]$ | $\pi/10$ |
| 2 | $y = x^3$ | $[-5;\ 5]$ | 0,5 |
| 3 | $y = x \sin x$ | $[0;\ 2\pi]$ | $\pi/10$ |
| 4 | $y = \lvert x \rvert$ | $[-10;\ 10]$ | 1 |
| 5 | $y = (\sin x)/x$ | $[\pi;\ 2\pi]$ | $\pi/10$ |
| 6 | $y = \sqrt{x}$ *) | $[0;\ 64]$ | 4 |
| 7 | $y = x \cos x$ | $[0;\ 2\pi]$ | $\pi/10$ |
| 8 | $y = x^2 + 4x$ | $[-5;\ 5]$ | 1 |
| 9 | $y = \operatorname{tg} x$ | $[-\pi;\ \pi]$ | $\pi/5$ |
| 10 | $y = x^2 + \sin x$ | $[-\pi;\ \pi]$ | $\pi/8$ |
| 11 | $y = 2x^2 + x + 4$ | $[-5;\ 5]$ | 1 |
| 12 | $y = \operatorname{sh} x$ | $[-1;\ 1]$ | 0,1 |
| 13 | $y = \operatorname{th} x$ | $[0;\ 2]$ | 0,2 |
| 14 | $y = \operatorname{ch} x$ | $[-1;\ 1]$ | 0,1 |
| 15 | $y = -2x^2$ | $[-5;\ 5]$ | 1 |

*) Предусмотреть вывод двух ветвей функции.

ратор PRINT USING. Модифицировать программу для вывода результатов на ПУ. Проверить работу всех программ на ЭВМ.

Задание III уровня требует вывода результатов в виде гистограмм и построения на экране дисплея и на ПУ геометрических фигур.

Составить программу в двух вариантах: с выводом результатов на экран дисплея и на ПУ. Проверить работу программы на ЭВМ. После отладки программы результаты выдать на ПУ.

*Варианты задач III уровня.*

1. Для анализа работы межколхозной ГЭС собраны среднемесячные данные по расходу воды за 1 год. Изобразить их в виде гистограммы. В начале линии гистограммы вывести название месяца. Диапазон данных: от 100 м³/с до 500 м³/с.

2. Построить гистограмму распределения количества крупных стихийных бедствий, происходящих в мире, по годам с 1960 по 1979. В начале линии вывести год. Диапазон: от 43 до 81 бедствия.

3. Построить гистограмму количества картофеля, собранного каждой из 10 студенческих групп. В начале каждой линии гистограммы вывести номер группы. Диапазон: от 10 т до 40 т.

4. ЭВМ ведет учет вторсырья, собранного школьниками за год. Построить гистограмму количества вторсырья, собранного в школе по классам (в школе по два класса с 1-го по 8-й и по одному классу 9, 10). В начале линии напечатать класс. Диапазон: от 800 кг до 2 т.

5. Построить гистограмму числа побед, одержанных за один сезон каждой из 16 команд высшей лиги по футболу. В начале линии напечатать название команды. Диапазон: от 5 до 25 побед.

6. При проведении конкурса песни оценивалась также популярность артиста по количеству голосов, поданных за него зрителями. Построить гистограмму сравнительной оценки популярности 10 артистов. В начале каждой линии вывести фамилию артиста. Диапазон: от 30 до 150 голосов.

7. Построить гистограмму распределения забитых мячей для 10 лучших футболистов СССР. В начале линии гистограммы вывести фамилию спортсмена. Диапазон: от 100 до 200 мячей.

8. Построить равнобедренный треугольник высотой H с основанием A (A=9, H=5).

9. Построить ромб, диагонали которого равны A и B. Ромб расположить так, чтобы одна из диагоналей располагалась горизонтально (A=7, B=9).

10. Построить равнобедренную трапецию, высота которой равна H, основания — A и B. Основания расположены горизонтально (A=14, B=4, H=6).

11. В задаче 10 основания расположить вертикально.

12. Построить прямоугольную трапецию (высота — H, осно-

вания — A и B). Основания расположить горизонтально (A=9, B=4, H=6).

13. Построить параллелограмм со сторонами A и B. Сторона A расположена горизонтально, сторона B состоит из символов *, смещаемых в каждой строке на одну позицию влево (A=8, B=5).

14. Построить правильный шестиугольник со стороной A. Две стороны расположены горизонтально (A=5).

15. Построить правильный шестиугольник со стороной A. Две стороны расположены вертикально (A=5).

Указания к решению задач III уровня. Фигуры, предлагаемые для построения в задачах 8—15, могут рассматриваться как объединения различных треугольников и квадратов. Программы для построения последних приведены в теоретическом введении.

Вопросы для самопроверки.

1. Роль функции TAB в операторе PRINT. Ограничения на значения аргумента функции TAB.

2. Для чего используется оператор PRINT USING? Как обеспечить требуемый формат печати для вывода значений числовых и символьных величин?

3. Построение графика при помощи функции TAB. Когда необходимо вводить масштаб и сдвиг для значений функции?

4. Как обеспечить вычерчивание осей координат при построении графика?

5. Построение графика с использованием массива. В каких случаях целесообразно использовать такое построение? Каковы особенности построения графика с использованием массива? Какие это дает дополнительные возможности по сравнению с использованием функции TAB?

6. Как вывести результаты в виде таблицы? Как вывести на экран горизонтальные линии? Чем обеспечивается вывод вертикальных линий таблицы?

7. Как организовать вывод результатов работы программы на ПУ?

8. Как организовать вывод результатов в виде гистограммы? Как вывести одну линию гистограммы?

9. Для чего необходимо вводить масштаб при построении гистограммы? Какие данные используются при выборе масштаба? Можно ли масштаб определять в программе, если заранее не известен диапазон выводимых значений?

10. Как построить на экране квадрат, прямоугольник, треугольник? Можно ли такой же способ использовать для вывода на экран букв большого формата?

## *Работа 8.* РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

Теоретическое введение. Перед выполнением работы 8 необходимо ознакомиться с п. 7 описания языка бейсик (глава 3) и выполнить работу 4.

Решение большинства реальных задач на ЭВМ требует организации данных в виде массивов. Можно выделить некоторые типовые алгоритмы, используемые как составная часть алгоритмов обработки данных, организованных в виде массивов. Далее рассматриваются часто встречающиеся типовые алгоритмы обработки массивов (простейшие алгоритмы рассмотрены в работе 4). Алгоритмы приводятся в виде подпрограмм на бейсике, для некоторых алгоритмов приводятся также схемы. Ввод данных и печать результатов в подпрограммах не предусматривается. При разработке алгоритмов использовались только типовые структуры. Приведенные подпрограммы можно включать как составные части в программы решения задач, требующих их использования. При этом нужно следить за тем, чтобы массивы, фигурирующие в подпрограмме, были описаны и чтобы исходные данные получили значения перед обращением к подпрограмме.

Все алгоритмы составлены применительно к одномерным массивам.

*1. Определение числа элементов массива, удовлетворяющих заданному условию.* Требуется определить, сколько элементов заданного массива $P$, содержащего $n$ элементов, удовлетворяет заданному условию (для определенности пусть условие имеет вид $P_i > T$, где $T$ — заданное число).

Для решения задачи следует организовать цикл по $i$ и для каждого значения $i$ проверять условие $P_i \leqslant T$. Если условие удовлетворяется, осуществляется переход к следующему элементу массива, а если не удовлетворяется (т. е. $P_i > T$), то к счетчику числа элементов, удовлетворяющих условию $P_i > T$, прибавляется 1 (программа 4,56).

*Список используемых переменных.* Исходные данные: N — размер массива, P — массив размером N, T — заданное значение, с которым сравниваются элементы массива.

Результат: K — число элементов массива P, удовлетворяющих заданному условию.

Вспомогательные переменные: I — индекс — управляющая переменная цикла.

**Программа 4.56**

```
1000 REM ОПРЕДЕЛЕНИЕ ЧИСЛА ЭЛЕМЕНТОВ, УДОВЛЕТВОРЯЮЩИХ
1001 REM ЗАДАННОМУ УСЛОВИЮ
1010 K=0
1020 FOR I=1 TO N
1030 IF P(I)<=T GO TO 1050
1040 K=K+1
1050 NEXT I
1060 RETURN
```

*2. Суммирование элементов массива, удовлетворяющих заданному условию.* Требуется найти сумму элементов массива $P$, содер-

жащего $n$ элементов, удовлетворяющих заданному условию, например условию $P_i > T$ (программа 4.57).

*Список используемых переменных.* Исходные данные: N — размер массива, P — массив размером N, T — заданное значение, с которым сравниваются элементы массива P.

Результат: S — сумма элементов массива P, удовлетворяющих заданному условию.

Вспомогательные переменные: I — индекс — управляющая переменная цикла.

**Программа 4.57**

```
1000 REM СУММИРОВАНИЕ ЭЛЕМЕНТОВ МАССИВА, УДОВЛЕТВОРЯЮЩИХ
1001 REM ЗАДАННОМУ УСЛОВИЮ
1010 S=0
1020 FOR I=1 TO N
1030 IF P(I)<=T GO TO 1050
1040 S=S+P(I)
1050 NEXT I
1060 RETURN
```

*3. Объединение двух массивов в один с чередованием элементов исходных массивов.* Требуется объединить два массива $A$ и $B$, содержащие по $n$ элементов, в один массив $C$, который будет содержать $2n$ элементов, т. е. получить массив $C=(a_1, b_1, a_2, b_2, \ldots, a_n, b_n)$.

Можно заметить, что индекс элемента массива $C$ зависит от индекса пересылаемого в него элемента массива $A$ (или $B$), а именно, $c_{2i-1}=a_i$, $c_{2i}=b_i$. Таким образом, организовав цикл по $i$ и выполняя для каждого $i$ эти присваивания, мы решим задачу (программа 4.58).

*Список используемых переменных.* Исходные данные: N — размер массивов, A, B — массивы размером N.

Результат: C — массив размером 2N.

Вспомогательные переменные: I — индекс элементов массивов A и B — управляющая переменная цикла.

**Программа 4.58**

```
1000 REM ОБ'ЕДИНЕНИЕ ДВУХ МАССИВОВ ОДИНАКОВОГО РАЗМЕРА
1010 FOR I=1 TO N
1020 C(2*I-1)=A(I)
1030 C(2*I)=B(I)
1040 NEXT I
1050 RETURN
```

*4. Инвертирование массива.* Требуется изменить порядок следования элементов массива $C$, состоящего из $n$ элементов, на обратный, используя одну вспомогательную переменную. Результат получить в том же массиве ($C$).

Сначала поменяем местами 1-й и $n$-й элементы, используя вспомогательную переменную $P$. Для этого перешлем $a_1$ в $P$ ($P=a_1$), затем в $a_1$ перешлем $a_n$ ($a_1=a_n$), а далее, значение $a_1$, которое временно находится в $P$, перешлем в $a_n$ ($a_n=P$). Так же поступим с элементами $a_2$ и $a_{n-1}$ и т. д., пока не дойдем до середины массива. Последними элементами, которые нужно поменять местами, будут $a_{n/2}$ и $a_{n/2+1}$, если $n$ — четное, и $a_{[n/2]}$ и $a_{[n/2]+2}$, если $n$ — нечетное, т. е. цикл по $i$ в общем случае можно организовать от $i=1$ до $[n/2]$ (программа 4.59).

*Список используемых переменных.* Исходные данные: N — размер массива, C — массив размером N.

Результат: C — инвертированный исходный массив.

Вспомогательные переменные: I — индекс — управляющая переменная цикла, M=[N/2] — вычисляется до входа в цикл для уменьшения объема вычислений.

**Программа 4.59**

```
1000 REM ИНВЕРТИРОВАНИЕ МАССИВОВ
1010 M=INT(N/2)
1020 FOR I=1 TO M
1030 P=C(I) \ C(I)=C(N-I+1) \ C(N-I+1)=P
1040 NEXT I
1050 RETURN
```

*5. Формирование массива из элементов другого массива, удовлетворяющих заданному условию.* Требуется из данного массива $A$, состоящего из $n$ элементов, выбрать элементы, удовлетворяющие заданному условию (например, условию $A_i \geqslant T$), и сформировать из них массив $B$.

Особенностью решения этой задачи является то, что индексы элементов массивов $A$ и $B$ не совпадают, так как не все элементы массива $A$ включаются в массив $B$. Следовательно, для обозначения индекса элементов массива $B$ нужно предусмотреть другую переменную ($j$), значение которой будем изменять на 1 перед занесением в массив $B$ нового значения. До входа в цикл нужно положить $j=0$ (программа 4.60).

*Список используемых переменных.* Исходные данные: N — размер массива, A — массив размером N, T — заданное значение для проверки условия включения элемента массива A в массив B.

Результат: B — массив размером не больше N, J — число элементов массива B,

Вспомогательные переменные: I — индекс элементов массива, A — управляющая переменная цикла.

*6. Поиск заданного элемента в массиве.* Требуется определить, есть ли в заданном массиве $P$, состоящем из $n$ элементов, элемент, равный $L$ (массив может быть как числовым, так и символьным). Результат присвоить символьной переменной (программа 4.61).

**Программа 4.60**

```
1000 REM ВКЛЮЧЕНИЕ В МАССИВ ЭЛЕМЕНТОВ
1001 REM УДОВЛЕТВОРЯЮЩИХ УСЛОВИЮ
1010 J=0
1020 FOR I=1 TO N
1030 IF A(I)<T GO TO 1050
1040 J=J+1 \ B(J)=A(I)
1050 NEXT I
1060 RETURN
```

*Список используемых переменных.* Исходные данные: N — размер массива (N<1000), P — массив размером N, L — значение, которое ищется в массиве.

Результат: R$ имеет значения «ЭЛЕМЕНТ, РАВНЫЙ L, ЕСТЬ» или «ЭЛЕМЕНТА, РАВНОГО L, НЕТ» в зависимости от результатов поиска.

Вспомогательные элементы: I — индекс — управляющая переменная цикла.

**Программа 4.61**

```
1000 REM ПОИСК ЗАДАННОГО ЭЛЕМЕНТА В МАССИВЕ
1010 R$="ЭЛЕМЕНТА, РАВНОГО L НЕТ"
1020 FOR I=1 TO N
1030 IF P(I)<>L GO TO 1050
1040 R$="ЭЛЕМЕНТ, РАВНЫЙ L ЕСТЬ" \ I=1000
1050 NEXT I
1060 RETURN
```

Пояснения к программе. Если элемент, равный L, найден, то, чтобы завершить цикл, I присваивается значение, большее размера массива (I=1000). Этот формальный прием позволяет использовать только типовые структуры алгоритмов, имеющие один вход и один выход.

*7. Циклический сдвиг элементов массива.* Требуется переместить элементы массива $A$, состоящего из $n$ элементов, вправо (влево) на $m$ позиций, при этом $m$ элементов из конца массива перемещается в начало. Например, результатом циклической перестановки исходного массива $A=(a_1, a_2, a_3, a_4, a_5)$ вправо на 2 позиции будет $A=(a_4, a_5, a_1, a_2, a_3)$.

Рассмотрим два варианта алгоритма решения задачи:

— с использованием вспомогательного массива;

— с использованием одной вспомогательной переменной.

В первом варианте (программа 4.62) «хвост» массива (элементы $a_{n-m+1}, \ldots, a_n$) пересылается во вспомогательный массив, все остальные элементы перемещаются вправо на $m$ позиций ($a_{i+m}=a_i$, $i=n-m, n-m-1, \ldots, 1$). Следует обратить внимание на обратный порядок перемещения элементов (с конца). Прямой порядок (с начала) привел бы к искажениям исходного массива.

Далее, в первые элементы массива $A$ $(a_1, \ldots, a_m)$ пересылаются элементы вспомогательного массива, в котором временно хранится «хвост» исходного массива.

Во втором варианте (программа 4.63) во вспомогательную переменную каждый раз пересылается последний элемент массива $A$, затем все элементы сдвигаются вправо на одну позицию (в обратном порядке) и на место первого элемента помещается содержимое вспомогательной переменной. Эта процедура повторяется $m$ раз.

Первый вариант требует больше памяти (для вспомогательного массива), а второй — больших затрат времени на многократное перемещение элементов массива.

*Список используемых переменных.* Исходные данные: N — размер массива, A — массив размером N, M — число позиций сдвига.

Результат: A — массив, циклически сдвинутый на M позиций вправо.

Вспомогательные переменные: I — индекс — управляющая переменная цикла, P — массив размером не менее M (вариант 1) для временного хранения «хвоста» массива, P — переменная (вариант 2) для временного хранения элемента массива A, J — управляющая переменная внутреннего цикла (вариант 2).

**Программа 4.62**

```
1000 REM ЦИКЛИЧЕСКИЙ СДВИГ - ВАРИАНТ 1
1010 FOR I=1 TO M
1020 P(I)=A(N-M+I)
1030 NEXT I
1040 FOR I=N-M TO 1 STEP -1
1050 A(I+M)=A(I)
1060 NEXT I
1070 FOR I=1 TO M
1080 A(I)=P(I)
1090 NEXT I
1100 RETURN
```

Программа 4.63

```
1000 REM ЦИКЛИЧЕСКИЙ СДВИГ - ВАРИАНТ 2
1010 FOR I=1 TO M
1020 P=A(N)
1030 FOR J=N TO 2 STEP -1
1040 A(J)=A(J-1)
1050 NEXT J
1060 A(1)=P
1070 NEXT I
1080 RETURN
```

*8. Упорядочение массива.* Требуется расположить элементы массива в порядке возрастания (убывания).

Для решения этой задачи существует много различных методов [4, т. 3]. Здесь рассматривается один из методов, основанный

на поиске минимального (максимального) элемента массива или его части.

Вначале найдем минимальный элемент массива и поменяем его местами с первым элементом, затем найдем минимальный элемент из оставшихся элементов (кроме первого) и поменяем его местами со вторым элементом. После нахождения минимального из последних двух элементов массива и размещения его на предпоследнем месте на последнем автоматически останется самый большой элемент (рис. 4.13 и программа 4.64).

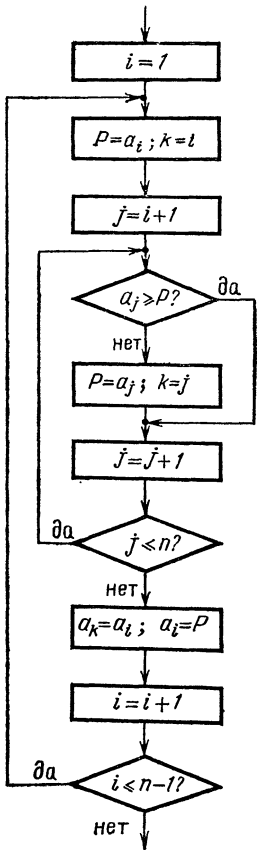


Рис. 4.13

*Список используемых переменных.* Исходные данные: N — размер массива, A — массив размером N.

Результат: A — массив размером N, упорядоченный по возрастанию.

Вспомогательные переменные: P — переменная для хранения промежуточного значения минимального элемента, K — индекс минимального элемента, I — индекс элемента упорядоченного массива — управляющая переменная внешнего цикла, J — индекс элемента части массива, в которой ищется минимальный элемент — управляющая переменная внутреннего цикла.

При перестановке элементов (минимального и I-го (строка 1070)) вспомогательная переменная не требуется, так как значение минимального элемента находится в переменной P.

*9. Проверка массива на упорядоченность.* Для заданного массива A размером $n$ требуется определить, является ли массив упорядоченным. Результат присвоить символьной переменной.

Для определенности предположим, что проверяется упорядоченность массива по возрастанию. Если массив упорядочен, то для каждой пары соседних элементов должно выполняться условие $a_i < a_{i+1}$, $i=1, \ldots, n-1$.

Если ни для какого $i$ условие не было нарушено, то массив упорядочен. Если для какого-либо $i$ условие нарушается, массив не является упорядоченным (программа 4.65).

*Список используемых переменных.* Исходные данные: N — размер массива, A — массив размером N.

**Программа 4.64**

```
1000 REM УПОРЯДОЧЕНИЕ МАССИВА ПО ВОЗРАСТАНИЮ
1010 FOR I=1 TO N-1
1020 P=A(I) \ K=I
1030 FOR J=I+1 TO N
1040 IF A(J)=P GO TO 1060
1050 P=A(J) \ K=J
1060 NEXT J
1070 A(K)=A(I) \ A(I)=P
1080 NEXT I
1090 RETURN
```

Результат: T$ имеет значение «Массив упорядочен» или «Массив не упорядочен».

Вспомогательные переменные: I — управляющая переменная цикла (индекс).

**Программа 4.65**

```
10 REM ПРОВЕРКА МАССИВА НА УПОРЯДОЧЕННОСТЬ
20 T$="МАССИВ УПОРЯДОЧЕН"
30 FOR I=1 TO N-1
40 IF A(I)<A(I+1) GO TO 60
50 I=N \ T$="МАССИВ НЕ УПОРЯДОЧЕН"
60 NEXT I
70 RETURN
```

Пояснения к программе. Если условие упорядоченности для всех соседних элементов выполняется, т. е. после проверки условия (строка 40) каждый раз осуществляется переход на оператор 60, строка 50 ни разу не выполняется и T$ не изменяет значения, присвоенного в начале программы.

Если хотя бы для одной пары условие упорядоченности (строка 40) не выполняется, осуществляется переход к строке 50, где T$ присваивается новое значение, а I приобретает значение, большее верхней границы, что вызовет завершение цикла и переход к оператору RETURN.

*10. Поиск заданного элемента в упорядоченном массиве.* Если в заданном массиве поиск осуществляется многократно, то целесообразно сначала массив упорядочить, чтобы затем быстрее производить поиск. Предположим, что массив упорядочен по возрастанию. Рассмотрим алгоритм так называемого бинарного поиска.

Требуется в массиве $M$ определить индекс $C$ элемента, равного заданному значению $K$. Массив $M$ упорядочен по возрастанию.

Идея алгоритма заключается в следующем. Заданное число $K$ сравнивается со средним элементом массива $M$ (имеющим, например, индекс $C=\left[\frac{A+B}{2}\right]$, где $A$ — нижняя граница, $B$ — верхняя

граница индексов (в начале $A=1$, $B=N$, где $N$ — размер массива). Если $M(C)\neq K$, то далее, поиск продолжается в одной из половин массива (исключая элемент $M(C)$) в зависимости от результата сравнения. Для этого изменяется значение или $A$ ($A=C+1$) или $B$ ($B=C-1$). Заданное число $K$ сравнивается со средним элементом в выбранной половине и т. д. (программа 4.66).

*Список используемых переменных.* Исходные данные: N — размер массива, M — упорядоченный по возрастанию массив размером N, K — заданное число.

Результат: L—L=0, если элемент со значением K не найден, L=1, если элемент найден, C — индекс элемента, имеющего значение K.

**Программа 4.66**

```
1000 REM БИНАРНЫЙ ПОИСК
1010 A=1 \ B=N \ L=0
1020 IF B<A GO TO 1090
1030 C=INT((A+B)/2)
1040 IF M(C)=K GO TO 1080
1050 IF M(C)>K GO TO 1070
1060 A=C+1 \ GO TO 1020.
1070 B=C-1 \ GO TO 1020
1080 L=1 \ B=A-1 \ GO TO 1020
1090 RETURN
```

Пояснения к программе. Часть массива, в которой ищется значение K, постоянно сужается. Поиск заканчивается, когда в ней не остается ни одного элемента (выполняется условие B<A в строке 1020).

Если значение K найдено (выполняется условие в строке 1040), то для обеспечения выхода из цикла полагается B=A—1 (строка 1080).

*11. Объединение двух упорядоченных массивов одного размера в один, так же упорядоченный.* Требуется объединить два упорядоченных по возрастанию (убыванию) массива $A$ и $B$ размером $N$ в один массив размером $2N$ также упорядоченный по возрастанию (убыванию).

Индексы массивов $A$ и $B$ будем менять независимо. Если выполняется условие $a_i<b_j$, то в массив $C$ пересылаем элементы массива $A$ (индекс $j$ при этом не меняется) до тех пор, пока это условие не нарушается. Тогда в массив $C$ начинаем пересылать элементы массива $B$ (индекс $i$ при этом не меняется) и т. д., пока не закончится пересылка элементов обоих массивов (будет выполняться условие $i>n$ и $j>n$). Если одно из этих условий выполнено, т. е. один из массивов полностью исчерпан, то «хвост» другого массива пересылается элемент за элементом без каких-либо проверок (см. рис. 4.14 и программу 4.67).

*Список используемых переменных.* Исходные данные: N — размер исходных массивов, A, B — упорядоченные массивы размером N.

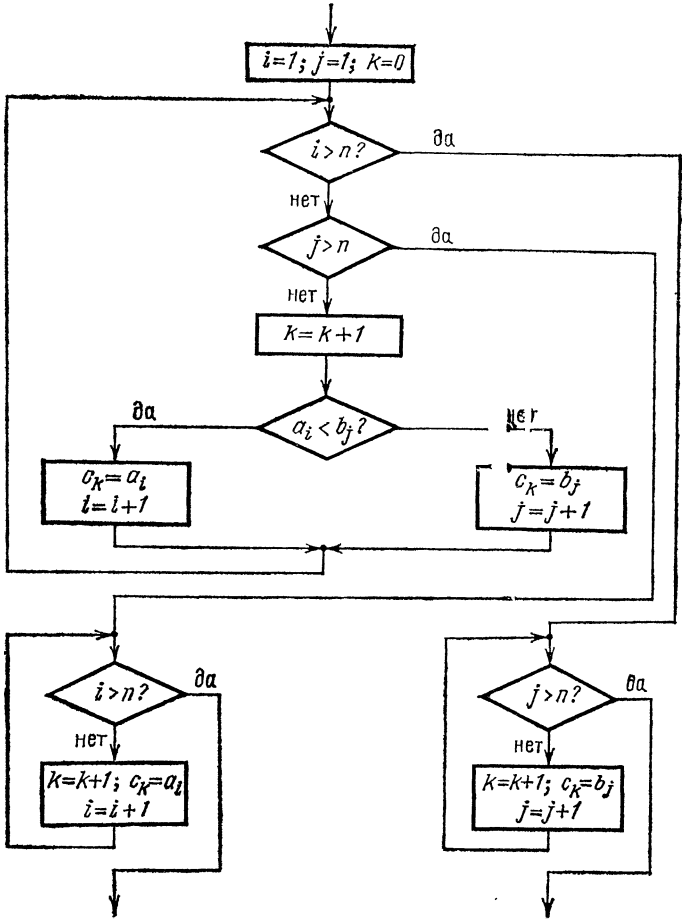



Рис. 4.14

Результат: C — упорядоченный массив размером 2N.

Вспомогательные переменные: I, J — индексы массивов A и B, K — индекс массива C.

*12. Сравнение двух упорядоченных по возрастанию массивов.* Требуется определить количество совпадающих элементов двух упорядоченных массивов A и B. Размеры массивов не обязательно одинаковы.

**Программа 4.67**

```
1000 REM ОБ'ЕДИНЕНИЕ ДВУХ УПОРЯДОЧЕННЫХ МАССИВОВ
1010 I=1 \ J=1 \ K=0
1020 IF I>N GO TO 1100
1030 IF J>N GO TO 1140
1040 K=K+1
1050 IF A(I)<B(J) GO TO 1080
1060 C(K)=B(J) \ J=J+1
1070 GO TO 1020
1080 C(K)=A(I) \ I=I+1
1090 GO TO 1020
1100 IF J>N GO TO 1130
1110 K=K+1 \ C(K)=B(J) \ J=J+1
1120 GO TO 1100
1130 RETURN
1140 IF I>N GO TO 1170
1150 K=K+1 \ C(K)=A(I) \ I=I+1
1160 GO TO 1140
1170 RETURN
```

Сравнение начинаем с первых элементов ($k=1$, $l=1$), и если они совпадают, то увеличиваем результат (число совпадающих элементов) на 1 и переходим к следующим элементам ($k=k+1$,

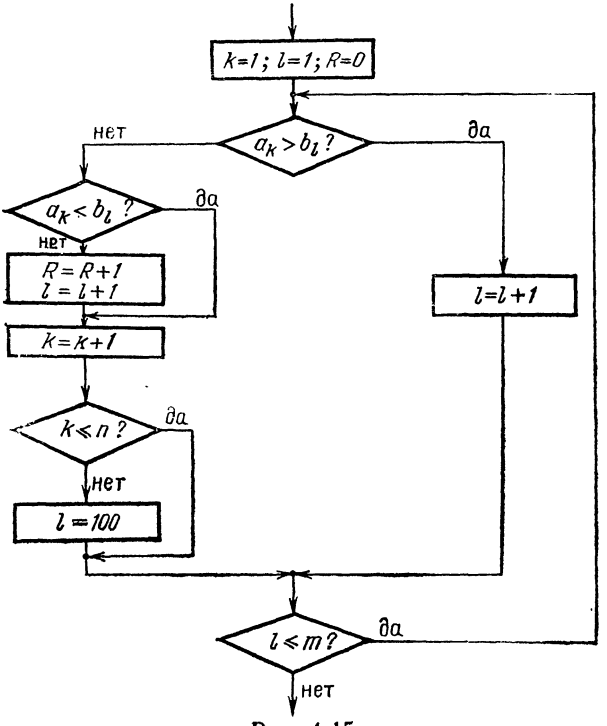



Рис. 4.15

$l=l+1$). В противном случае происходит движение (изменение индекса) по тому массиву, значение элементов которого меньше, индекс второго массива не меняется. Алгоритм должен закончить работу, если исчерпан хотя бы один массив (см. рис. 4.15 и программу 4.68).

*Список используемых переменных.* Исходные данные: N, M — размеры заданных массивов, N, M<100, A, B — упорядоченные массивы размером N и M.

Результат: R — число совпадающих элементов массивов A и B.

Вспомогательные переменные: K, L — индексы массивов A и B.

**Программа 4.68**

```
1000 REM СРАВНЕНИЕ ДВУХ УПОРЯДОЧЕННЫХ МАССИВОВ
1010 K=1 \ R=0
1020 FOR L=1 TO M
1030 IF A(K)>B(L) GO TO 1090
1040 IF A(K)<B(L) GO TO 1060
1050 R=R+1  \ L=L+1
1060 K=K+1
1070 IF K<=N GO TO 1090
1080 L=100
1090 NEXT L
1100 RETURN
```

Пояснения к программе. Так как внешний цикл организован по L, то, чтобы обеспечить формально выход из этого цикла при окончании просмотра элементов массива A (при K>N), полагается L=100.

При решении задач с использованием массивов следует иметь в виду, что в качестве границ индексов в описании массивов не могут фигурировать переменные, поэтому, чтобы составить программу в общем виде, в программе следует описывать массивы максимальных размеров, какие только могут встретиться при использовании данной программы. Размеры массивов для конкретной задачи обозначаются при помощи переменных и вводятся оператором ввода. Эти переменные используются далее во всей программе для обозначения границ изменения индексов. Это делает программу независимой от данных конкретной задачи и позволяет использовать ее без изменений для обработки массивов различных размеров.

Этот прием рекомендуется использовать во всех случаях, даже когда размеры массивов менять не предполагается. Во-первых, этим облегчается внесение в программу любых изменений, связанных с границами индексов, если это все же понадобится. Во-вторых, отладку программы целесообразно проводить на меньших объемах данных, что легко реализуется, если следовать предлагаемому способу.

Пример. Соревнования по плаванию проводятся отдельно в Европе и в Америке. Результаты 100 лучших спортсменов каждого континента представлены в виде таблиц, содержащих в порядке занятых мест фамилии спортсменов, страны, которые они представляют, и результаты.

Составить список 100 лучших спортсменов мира в порядке, определяемом результатами.

Решение. Результаты по каждому континенту могут быть представлены в виде двух символьных массивов (фамилий и стран) и одного числового (результатов) размером 100. Массивы результатов упорядочены по возрастанию.

Задача сводится к объединению двух упорядоченных массивов результатов в один упорядоченный массив. Массивы фамилий и стран также объединяются в соответствии с результатами.

При решении задачи может быть использован алгоритм объединения двух упорядоченных массивов в один также упорядоченный (программа 4.67) с некоторой модификацией, связанной с необходимостью объединения не одного, а трех массивов, и получением при объединении только первых $M=100$ элементов, так что ситуация, когда один из исходных массивов полностью исчерпан, не возникает и, следовательно, завершение цикла определяется окончанием формирования результирующих массивов (программа 4.69).

*Список используемых переменных.* Исходные данные: A◯, E◯— массивы фамилий, размером N; A1◯, E1◯— массивы представляемых стран, размером N; A2, E2 — упорядоченные массивы результатов, размером N; N — размер исходных таблиц, M — размер результирующей таблицы ($M \leqslant N$).

Результат: W◯, W1◯, W2 — массивы фамилий, стран и результатов лучших спортсменов, размером M.

Вспомогательные переменные: K — индекс — управляющая переменная цикла, I, J — индексы.

Пример. В соревнованиях по прыжкам в длину принимают участие 10 спортсменов. Считая заданным список фамилий спортсменов и их результатов в порядке стартовых номеров, получить и напечатать итоговую таблицу, в которой содержатся фамилии и результаты в порядке занятых мест.

Решение. Исходную информацию введем в два массива: символьный (фамилии) и числовой (результаты). Итоговая таблица должна состоять из трех граф: место, фамилия, результат. Перед выводом итоговой таблицы исходные массивы нужно упорядочить по убыванию результатов (программа 4.70).

*Список используемых переменных.* Исходные данные: A◯— массив фамилий размером 10, R — массив результатов размером 10.

**Программа 4.69**

```
10 DIM A$(100),E$(100),A1$(100),E1$(100)
20 DIM A2(100),E2(100),W$(100),W1$(100),W2(100)
30 PRINT "ВВЕДИТЕ РАЗМЕРЫ ТАБЛИЦ (ИСХОДНОЙ И РЕЗУЛЬТАТОВ)"
40 INPUT N,M
50 PRINT "ВВОД РЕЗУЛЬТАТОВ ПЕРВЕНСТВА ЕВРОПЫ - ФАМИЛИЯ,";
60 PRINT "СТРАНА,РЕЗУЛЬТАТ"
70 FOR I=1 TO N
80 INPUT E$(I),E1$(I),E2(I)
90 NEXT I
100 PRINT "ВВОД РЕЗУЛЬТАТОВ ПЕРВЕНСТВА АМЕРИКИ - ФАМИЛИЯ,";
110 PRINT "СТРАНА,РЕЗУЛЬТАТ"
120 FOR I=1 TO N
130 INPUT A$(I),A1$(I),A2(I)
140 NEXT I
150 GOSUB 220
160 PRINT TAB(10);M;"ЛУЧШИХ СПОРТСМЕНОВ МИРА"
170 PRINT
180 FOR I=1 TO M
190 PRINT TAB(5);W$(I);TAB(30);W1$(I);TAB(50);W2(I)
200 NEXT I
210 STOP
220 REM ПОДПРОГРАММА ОБ'ЕДИНЕНИЯ ТАБЛИЦ
230 I=1 \ J=1 \ K=0
240 IF K>M GO TO 290
250 K=K+1
260 IF A2(I)<E2(J) GO TO 280
270 W2(K)=E2(J) \ W$(K)=E$(J) \ W1$(K)=E1$(J) \ J=J+1 \ GO TO 240
280 W2(K)=A2(I) \ W$(K)=A$(I) \ W1$(K)=A1$(I) \ I=I+1 \ GO TO 240
290 RETURN
```

Результат: A() — массив фамилий, упорядоченный по результатам, R — упорядоченный массив результатов.

Вспомогательные переменные: P — переменная для хранения промежуточного значения максимального элемента, K — индекс максимального элемента, I — индекс элемента упорядоченного массива — управляющая переменная внешнего цикла подпрограммы упорядочения, J — управляющая переменная внутреннего цикла в подпрограмме упорядочения, Q() — переменная, используемая при перестановке фамилий (элементов массива A()).

В программе используется алгоритм упорядочения массива (см. программу 4.64), модифицированный в связи с необходимостью при перестановке результатов (элементов массива R) переставлять также фамилии (элементы массива A()).

После упорядочения место спортсмена определяется индексом соответствующего элемента массива (возможность одинаковых результатов игнорируется).

Задание I уровня. Выбрать способ представления данных и составить список используемых переменных *). Сфор-

*) Под переменными здесь понимаются не только простые переменные, но и массивы.

**Программа 4.70**

```
10 REM ПРОГРАММА СОСТАВЛЕНИЯ ТАБЛИЦЫ РЕЗУЛЬТАТОВ
20 REM СОРЕВНОВАНИЙ ПО ПРЫЖКАМ В ДЛИНУ
30 DATA "АЛЕКСЕЕВ",7.60,"БОРИСОВ",8.10,"ВЛАСОВ",8.25,"ГЛЕБОВ",8.15
40 DATA "ДАВЫДОВ",8.31,"ЕФРЕМОВ",8.16,"ИГНАТЬЕВ",8.34,"КЛИМОВ",8.44
50 DATA "МАКСИМОВ",8.04,"ПЕТРОВ",8.19
60 DIM A$(10),R(10)
70 REM ВВОДИМ ФАМИЛИИ И РЕЗУЛЬТАТ
80 FOR I=1 TO 10 \ READ A$(I),R(I) \ NEXT I
90 GOSUB 1000
100 PRINT TAB(10);"РЕЗУЛЬТАТЫ СОРЕВНОВАНИЙ ПО"
110 PRINT TAB(15);"ПРЫЖКАМ В ДЛИНУ"
120 GOSUB 1100
130 PRINT TAB(5);"I МЕСТО I";TAB(20);"ФАМИЛИЯ";
140 PRINT TAB(35);"I";TAB(38);"РЕЗУЛЬТАТ";TAB(50)"I"
150 GOSUB 1100
160 FOR I=1 TO 10
170 PRINT TAB(5);"I";TAB(8);I;TAB(13);"I";TAB(20);A$(I);
180 PRINT TAB(35);"I";TAB(39);R(I);TAB(47);"M";TAB(50);"I"
190 NEXT I
200 GOSUB 1100
210 STOP
1000 REM ПОДПРОГРАММА УПОРЯДОЧИВАНИЯ МАССИВА
1010 FOR I=1 TO 9
1020 P=R(I) \ K=I
1030 FOR J=I+1 TO 10
1040 IF R(J)<=P GO TO 1060
1050 P=R(J) \ K=J
1060 NEXT J
1070 R(K)=R(I) \ R(I)=P \ Q$=A$(I) \ A$(I)=A$(K) \ A$(K)=Q$
1080 NEXT I
1090 RETURN
1100 REM ПОДПРОГРАММА "ГОРИЗОНТАЛЬНАЯ ЛИНИЯ"
1110 PRINT TAB(5);"-";
1120 FOR K=1 TO 45 \ PRINT "-"; \ NEXT K \ PRINT
1130 RETURN

RUNNH

          РЕЗУЛЬТАТЫ СОРЕВНОВАНИЙ ПО
              ПРЫЖКАМ В ДЛИНУ
 ---------------------------------------------
 I МЕСТО I      ФАМИЛИЯ        I  РЕЗУЛЬТАТ   I
 ---------------------------------------------
 I   1   I      КЛИМОВ         I    8.44   M  I
 I   2   I      ИГНАТЬЕВ       I    8.34   M  I
 I   3   I      ДАВЫДОВ        I    8.31   M  I
 I   4   I      ВЛАСОВ         I    8.25   M  I
 I   5   I      ПЕТРОВ         I    8.19   M  I
 I   6   I      ЕФРЕМОВ        I    8.16   M  I
 I   7   I      ГЛЕБОВ         I    8.15   M  I
 I   8   I      БОРИСОВ        I    8.1    M  I
 I   9   I      МАКСИМОВ       I    8.04   M  I
 I   10  I      АЛЕКСЕЕВ       I    7.8    M  I
 ---------------------------------------------

STOP AT LINE 210
```

мулировать задачу математически. Определить, какой из типовых алгоритмов, описанных в теоретическом введении к работе 7 или к работе 4, может быть использован для решения задачи. Конкретизировать алгоритм применительно к задаче. Составить схему и программу. Подготовить тесты. Проверить работу программы на ЭВМ.

*Варианты задач I уровня.*

1. Информация о количестве выпадавших в течение месяца осадков задана в виде массива. Определить общее количество осадков за месяц.

2. Информация о температуре воздуха за месяц задана в виде массива. Определить, сколько раз температура опускалась ниже 0 °С.

3. Информация о средней суточной температуре воздуха за месяц задана в виде массива. Определить, температура скольких дней была ниже среднемесячной.

4. Регистрация направления ветра на горном плато производится один раз в день по очереди двумя исследователями. Каждый месяц все результаты сводятся в одну таблицу. Составить программу, выполняющую эту операцию.

Указание. Направление ветра кодируется следующим образом:

| | |
|---|---|
| 1 — северный | 5 — северо-западный |
| 2 — южный | 6 — северо-восточный |
| 3 — восточный | 7 — юго-западный |
| 4 — западный | 8 — юго-восточный |

5. Информация о количестве осадков, выпадавших в течение месяца, и о температуре воздуха задана в виде двух массивов. Определить, какое количество осадков выпало в виде дождя, какое в виде снега. (Считать, что идет дождь, если температура воздуха $>0$ °С).

6. Рост учеников класса представлен в виде массива. Рост девочек кодируется знаком +, рост мальчиков знаком —. Определить средний рост мальчиков.

7. В области 10 районов. Известны площади, засеваемые пшеницей, и средняя урожайность (*ц/га*) в каждом районе. Определить количество пшеницы, собранное в области, и среднюю урожайность по области.

8. В области 10 районов. Заданы площади, засеваемые в каждом районе пшеницей, и урожай, собранный в каждом районе. Определить среднюю урожайность пшеницы по каждому району и по области в целом.

9. Результаты переписи населения хранятся в памяти ЭВМ. Используя массивы фамилий и года рождения, напечатать фамилии и подсчитать общее число жителей, родившихся раньше 1928 года.

10. Ртутные термометры могут использоваться для измерения температуры до —39,4 °С. Используя информацию о минимальной температуре, зафиксированной в каждом году из последних 100 лет в г. Воронеже, определить, можно ли поставлять ртутные термометры в этот город.

11. Задан список участников соревнований по плаванию и их результаты. Напечатать фамилию и результат чемпиона.

12. В памяти ЭВМ хранится информация о валовом сборе зерна по союзным республикам отдельно за 1984 и 1985 гг. Определить суммарный сбор зерна по каждой республике за 2 года.

13. В задаче 11 расположить результаты и фамилии участников в соответствии с занятыми местами.

14. Фамилии участников соревнований по фигурному катанию после короткой программы расположены в порядке, соответствующем занятому месту. Составить список участников в порядке их стартовых номеров для произвольной программы (участники выступают в порядке, обратном занятым местам).

15. При поступлении в институт лица, получившие двойку на первом экзамене, ко второму экзамену не допускаются. Считая фамилии абитуриентов и их оценки после первого экзамена исходными данными, составить список абитуриентов, допущенных ко второму экзамену.

Указания к решению задач I уровня.

1. Задача сводится к суммированию элементов массива. Максимальный размер массива — 31 (наибольшее число дней в месяце). Количество дней конкретного месяца ввести оператором ввода (при желании можно количество дней в месяце определять программно по его названию).

2. См. указание к задаче 1.

3. Задача сводится к комбинации двух алгоритмов: суммирование элементов массива для определения среднемесячной температуры и определение количества элементов массива, удовлетворяющих заданному условию. См. также указание к задаче 1.

4. Задача сводится к объединению двух массивов в один с чередованием элементов исходных массивов. См. также указание к задаче 1.

5. Задача сводится к суммированию элементов массива осадков в двух переменных. Элемент массива осадков должен прибавляться к одной из двух сумм в зависимости от значения соответствующего элемента массива температур. См. также указание к задаче 1.

6. Задача сводится к вычислению суммы и количества отрицательных чисел в массиве, после окончания суммирования знак суммы нужно изменить.

7. Площади, засеваемые пшеницей, и урожайность для каждого района нужно задать в двух массивах. Количество пшеницы, со-

бранное в области, вычисляется как сумма произведений элементов двух массивов, а для получения средней урожайности нужно вычисленное общее количество пшеницы разделить на сумму элементов массива площадей.

8. Исходные данные — урожай в каждом районе и засеваемые площади — задать в двух массивах. Для определения урожайности по каждому району найти частное от деления элементов массивов, содержащих исходные данные. Для определения средней урожайности по области найти частное от деления сумм элементов массивов исходных данных.

9. Задача сводится к определению числа элементов массива года рождения со значениями меньше 1928 и вывода на печать соответствующих элементов массива фамилий.

10. Задача сводится к определению того, есть ли в массиве минимальных температур хотя бы один элемент, меньший или равный —39,4.

11. Задача сводится к определению минимального элемента массива результатов и его индекса, и выводу на печать элемента массива фамилий с этим индексом.

12. Задача сводится к суммированию двух массивов.

13. Задача сводится к упорядочению массива результатов в порядке возрастания, но при выполнении перестановки в массиве результатов нужно переставлять соответствующие элементы и в массиве фамилий.

14. Задача сводится к инвертированию массива фамилий участников соревнований.

15. Задача сводится к формированию массива из заданного включением в него только элементов, удовлетворяющих заданному условию (условие проверяется для соответствующего элемента массива оценок).

Задание II уровня. Выбрать целесообразный способ представления данных, используя, если нужно, и двумерные массивы. Составить список используемых переменных. Сформулировать задачу математически. Разработать алгоритм решения задачи, используя, если возможно, типовые алгоритмы, изложенные в теоретическом введении. Решение многих задач не может быть полностью сведено к типовым алгоритмам и требует творческого подхода.

Составить схему и программу. Особое внимание обратить на вывод результатов в наглядной форме — с подробными пояснениями, если можно — в виде таблиц. Выполнить программу вручную для некоторых наборов данных. Проверить работу программы на ЭВМ.

*Варианты задач II уровня.*

1. При выборе места строительства жилого комплекса при ме-

таллургическом заводе необходимо учитывать «розу ветров» в данной местности. На основании данных ежедневного определения направления ветра, проводимого в течение года, определить целесообразное взаимное расположение промышленной жилой зоны (кодирование направлений ветра задано в задаче 4 уровня I).

2. Известно, что в Москве самыми теплыми являются дни с 15 июля по 15 августа. Для проведения фестиваля были выбраны 7 следующих подряд дней, наиболее теплых по данным за последние 10 лет. Составить программу для выполнения этой работы на ЭВМ.

3. Составить результирующую таблицу первенства по футболу, в котором участвуют 10 команд. В качестве исходной информации задан счет: количество забитых (пропущенных) мячей в каждой проведенной игре. Для получения итогового результата необходимо по заданной таблице забитых (пропущенных) мячей составить таблицу очков (выигрыш — 2, ничья — 1, проигрыш — 0). Далее определить сумму очков для каждой команды и в соответствии с этим распределить команды по местам. Если сумма очков у двух команд одинакова, то сравниваются разности забитых и пропущенных мячей.

4. Для формирования сборной страны по хоккею предварительно выбрано 30 игроков. На основании протоколов игр (всего 10 игр) составлена таблица, в которой содержится штрафное время каждого игрока по каждой игре (штрафное время может составлять 2, 5 или 10 мин). Составить программу, которая составляет предварительный список кандидатов в сборную и определяет для каждого из них суммарное штрафное время. Игроки, оштрафованные хотя бы один раз на 10 мин, из кандидатов в сборную исключаются.

5. В задаче 3 определить, в скольких играх разность забитых и пропущенных мячей была большей или равной 3. Какая команда имеет наибольшее количество побед с таким крупным счетом?

6. Составить программу для ведения протокола баскетбольной игры. Во время игры машина ведет учет набранных очков и фолов каждого игрока. Игрок, получивший 5 фолов, удаляется из игры (эта информация должна появляться на экране). В конце игры должна выводиться информация о сумме очков, набранных каждым игроком, в порядке убывания.

7. Составить программу для обработки результатов кросса на 500 м для женщин. В кроссе участвует не более 100 студенток. Для каждой участницы ввести фамилию, шифр группы, фамилию преподавателя, результат. Получить результирующую таблицу, упорядоченную по результатам, в которой содержится также информация о выполнении нормы ГТО. Определить суммарное количество студенток, выполнивших норму ГТО.

8. Для шифрования используется смешанный алфавит, полу-

ченный случайной перестановкой букв исходного алфавита. Например:

АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
ШЗУКЛЖЩАЯЭЮФЪДВБПГЧЫЬЕОИМРХНЦСТЙ

Для усложнения шифра вместо букв используются целые числа (А соответствует 1, Б — 2 и т. д.), и для кодирования каждой следующей буквы сообщения используют новый смешанный алфавит, полученный из исходного циклической перестановкой букв вправо на заданное число позиций. Число позиций, на которые осуществляется сдвиг, и частота их повторения известны заранее. Пусть, например, это числа 5, 3, 1, 8. Тогда для кодирования первой буквы будет использован алфавит, полученный из исходного циклическим сдвигом на 5 позиций вправо, второй буквы — на 3 позиции, третьей — на 1, четвертой — на 8, пятой — снова на 5 и т. д. Составить программу, которая подготавливает таблицу для шифрования, содержащую четыре необходимых алфавита. Считать, что исходный алфавит задан в числовом виде.

9. Результаты соревнований фигуристов по одному из видов многоборья представлены оценками судей в баллах (от 0 до 6). По результатам оценок судьи определяется место каждого участника у этого судьи. Места участников определяются далее по сумме мест, которые каждый участник занял у всех судей. Составить программу, определяющую по исходной таблице оценок фамилии и сумму мест участников в порядке занятых ими мест. Число участников не более 15, число судей не более 10.

10. Пластинка размером $30\times50$ см$^2$ разделена на квадраты размером $2\times2$ см$^2$. Произведено измерение температуры каждого квадрата. Используя эту информацию, построить изотермы при помощи символов *, +, О, Р, %, #, разделив интервал изменения температуры на 6 равных интервалов и поставив в соответствие каждому подынтервалу один из указанных символов.

11. Японская радиокомпания провела опрос 250 радиослушателей по вопросу: «Какое животное Вы связываете с Японией и японцами?». Составить программу получения пяти наиболее часто встречающихся ответов и их долей (в %).

12. Траектории движения двух самолетов заданы уравнениями $x_i=x_i(t)$, $y_i=y_i(t)$, $z_i=z_i(t)$, $i=1, 2$. Определить, в какой момент времени расстояние между самолетами будет минимальным. Рассмотреть интервал времени между двумя заданными моментами. Время определить с точностью 1/50 исходного интервала.

13. Сформировать из матрицы $A(10, 10)$ матрицу $B(10, 10)$ по следующим правилам:

*а)* элементы матриц $A$ и $B$ принимают только значения 0 или 1;

*б)* соседями элемента $a_{ij}$ считаются все элементы, расположенные рядом с данным по горизонтали, вертикали или диагонали;

*в)* если сумма $S$ значений соседей элемента $a_{ij}$ меньше двух или больше трех, то $b_{ij}=0$;

*г)* если сумма $S$ значений соседей элемента $a_{ij}$ равна двум, то $b_{ij}=a_{ij}$;

*д)* если сумма $S$ значений соседей элемента $a_{ij}$ равна трем, то $b_{ij}=1$.

По окончании формирования матрицы $B$ значения элементов построчно вывести на печать, заменяя 0 — символом _, 1 — символом *.

Это преобразование является основой так называемой игры «Жизнь» (см., например, Уэзерелл Ч. Этюды для программистов.— М.: Мир, 1982), позволяющей из исходного получить новый вариант расположения в матрице нулей и единиц (или при выводе в символьном виде — конфигурацию звездочек).

14. Составить программу — тест «Решительны ли Вы?» (см. «Знание — сила» — 1983, № 11).

На экране по очереди появляются 12 вопросов. На каждый из них нужно ввести ответ (*да* или *нет*). В зависимости от вопроса и ответа к общей сумме прибавляется определенное число очков. После окончания опроса по общей сумме очков делается заключение по поводу решительности опрашиваемого. В программе предусмотреть защиту от неправильного ответа.

Исходные данные для программы:

1). Сможете ли Вы на старом месте работы приспособиться к новым правилам?

2). Быстро ли адаптируетесь в новом коллективе?

3). Способны ли высказать свое мнение публично?

4). Согласитесь ли Вы без колебаний перейти в другое учреждение на должность с более высоким окладом?

5). Склонны ли отрицать свою вину в допущенной ошибке и искать оправдания?

6). Объясняете ли Вы обычно причины своего отказа от чего-либо истинными мотивами или стараетесь их завуалировать?

7). Сможете ли изменить свой прежний взгляд или убеждения в результате серьезной дискуссии?

8). Если Вам не нравится стиль изложения работы, с которой Вы знакомитесь, будете ли Вы править текст и настойчиво предлагать изменить его?

9). Купите ли Вы вещь, которая Вам очень нравится, но не так уж необходима?

10). Можете ли изменить свое решение под влиянием уговоров обаятельного человека?

11). Планируете ли заранее свой отпуск?

12). Всегда ли выполняете данное Вами обещание?

Таблица для определения набранных очков

| Вопрос | Да | Нет | Вопрос | Да | Нет |
|---|---|---|---|---|---|
| 1 | 3 | 0 | 7 | 3 | 0 |
| 2 | 4 | 0 | 8 | 2 | 0 |
| 3 | 3 | 0 | 9 | 0 | 2 |
| 4 | 2 | 0 | 10 | 0 | 3 |
| 5 | 0 | 4 | 11 | 1 | 0 |
| 6 | 2 | 0 | 12 | 3 | 0 |

Определение результата тестирования:

1). От 0 до 9 очков — Вы очень нерешительны. Старайтесь изменить свой характер.

2). От 10 до 18 очков — Вы принимаете решения осторожно, но не пасуете перед серьезными проблемами. Больше полагайтесь на свой опыт.

3). От 19 до 28 очков — Вы достаточно решительны. Принятые решения отстаиваете до конца. Старайтесь быть объективным. Консультируйтесь по вопросам, в которых недостаточно компетентны.

4). От 29 очков и выше — Вы принимаете решения единолично, не считаясь с мнением других. Это подавляет инициативу подчиненных, наносит ущерб психологическому климату коллектива. Нужно срочно меиять стиль работы.

15. Составить программу для контроля знаний по курсу «Химия», раздел: «Номенклатура неорганических веществ».

В памяти ЭВМ хранятся вопросы и правильные ответы. Вопросы высвечиваются на экране по очереди, и на каждый вопрос обучаемый вводит ответ, который сравнивается с правильным. После ответа на все вопросы предусмотреть выставление оценки.

Указания к решению задач II уровня.

1. Задача сводится к определению того, какое значение чаще всего встречается в массиве направлений ветра. Для ее решения нужно сформировать массив размером 8 (индекс совпадает с кодом направления), каждый элемент которого является количеством элементов массива направлений ветра, имеющих заданное значение, а затем определить максимальный элемент этого массива и соответствующий ему индекс.

2. Исходные данные целесообразно представить в виде двумерного массива 10×32. По исходному массиву нужно далее сформировать массив 10×26, каждый элемент которого — сумма температур за каждые 7 из 32 дней. Далее нужно просуммировать

столбцы этого массива и определить, для какого столбца сумма элементов является максимальной.

3. Предусмотреть числовой массив размером 10×10 для количества забитых (пропущенных) мячей по каждой игре и символьный массив размером 10 для названий команд. Количество мячей, забитых $i$-й командой $j$-й команде, следует помещать в $j$-й столбец $i$-й строки. Для формирования результирующей таблицы суммировать строки массива очков. Полученный в результате одномерный массив нужно упорядочить по убыванию, меняя при этом и названия команд. Полученный упорядоченный массив нужно проверить на совпадение его элементов. В случае равенства очков по названию команд следует отыскать соответствующие строки в матрице забитых (пропущенных) мячей и найти разности сумм забитых и пропущенных мячей. (Для получения суммы мячей, забитых $i$-й командой, необходимо вычислить сумму элементов $i$-й строки, а пропущенных $i$-й командой — сумму элементов $i$-го столбца.) В случае необходимости соседние команды поменять местами. Вычисленную худшую разность сумм забитых и пропущенных мячей нужно сохранять на тот случай, если следующая сумма очков совпадает с двумя предыдущими.

4. Исходную информацию целесообразно хранить в виде двумерного массива 30×10. Предусмотреть также массив размером 30 для фамилий игроков. Для решения задачи нужно суммировать строки массива, сравнивая при этом значение каждого слагаемого с 10. Если найдено такое слагаемое, то соответствующая фамилия в результирующий список не включается и суммирование строки дальше не производится.

5. См. указание к задаче 3. Для решения задачи можно изменять индексы только одного треугольника матрицы (например, лежащего выше главной диагонали), а при вычислении разности мячей вычитаемое должно извлекаться из симметричного элемента другого треугольника (для этого индексы нужно поменять местами). Если разность мячей по модулю больше или равна 3, то нужно увеличить результат (количество игр с крупным счетом) на 1. Для решения второй части задачи предусмотреть массив размером 10. Для игры с крупным счетом надо увеличивать на 1 элемент этого массива с индексом, равным номеру строки исходного массива, если разность мячей положительна, или номеру столбца, если разность мячей отрицательна.

6. Нужно предусмотреть три массива для фамилий игроков, числа фолов и очков каждого игрока. При окончании ввода данных (окончании игры) предусмотреть ввод специального значения (см. работу 3). При увеличении числа фолов оно каждый раз должно сравниваться с 5. После окончания игры нужно упорядочить массив очков (меняя при этом местами и фамилии игроков).

7. Использовать алгоритмы, излагаемые в пп. 5, 8 теоретического введения.

8. Исходный алфавит задан в виде массива размером 32, элементами которого являются целые числа от 1 до 32 в произвольном порядке. Результат получить в виде двумерного массива размером $4\times32$, в котором каждая строка получена циклической перестановкой исходного массива на заданное количество позиций.

9. Таблицу оценок (массив размером не менее, чем $10\times15$) преобразовать в таблицу мест. Просуммировать элементы ее столбцов. Полученный массив упорядочить по возрастанию, переставляя одновременно элементы массива фамилий участников (размером не менее, чем 15).

10. Исходная информация (температура каждого квадрата $2\times2$) может быть представлена в виде двумерного массива размером $15\times25$. Этот двумерный числовой массив нужно преобразовать в символьный массив такого же размера и напечатать его. Для выполнения этого преобразования нужно:

— найти значения минимального и максимального элементов исходного массива для определения интервала изменения температур;

— разделить этот интервал на 6 для определения шести подынтервалов;

— сформировать символьный массив, просматривая элементы исходного массива и определяя, какому из шести подынтервалов принадлежит элемент этого массива;

— вывести на печать полученный символьный массив.

11. Исходную информацию нужно разместить в символьном массиве размером не менее 250. Далее сформировать два массива — символьный, в который нужно поместить все отличающиеся друг от друга ответы, и числовой, в котором подсчитывается частота их появления в исходном массиве. Упорядочить массив частот (переставляя при этом и элементы массива ответов). Определить доли первых пяти ответов в % и напечатать их вместе с соответствующими ответами.

Замечание. Если для упорядочения используется алгоритм, изложенный в п. 8, то упорядочение можно прекратить, как только будут сформированы первые пять элементов упорядоченного массива.

12. Расстояние $L$ между самолетами в каждый момент времени вычисляется как $L=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$. Для решения задачи нужно сформировать массив размером 50, вычисляя расстояние между самолетами с шагом по времени в 1/50 исходного интервала. Далее нужно найти минимальный элемент этого массива. Функции $x_i(t)$, $y_i(t)$, $z_i(t)$, $i=1, 2$, задать самостоятельно. Для их определения в программе использовать оператор DEF FN.

13. Для решения задачи нужно, заполнив предварительно нулями матрицу $B$, просматривать элементы матрицы $A$ и для каждого определять сумму элементов, являющихся ее соседями. В зависимости от результата присваивать элементу массива $B$ соответствующее значение (значение 0 присваивать не нужно). Перед выводом на печать преобразовать матрицу $B$ к символьному виду. Чтобы просмотр элементов матрицы $A$ и суммирование ее соседей не зависело от того, находится элемент на границе или со всех сторон имеет соседей, можно работать с расширенной матрицей $12\times12$. Элементам, расширяющим матрицу ($i=1, 12$; $j=1, 12$), значений не присваивать, они всегда имеют значения 0.

14. Для исходных данных предусмотреть символьный массив вопросов, два числовых массива очков, начисляемых в случае ответов *да* и *нет*, а также символьный массив размером 4 с текстами, выводимыми на экран в зависимости от набранной суммы очков.

15. Вопросы (например, «Название вещества $Ca(HCO_3)_2$?») и правильные ответы на них (например, «Гидрокарбонат кальция») помещаются в два символьных массива. В третий символьный массив можно поместить комментарий, который будет высвечиваться в случае неправильного ответа (например, «Повторите раздел «Номенклатура солей»). Причем, поскольку слова «Название вещества» в каждом вопросе повторяются, их можно в явном виде указать в операторе PRINT, а в массив вопросов поместить только название вещества.

Вопросы (из первого массива) высвечиваются по очереди. Обучаемый вводит ответ в символьную переменную. Этот ответ сравнивается с правильным ответом (соответствующим элементом второго массива). Результат сравнения можно высветить на экране вместе с комментарием.

В программе предусмотреть определение количества правильных ответов и выставление оценки после ответа на все вопросы. По количеству вопросов N и количеству правильных ответов P оценку можно вычислить следующим образом:

```
M=INT(P/N*5+0.9)
```

и, если $M<3$, положить $M=2$.

Исходные данные для работы программы (массивы вопросов, ответов и комментарии) целесообразно разместить в программе (в операторах DATA: в одном операторе DATA — вопрос, ответ и комментарий).

Задание III уровня. Решение задач III уровня требует использования подпрограмм. Требуется также выполнить все пункты задания II уровня.

*Варианты задач III уровня.*

1. Составить программу для обработки результатов соревнований по фигурному катанию, проводившихся по трем видам многоборья (обязательная, короткая и произвольная программы).

Обработку результатов по каждому из видов (см. задачу 9 уровня II) осуществлять в подпрограмме.

2. Японская радиокомпания провела опрос 250 радиослушателей по трем вопросам:

1). Какое животное Вы связываете с Японией и японцами?

2). Какая черта характера присуща японцам больше всего?

3). Какой неодушевленный предмет или понятие Вы связываете с Японией?

Большинство опрошенных прислали ответы на все или часть вопросов. Составить программу получения первых пяти наиболее часто встречающихся ответов по каждому вопросу и доли (в %) каждого такого ответа. Предусмотреть необходимость сжатия столбца ответов в случае отсутствия ответов на некоторые вопросы.

Обработку информации по каждому вопросу осуществлять в подпрограмме.

3. Соревнования по футболу между командами проводятся в два этапа. Для проведения первого этапа участники разбиваются на две группы по 12 команд. Для проведения второго этапа выбирается 6 лучших команд каждой группы по результатам первого этапа. Составить список команд участников второго этапа.

Определение результатов соревнования внутри одной группы осуществлять в подпрограмме. Задаваемые исходные данные и правила определения результатов соревнований приведены в задаче 3 уровня II.

4. Чемпионат города по футболу проводится по двум группам: сильная — 12 команд и слабая — 12 команд. Замещение мест в сильной группе проводится по результатам переходного турнира, в котором участвуют 3 худших команды из сильной группы и 3 лучших — из слабой. Составить список команд — участниц переходного турнира.

Обработку результатов соревнований в одной группе (см. задачу 3 уровня II) осуществлять в подпрограмме.

5. В соревнованиях участвуют три команды по 6 человек. Результаты соревнований в виде мест участников каждой команды (1—18) размещены в трех массивах, содержащих по 6 компонент. Определить команду — победителя, вычислив количество баллов, набранное каждой командой. Участнику, занявшему 1-е место, начисляется 5 баллов, 2-е — 4, 3-е — 3, 4-е — 2, 5-е — 1, остальным — 0 баллов.

Определение числа баллов, набранных одной командой, осуществлять в подпрограмме.

6. В памяти ЭВМ хранится информация о засеваемых площадях и урожае зерновых по районам для 10 областей (в каждой области не более 10 районов). Определить среднюю урожайность по каждой области и область, добившуюся наибольшей урожайности.

Определение средней урожайности одной области выполнять в подпрограмме.

7. Информация о результатах сессии (5 экзаменов) по каждой группе хранится в памяти ЭВМ. Для подведения итогов социалистического соревнования определить средний балл для пяти групп одного потока студентов и выдать список групп в порядке убывания среднего балла.

Определение среднего балла одной группы осуществлять в подпрограмме.

8. Соревнования (лыжные гонки) проводятся двумя группами по 10 человек. Результаты соревнований представлены списками участников и их результатов по каждой группе. Предварительное подведение итогов проводится по каждой группе, результатом его являются списки участников по группам в порядке занятых ими мест. Необходимо получить общий список, в котором участники расположены в порядке, соответствующем показанным результатам.

Определение результатов по каждой группе осуществлять в подпрограмме.

9. Написать программу построения латинского квадрата. Латинский квадрат — это матрица $N \times N$, элементы которой выбраны от 1 до $N$ так, что каждое число встречается только один раз в каждой строке и в каждом столбце.

Циклический сдвиг (см. указание к задаче) осуществляется в подпрограмме.

10. В задаче 13 уровня II исходную конфигурацию задать как десять символьных строк, состоящих из символов _ и *. Все преобразования осуществлять с числовой матрицей. Проследить за развитием первоначальной конфигурации в нескольких поколениях. Каждую новую конфигурацию и ее номер выводить на экран. Предусмотреть возможность прекращения работы программы по желанию пользователя. Преобразование из числового представления в символьное и вывод на печать очередной конфигурации осуществлять в подпрограмме.

11. Имеется $n$ сосудов, в каждом из которых лежит по одному камню белого, синего или красного цвета. «Заглядывая» в каждый сосуд по одному разу, расположить их в таком порядке, чтобы в первой группе сосудов лежали только красные, затем — синие, далее — белые камни (этот порядок цветов соответствует голландскому национальному флагу).

12. В задаче 15 уровня II предусмотреть возможность выбора

случайным образом части вопросов, используемых для контроля. Для выставления оценки считать, что вопросы имеют разные веса по важности.

Формирование номеров вопросов для контроля одного обучаемого осуществлять в подпрограмме.

13. Составить программу для контроля знаний. В программе задается один вопрос, ответ на который включает несколько наименований (например, «Назовите все элементы Периодической системы, представляющие группу галогенов» или «Назовите все города с населением свыше 1 млн» и т. п.). В памяти ЭВМ хранится список наименований, являющийся полным ответом на вопрос. Введенный ответ необходимо сравнить с правильным.

14. В памяти ЭВМ хранится список фамилий абонентов в алфавитном порядке и номеров их телефонов. Составить программу, обеспечивающую быстрый поиск фамилии абонента по номеру телефона.

15. В памяти ЭВМ хранятся списки номеров телефонов и фамилий абонентов, упорядоченные по номерам телефонов, для каждого из пяти телефонных узлов города. Один телефонный узел включает несколько АТС (не более 10). Номера АТС (первые две цифры номера телефона), относящихся к каждому телефонному узлу, также хранятся в памяти ЭВМ. Составить программу, обеспечивающую быстрый поиск фамилии абонента по заданному номеру телефона *).

Указания к решению задач III уровня.

1. Результаты соревнований по каждому виду в виде числа участников (некоторые из участников могут выбыть), их фамилий и мест судей вводить в память по очереди (можно в один и тот же массив). Результаты обработки по каждому виду хранить в памяти для получения окончательного результата. Окончательный результат получать только для участвовавших во всех трех видах. Подпрограмму составить в общем виде для произвольного числа участников и судей (в пределах задаваемых ограничений). См. также указание к задаче 9 уровня II.

2. Исходную информацию задать в виде двумерного массива размером 250×3. Перед обращением к подпрограмме, обрабатывающей ответы на один вопрос, переслать столбец исходного массива в одномерный массив, используемый в подпрограмме (пустые ответы не пересылать). В результате пересылки нужно определить также количество непустых ответов, чтобы использовать его в подпрограмме и печатать в качестве итоговой информации.

5. Исходную информацию задать в виде двумерного массива размером 3×6 и массива названий команд. При обработке резуль-

*) Количественные данные по телефонной сети не относятся к г. Москва.

татов одной команды использовать то соображение, что сумма номера места, если оно $\leqslant 6$, и количества очков равна 6.

8. Для решения задачи нужно упорядочить два массива результатов и объединить их в один так же упорядоченный массив. Одновременно нужно объединить и массивы фамилий (в соответствии с результатами).

9. Первую строку матрицы заполнить цифрами от 1 до $N$, каждую следующую строку получать, циклически сдвигая предыдущую на 1 элемент вправо (влево). Очередную строку матрицы получать в виде одномерного массива в подпрограмме, используя массив, сформированный при предыдущем обращении, как исходный при выполнении следующего.

10. Исходную конфигурацию задать как символьный массив размером $10\times 10$. В программе нужно предусмотреть переменную, в которую перед получением очередной конфигурации можно вводить специальное значение для прекращения работы программы (см. теоретическое введение к работе 3).

11. Цвета камней можно представить либо символами, например, К, В, С, либо закодировать их числами, например, 1 — красный, 2 — белый, 3 — синий. Просматривая элементы массива (сосуды), начиная с первого, накапливать количество встретившихся синих камней (например, в переменной С) и белых (например, в переменной В) и сразу ставить их в нужном порядке. Для этого, если в I-м ($I>1$) сосуде — С, то его нужно поменять местами с первым из В (элемент с индексом (I — В)). Если в I-м сосуде — К, то его нужно поменять местами сначала с первым из В (элемент с индексом (I — В)), а затем с первым из С (т. е. поменять местами (I — В)-й и (I — В — С)-й элементы). Перестановку элементов осуществлять в подпрограмме.

12. См. указание к задаче 15 уровня II. Для выбора случайным образом заданного числа (К) вопросов можно использовать алгоритм получения К различных случайных целых чисел из интервала [А; В], где $A=1$, $B=N$, N — общее число вопросов (см. теоретическое введение к работе 12, программа 5.11).

Номера вопросов, которые будут предложены одному обучаемому, из общего списка вопросов целесообразно поместить в массив, например, D.

Веса вопросов хранить в массиве, например, V. Для получения оценки суммировать веса правильных ответов ($P=P+V(I)$, где $I=D(J)$ — номер вопроса из общего списка, J — порядковый номер задаваемого вопроса). Далее пронормировать эту сумму, разделив ее на общий вес всех правильных ответов $P=P\Big/\sum_{J=1}^{K} V(D(J))$.

Оценку можно вычислять, например, по формуле M=INT(P*5+0.9), и если M<3, то положить M=2.

18. Список наименований, являющийся полным ответом на вопрос, хранить в символьном массиве. Ответы обучаемого вводить в другой символьный массив.

Для повышения эффективности программы можно правильные ответы ввести в массив упорядоченными (или упорядочить сразу после ввода), а после ввода ответов, перед сравнением, упорядочить и массив ответов. Для сравнения использовать алгоритм сравнения двух упорядоченных массивов.

Алгоритмы упорядочения массивов и сравнения двух упорядоченных массивов оформить в виде подпрограмм.

14. Переформировать исходные списки, упорядочив их по номерам телефонов. Для поиска нужной фамилии использовать алгоритм бинарного поиска в массиве номеров телефонов.

Алгоритм упорядочения оформить в виде подпрограммы.

15. Список номеров телефонов хранить в двухмерном массиве (например, T), содержащем пять строк (одна строка относится к одному узлу). Список фамилий — в двухмерном символьном массиве (например, F$) того же размера. Список АТС, сгруппированный по телефонным узлам, хранить в одномерном массиве (например, A). Индексы начала списка по каждому узлу хранить в отдельном массиве, например, C.

Решение задачи включает следующие шаги:

1). Выделить первые две цифры в заданном номере телефона (см. теоретическое введение к работе 11), т. е. определить номер АТС.

2). Определить индекс элемента массива A, содержащего этот номер.

3). Определить, к какому узлу относится АТС. Для этого определить, к какому из интервалов изменения индексов в массиве A (номер интервала — номер узла) относится найденный индекс. (Использовать массив C.)

4). В строке массива T, относящейся к этому узлу, найти элемент, совпадающий с заданным номером телефона, используя алгоритм бинарного поиска.

5). Напечатать соответствующий элемент массива F$.

Вопросы для самопроверки.

1. Какими способами можно задать исходные данные для программы? Операторы READ, DATA. В каких случаях целесообразно их использование?

2. Отладка и тестирование программы. Средства отладки в бейсике. Что такое тест?

3. Алгоритм определения числа и суммы элементов, удовлетворяющих заданному условию. Какие при этом используются типовые структуры алгоритмов?

4. Алгоритм объединения двух массивов одинакового размера в один с чередованием элементов. Какова связь между индексами исходного массива и формируемого?

5. Алгоритм инвертирования массива. Что является управляющей переменной цикла и в каких пределах она изменяется?

6. Алгоритм формирования массива из элементов другого массива, удовлетворяющих заданному условию. Индекс элементов какого массива является управляющей переменной цикла? Как определить количество элементов в сформированном массиве? Почему индекс для формируемого массива целесообразно получать перед пересылкой очередного элемента в этот массив?

7. Алгоритм циклического сдвига элементов массива: а) с использованием одной вспомогательной переменной; б) с использованием вспомогательного массива. В чем достоинства и недостатки этих способов? Почему при циклическом сдвиге вправо перемещение элементов массива в «хвост» необходимо осуществлять с конца?

8. Алгоритм упорядочения массива. Почему при перестановке элементов массива не требуется дополнительная переменная? В каких пределах изменяется управляющая переменная внешнего и внутреннего циклов и почему? Какие изменения необходимо внести в программу, если при перестановке элементов упорядочиваемого массива нужно точно так же переставлять элементы другого массива?

9. Алгоритм проверки массива на упорядоченность. Что является результатом этого алгоритма? Как формально обеспечить завершение цикла, если условие упорядоченности для какой-либо пары элементов не выполняется?

10. Алгоритм поиска заданного элемента в упорядоченном массиве (бинарный поиск). В чем преимущество бинарного поиска?

11. Алгоритм объединения двух упорядоченных массивов в один так же упорядоченный. Какие типовые структуры алгоритмов здесь используются? Почему целесообразно использование циклов *Пока*?

12. Алгоритм определения количества совпадающих элементов двух упорядоченных массивов. Какие типовые структуры алгоритмов здесь используются? Какая переменная является управляющей переменной цикла? Как организовать завершение цикла, если один из массивов полностью исчерпан?

13. Как составить программу в общем виде для обработки массивов произвольного размера? Какие преимущества дает использование переменных для обозначения размеров массивов?

## *Работа 9.* ОБРАБОТКА ТЕКСТОВЫХ ДАННЫХ

Теоретическое введение. Помимо обработки числовой информации (см. работы 1—8), на ЭВМ можно осуществлять также обработку и анализ текстов. Тексты могут задаваться при помощи символьных переменных и массивов (см. пп. 6, 7 главы 3).

Обработка текстов имеет некоторое сходство с обработкой массивов. Символы в тексте (как и элементы в массиве) упорядочены по номеру позиции, которую они занимают, и переход к следующему символу легко осуществляется изменением номера позиции на 1.

Поэтому многие типовые алгоритмы обработки массивов (см. работы 4, 8) в несколько модифицированном виде могут использоваться и при обработке текстов.

При работе с текстами можно выбирать по одному символу, изменяя номер позиции на 1, а можно сразу выбрать цепочку следующих подряд символов любой длины. Поэтому шаг просмотра текста может быть переменным (и, в частности, зависеть от промежуточных результатов). В связи с этим при составлении программ для обработки текстов не всегда целесообразно использование операторов цикла FOR — NEXT, требующих использования заранее определенного шага.

Для работы с текстами определены также некоторые специальные операции и функции (см. п. 6 главы 3).

Тексты в общем случае могут содержать и числовую информацию, заданную в символьной форме. Чтобы иметь возможность работать с этой информацией как с числами, нужно преобразовать ее к числовой форме при помощи функции VAL.

В работе 9 требуется выполнить преобразование или анализ текста, используя, если можно, типовые алгоритмы обработки массивов, а также определенные для символьных переменных операции и функции.

Типовыми действиями, которые необходимо выполнять при работе с текстами, являются следующие:

— выделить $i$-й символ из текста;

— найти позицию (позиции), в которой располагается заданный символ или цепочка заданных символов;

— сжать текст, удалив из него один символ или цепочку символов;

— раздвинуть текст, вставив заданную последовательность символов между $i$-м и $(i+1)$-м символами исходного текста;

— разделить текст на строки, если в качестве ограничителя строки в тексте используется какой-либо специальный символ;

— разделить текст на строки, используя какое-либо специальное условие (например, ограничение на длину строки);

— сравнить символы или цепочки символов;

— выделить слово из текста;

— определить, является буква гласной или согласной;

— выделить из текста число и преобразовать его в числовую форму;

— преобразовать число в символьную форму;

— сформировать новый текст из частей исходного текста, удовлетворяющих заданному условию.

Рассмотрим реализацию этих действий на языке бейсик. Для выполнения некоторых из них можно использовать специальные функции. Выполнение других нужно программировать.

*1. Выделение i-го символа из текста* T□ осуществляется при помощи функции

SEG□(T□, I, I).

*2. Определение позиции, начиная с которой располагается в тексте* T□ *цепочка символов* D□, осуществляется при помощи функции

POS(T□, D□, K),

где K — позиция, с которой начинается поиск.

*3. Чтобы сжать исходный текст* T□, удалив из него цепочку символов, от $(i+1)$-го до $(i+n)$-го, нужно переслать по частям текст T□ в новую символьную переменную G□: от начала до $i$-го символа и от символа $(i+n+1)$-го до конца, используя *операцию сочленения*:

G□=SEG□(T□, 1, I)+SEG□(T□, I+N+1, N)

(N — длина исходного текста).

*4. Чтобы раздвинуть текст* T□, вставив последовательность символов V□ между $i$-м и $(i+1)$-м символами исходного текста, нужно текст T□ переслать по частям от начала до $i$-го символа и от $(i+1)$-го символа до конца в новую символьную переменную G□, вставив между ними требуемую цепочку символов V□, используя операцию сочленения:

G□=SEG□(T□, 1, I)+V□+SEG□(T□, I+1, N),

V□ должно быть определено в программе до выполнения приведенного оператора. V□ может быть задано явно в виде последовательности символов, заключенных в кавычки. Например, оператор

G□=SEG□(T□, 1, I)+"␣"+SEG□(T□, I+1, N)

формирует текст G□, отличающийся от исходного текста T□ наличием пробела между $i$-м и $(i+1)$-м символами исходного текста.

Раздвигая текст, нужно следить за тем, чтобы суммарная длина нового текста не превышала 255.

*5. Чтобы выделить и напечатать первую строку*, нужно определить позицию первого появления в тексте T□ символа, являющегося разделителем (пусть это — символ #):

P1=POS(T□, "#", 1).

Первая строка V□ — это часть текста от 1-го до (P1—1)-го символов:

V□=SEG□(T□, 1, P1—1).

Чтобы выделить вторую строку, нужно найти позицию разделителя в части текста, начиная с (P+1)-й позиции, где P=P1:

P1=POS(T□, "#", P+1).

Вторая строка — это часть текста от (P+1)-й до (P1—1)-й позиции

V$ = SEG$(T$, P+1, P1—1)

и т. д.

Далее приводится фрагмент программы (программа 4.71) для печати текста T$ по строкам (предполагается, что последняя строка кончается разделителем). N — длина текста.

**Программа 4.71 (фрагмент)**

```
1000 P=0
1010 P1=POS(T$,"╪",P+1)
1020 V$=SEG$(T$,P+1,P1-1)
1030 PRINT V$
1040 P=P1
1050 IF P<N GO TO 1010
```

*6. Чтобы разделить текст на строки*, содержащие не более 50 символов, делая перенос на другую строку на месте пробела, можно действовать двумя способами:

— определять позиции последовательных пробелов в тексте, запоминая позицию предыдущего пробела. Как только позиция очередного пробела станет > 50, считать первой строкой текста часть текста от начала до предыдущего пробела. Просмотр текста продолжать, начиная от новой строки, применяя описанный алгоритм;

— определить позицию первого пробела, начиная с 51-й позиции, а дальше, двигаясь по тексту назад, искать первый слева пробел. Считать его разделителем строк.

Составим программу (фрагмент) для второго варианта (см. программу 4.72).

*Список использованных переменных*. Исходные данные: T$ — исходный текст, N — длина текста.

Результат: V$ — очередная строка текста (⩽ 50 символов) для вывода на печать.

Вспомогательные переменные: P1 — позиция начала очередной строки в исходном тексте, P — позиция первого пробела после очередной строки в исходном тексте, I — текущий номер позиции. После окончания поиска предыдущего пробела I — позиция предыдущего пробела.

**Программа 4.72 (фрагмент)**

```
1010 P1=1
1020 IF N-P1<50 GO TO 1100
1030 P=POS(T$," ",P1+50)
1040 IF P<>P1+50 GO TO 1060
1050 I=P \ GO TO 1090
1060 I=50
1070 IF SEG$(T$,I,I)=" " GO TO 1090
1080 I=I-1 \ GO TO 1070
1090 V$=SEG$(T$,P1,I-1) \ P1=I+1 \ GO TO 1020
1100 V$=SEG$(T$,P1,N)
```

*7. Сравнение символов или цепочек символов* основано на сравнении числовых значений их кодов. Сравнение можно производить в операторе условного перехода. См., например, оператор 1070 в предыдущей программе.

*8. Чтобы выделить слово из текста,* воспользуемся следующими соображениями. *Слово* — это последовательность букв, заключенная между символами, не являющимися буквами. В некоторых задачах одно слово от другого отделяет заданный символ (пробел, запятая и т. п.). В этом случае нужно найти очередной символ — разделитель, и часть текста, заключенная между позициями двух соседних разделителей, будет словом. Например, чтобы выделить первое слово, нужно найти позицию K первого появления разделителя (например, «,») в тексте:

K=POS(T□, ", ", 1).

Тогда первым словом будет:

SEG□(T□, 1, K—1).

Позиция второго разделителя:

K1=POS(T□, ", ", K+1),

и второе слово — это

SEG□(T□, K+1, K1—1).

Таким образом, положив до входа в цикл K=0 и выполняя в цикле операторы:

K1=POS(T□, ", ", K+1)
V□=SEG□(T□, K+1, K1—1)
K=K1

в переменной V□ будем последовательно получать слова текста.

В качестве условия выхода из цикла можно использовать условие K1=0, которое нужно проверять после вычисления K1. (Если в тексте, начиная с (K+1)-й позиции, нет больше «,», то значение функции POS равно 0.)

Итак, программа 4.73 выделяет по очереди в переменную V□ и печатает слова, разделенные в исходном тексте запятыми.

**Программа 4.73 (фрагмент)**

```
1000 K=0
1010 K1=POS(T$,",",K+1)
1020 IF K1=0 GO TO 1040
1030 V$=SEG$(T$,K+1,K1-1) \ PRINT V$ \ K=K1 \ GO TO 1010
1040 V$=SEG$(T$,K+1,N) \ PRINT V$
```

Оператор 1040 выделяет последнее слово из текста, так как его выделение не подчиняется общим правилам, используемым для других слов.

В общем случае слова в тексте (например, в предложении русского языка) могут разделяться различными символами и последовательностями символов. В этом случае разделителем является любой символ (или последовательность символов), не являющийся буквой. Тогда для очередного символа нужно выяснить, является ли он буквой. Появление символа — не буквы является признаком конца очередного слова.

Чтобы проверить, является ли символ буквой, можно воспользоваться тем, что все буквы русского алфавита имеют коды, численные значения которых следуют друг за другом. Самый младший код имеет буква *Ю*, самый старший *Ч* (это зависит от кодировки букв). Тогда *i*-й символ в тексте не является русской буквой, если выполняется одно из двух условий:

SEG$(T$, I, I)<"Ю",
SEG$(T$, I, I)>"Ч".

Приводимая ниже программа 4.74 выделяет в переменную V$ слова из текста T$ длиной L, разделенные любыми символами, не являющимися буквами. В переменной P подсчитывается количество букв в очередном слове. После печати очередного слова полагается P=0.

Программа 4.74 (фрагмент)

```
1000 P=0
1010 FOR I=1 TO L
1020 Z$=SEG$(T$,I,I)
1030 IF Z$<"Ю" GO TO 1060
1040 IF Z$>"Ч" GO TO 1060
1050 P=P+1 \ GO TO 1070
1060 V$=SEG$(T$,I-P,I-1) \ PRINT V$ \ P=0
1070 NEXT I
```

*9. Чтобы выяснить, является буква гласной или согласной,* нужно сравнить ее с элементами подготовленного заранее массива гласных или согласных (массив гласных предпочтительнее — требует меньше памяти). Если буква не совпадает ни с одним элементом массива гласных, то эта буква — согласная.

*10.* Если позиции числа в тексте известны, то *выделение* этого *числа* осуществляется функцией SEG$, а преобразование в числовую форму функцией VAL.

Если позиции числа в тексте неизвестны, то *поиск числа* (чисел) представляет самостоятельную задачу.

Если числа — натуральные, т. е. являются последовательностями цифр, то выделение таких чисел можно осуществлять аналогично выделению слов, учитывая, что числовые значения кодов цифр следуют друг за другом. Наименьший код имеет цифра 0, наибольший — 9.

Выделение чисел, записанных в другой допускаемой в бейсике форме, представляет более сложную задачу и здесь не рассматривается.

*11. Преобразование числа из числовой формы в символьную* осуществляет функция STR□.

*12. Чтобы сформировать новый текст из исходного,* включая многократно (в цикле) в конец нового текста части, выделяемые из исходного, возможно, с добавлением дополнительных символов, приходится многократно выполнять операцию сочленения нового текста с новыми цепочками символов, т. е. выполнять оператор

G□=G□+SEG□(T□, I, J)+V□,

где SEG□(T□, I, J) — часть исходного текста T□, V□ — дополнительная последовательность символов.

Пример 1. В исходном тексте T□ одно слово от другого отделено одним пробелом. Сформировать текст G□, в котором одно слово от другого отделяется двумя пробелами (программа 4.75).

*Список используемых переменных.* Исходные данные: T□ — исходный текст размером ⩽255; L — длина исходного текста.

Результат: G□ — текст, в котором слова разделены двумя пробелами (длина G□ не должна превышать 255).

Вспомогательные переменные: K — позиция предыдущего пробела, K1 — позиция следующего пробела.

**Программа 4.75 (фрагмент)**

```
1000 K=0
1010 K1=POS(T$," ",K+1)
1020 IF K1=0 GO TO 1050
1030 G$=G$+SEG$(T$,K+1,K1)+" "
1040 K=K1 \ GO TO 1010
1050 G$=G$+SEG$(T$,K+1,L)
```

Оператор 1050 включает в G□ последнее слово из текста T□.

Чтобы проверить работу этого алгоритма, нужно дополнить программу операторами для ввода текста T□ и определения его длины, а также печати результата (текста G□).

Пример 2. Выписать из текста слова, начинающиеся и заканчивающиеся на одну и ту же букву (программа 4.76).

Для решения задачи будем выделять слова (см. п. 8, программа 4.74) и проверять на равенство первую и последнюю буквы слова (рис. 4.16).

*Список используемых переменных.* Исходные данные: T□ — текст размером ⩽255.

Результат: V□ — слово, первая и последняя буквы которого совпадают, выводится на печать.

Вспомогательные переменные: I — номер позиции символа в тексте — управляющая переменная цикла, D□ — символьная

переменная для хранения первой буквы слова, P — принимает значение 1, если просматривается последовательность букв, принимает значение 0, если просматривается последовательность «не

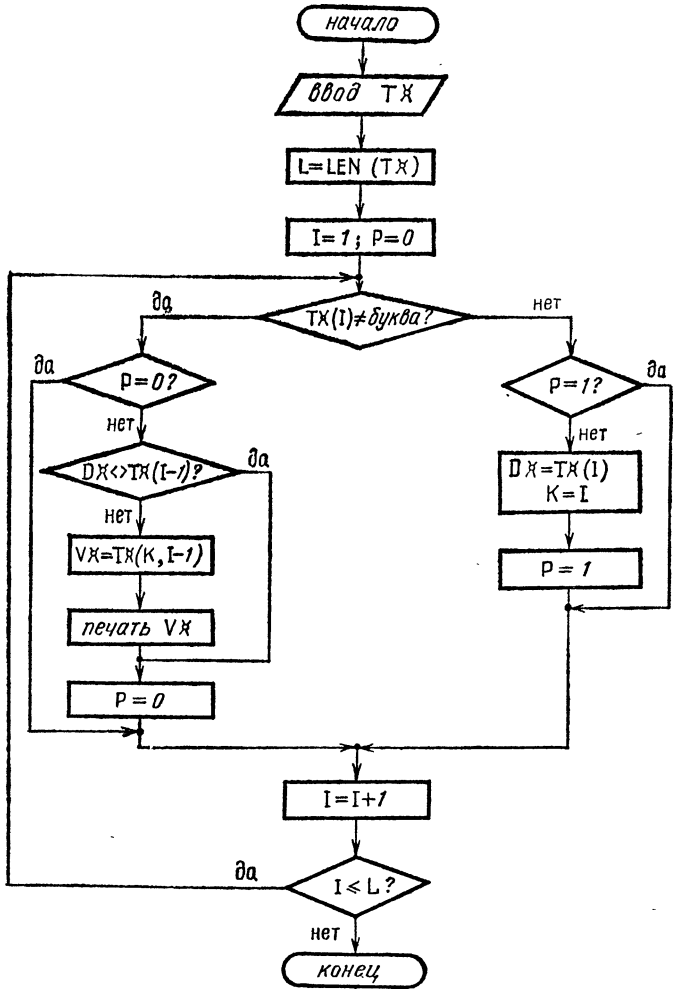



Рис. 4.16

букв». Если P изменяет значение на 1, то нужно запомнить в DO текущий символ текста (первую букву слова). Если изменяет значение на 0, нужно сравнить значение DO и предыдущий символ текста (последнюю букву слова), и в случае их равенства напеча-

тать слово; K — номер позиции первой буквы слова, L — длина текста.

На схеме T□(I) обозначает I-й символ текста, T□(K, I—1) — последовательность от K-го до (I—1)-го символов.

**Программа 4.76**

```
10 PRINT "ВВЕДИТЕ ТЕКСТ"
20 INPUT T$
30 L=LEN(T$) \ P=0
40 FOR I=1 TO L
50 Z$=SEG$(T$,I,I)
60 IF Z$<"Я" GO TO 100
70 IF Z$>"Ч" GO TO 100
80 IF P=1 GO TO 140
90 D$=Z$ \ K=I \ P=1 \ GO TO 140
100 IF P=0 GO TO 140
110 IF D$<>SEG$(T$,I-1,I-1) GO TO 130
120 V$=SEG$(T$,K,I-1) \ PRINT V$
130 P=0
140 NEXT I
150 STOP
```

Пример 3. Подсчитать, сколько слов в тексте начинается на букву K. Слова в тексте разделены пробелами (см. рис. 4.17 и программу 4.77).

*Список используемых переменных.* Исходные данные: T□ — текст размером ⩽255.

Результат: N — число слов, начинающихся на K.

Вспомогательные переменные: P — позиция разделителя (␣), I — текущая позиция текста.

**Программа 4.77**

```
10 PRINT "ВВЕДИТЕ ТЕКСТ"
20 INPUT T$
30 L=LEN(T$) \ N=0
40 IF (SEG$(T$,1,1))="К" THEN N=1
41 I=2
50 P=POS(T$," ",I)
60 IF P=0 GO TO 90
70 IF SEG$(T$,P+1,P+1)="К" THEN N=N+1
80 I=P+1 \ GO TO 50
90 PRINT "НА БУКВУ  К  НАЧИНАЕТСЯ";N;" СЛОВ"
100 STOP
```

Задание I уровня. Текст задать как символьную переменную. Определить, какой типовой алгоритм и функции, определенные для символьных переменных, можно использовать для решения задачи. Выявить циклы, продумать их организацию. Составить список используемых переменных. Составить схему и программу. Подготовить тесты. Выполнить программу вручную. Проверить работу программы на ЭВМ.

*Варианты задач I уровня.*

1. Определить, сколько раз в тексте встречается буква А.

2. В заданном тексте везде заменить слово A1□ на слово A2□ (длины слов не совпадают).

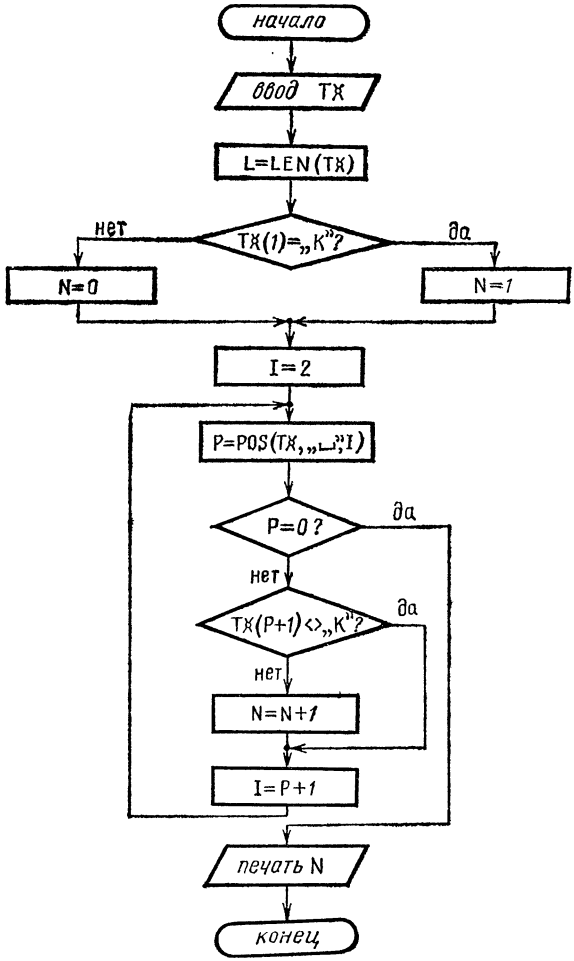



Рис. 4.17

3. В заданном тексте удалить часть текста, заключенную в скобки (вместе со скобками).

4. Определить количество слов в тексте.

5. Указать минимальное количество первых букв, по которым можно различить слова из заданного набора (слова разделены запятыми).

6. Текст задан следующим образом: первый символ — цифра, указывающая длину первого слова, за первым словом — вновь цифра, указывающая длину второго слова (длина каждого слова ⩽9) и т. д. Выписать K-е слово из текста.

7. Текст задан способом, указанным в задаче 6. Вместо цифр, указывающих длину слова, вставить числа, указывающие координату начала следующего слова.

8. Напечатать самое длинное слово из заданного текста.

9. Определить, какой процент слов в тексте содержит удвоенную согласную (слова разделены пробелами).

10. Сколько раз в тексте встречается заданное слово (слова разделены пробелами).

11. В тексте убрать лишние пробелы между словами, оставив по одному.

12. В тексте вставить между словами вместо одного пробела запятую и пробел.

13. Разделить заданный текст на строки. В качестве разделителя в тексте используется символ %.

14. Определить, какой процент слов в тексте начинается на букву К. Слова разделены пробелами.

15. Сколько раз в тексте встречается заданное слово (в качестве разделителей могут употребляться пробелы, знаки препинания и т. п.).

Указания к решению задач I уровня.

1. Определить длину текста. Выделять по одному символу из текста и сравнивать его с символом A. При совпадении к сумме числа символов A прибавлять 1.

2. Сформировать новый текст G○, в котором вместо A1○ будет A2○. Для этого выделять из исходного текста последовательно цепочки символов длиной, равной длине A1○, и сравнивать их с A1○. При совпадении переслать в G○ часть исходного текста до цепочки A1○, затем A2○, используя операцию сочленения. Продолжить просмотр исходного текста. При появлении цепочки A1○ присоединить к G○ часть текста между двумя цепочками A1○, затем A2○ и т. д. Когда исходный текст будет просмотрен до конца, присоединить к G○ «хвост» исходного текста после последней цепочки A1○ (если A1○ — не последняя цепочка символов в тексте).

3. Найти позицию символа «(». Часть исходного текста до «(» переслать в новый текст G○. Дальнейший просмотр исходного текста осуществлять с позиции, следующей за позицией первого символа «)». Если в оставшейся части будет обнаружен символ «(», переслать в G○ часть исходного текста между символами «)» и «(» и т. д.

4. См. п. 8 теоретического введения к работе.

5. Выделять первую букву каждого слова и запоминать в сим-

вольном массиве, сравнивая каждую новую букву со всеми предыдущими элементами массива. Если обнаружатся две совпадающие буквы, то, просматривая слова сначала, выбирать по две первые буквы каждого слова, запоминать в том же символьном массиве, сравнивая каждую новую пару со всеми предыдущими и т. д. Если для какой-то длины последовательности первых букв удается просмотреть все слова и не обнаружить двух совпадающих последовательностей, то ее длина и будет искомым числом.

6. Извлечь первый символ из текста, преобразовать в числовую форму, определить позицию следующей цифры, преобразовать ее в числовую форму. Повторив К раз эту операцию, получим позицию, с которой начинается К-е слово (позиция К-й цифры +1), и ее длину (это сама К-я цифра).

7. Позиция начала следующего слова может быть двух- или трехзначным числом, что делает необходимым раздвинуть текст, чтобы поместить эти цифры вместо одной, определяющей по условию задачи длину слова. Но это обстоятельство, в свою очередь, изменяет и само число (номер позиции начала следующего слова). Поэтому, чтобы найти число, которое нужно поставить перед очередным словом, нужна информация как о длине этого слова, так и о длине следующего, чтобы определить, сколько цифр будет содержать номер позиции, который нужно расположить перед этим следующим словом в новом тексте. Новый текст нужно формировать в символьной переменной, пересылая в нее (используя операцию сочленения) определяемый номер позиции начала следующего слова, преобразовав его в символьную форму при помощи функции STR$, и сами слова.

8. Определяя номера позиций, в которых расположены соседние пробелы, находить длину каждого слова. В символьную переменную пересылать слово, длина которого больше длины слов просмотренных ранее, а в числовую — длину слова, которое оказалось длиннее всех предыдущих. После просмотра всех слов напечатать содержимое символьной переменной (алгоритм похож на поиск максимального из чисел, вводимых по очереди в одну и ту же переменную).

9. Сформировать массив согласных (или лучше гласных). Находить позицию очередного пробела и, извлекая по одной букве из части текста между соседними пробелами (слова), сравнивать каждую его букву со следующей Если буквы совпадают, то сравнить эту букву с каждой буквой из массива гласных. Если она не совпадает ни с одной из гласных, то, значит, в слове была удвоенная согласная, и к сумме слов, содержащих удвоенную согласную, нужно прибавить 1.

10. Определить длину заданного слова. Определяя позиции соседних пробелов, сравнивать длину каждого слова с длиной за-

данного слова. Если длины совпадают, то сравнить слова, и если слова совпадут, то к сумме количества слов прибавить единицу. Продолжать просмотр текста до конца.

11. Определить позицию первого пробела. Проверить, не является ли следующий символ пробелом и т. д., т. е. определить позиции следующих подряд пробелов. Часть текста, включая первый найденный пробел, переслать в новый текст. Продолжить просмотр, начиная с текущей позиции. Найдя повторяющиеся пробелы, повторить процедуру.

12. Найдя пробел, переслать в новый текст, используя сочленение, часть текста до пробела, запятую и пробел. Просмотр продолжить до конца текста, и каждый раз, встречая пробел, выполнять описанную выше процедуру.

13. Текст просматривать до первого появления символа-разделителя. Во вспомогательную символьную переменную переслать начало текста — до разделителя и напечатать. Просмотр продолжить до появления следующего разделителя и повторить процедуру.

14. Первый символ текста сравнить с К. Далее находить очередной пробел, и первый символ после пробела (первую букву слова) сравнивать с К. Вести подсчет числа совпадений и общего числа слов в тексте.

15. Выделять слова (см. программу 4.74) и сравнивать с заданным словом.

Другой способ: определить длину заданного слова, выделить из текста последовательности той же длины и сравнить с заданным словом. Первая выделенная последовательность включает символы с 1-го до L-го, вторая — со 2-го до (L+1)-го и т. д.

Если в тексте найдено заданное слово, то при выделении новой последовательности найденное слово не просматривается (делается шаг по тексту на L позиций (а не на одну) вправо).

Задание II уровня. Выполнение задания II уровня требует использования методов и средств работы с текстами, изложенными в теоретическом введении к работе. В некоторых задачах необходимо использовать числовые и символьные массивы. Требуется также выполнить все пункты задания I уровня.

*Варианты задач II уровня.*

1. Задан текст, содержащий не более 255 символов. Определить частоту, с которой встречаются в тексте различные буквы русского алфавита (в долях от общего количества букв).

2. Зашифровать заданный текст (не более 255 символов), используя один перемешанный алфавит (полученный случайной перестановкой всех букв исходного алфавита).

АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
ЕЖЧБПКЛАОТДУГЦЯЙХЫВЪЮИФСЩНЬЭЗШРМ

3. Используя один перемешанный алфавит, получить четыре других, циклически сдвинутых относительно исходного на $n_1$, $n_2$, $n_3$, $n_4$ букв (массивы не использовать).

4. Ученики зашифровывают свои записки, записывая все слова наоборот. Составить программу, зашифровывающую и расшифровывающую сообщение.

5. Разбить исходный текст на строки длиной не более 50 символов. Перенос на новую строку осуществлять на месте пробела (слова не переносить).

6. В текст, содержащий меньше 50 символов, равномерно вставить пробелы между словами, чтобы его длина составляла ровно 50 символов.

7. Назовем *сложностью предложения* сумму количества слов и знаков препинания. Определить сложность заданного предложения.

8. Задан текст длиной $\leqslant$1000 символов. Напечатать буквы, на которые начинаются слова в тексте, в порядке убывания частоты их употребления.

9. Проверить сбалансированность скобок в тексте (скобки сбалансированы, если закрывающая скобка расположена после соответствующей открывающей и их количества совпадают).

10. Определить, сколько слов в тексте содержит 1 слог, 2 слога, 3 слога и т. д.

11. На какую букву начинается больше слов в тексте.

12. Определить, какие символы и сколько раз встречаются в тексте.

13. Задан список группы и пять оценок каждого студента. Фамилии от оценок и оценки друг от друга отделены символом *. Напечатать список группы и средний балл каждого студента.

14. Задан текст длиной $\leqslant$1000 символов. Выписать все слова, включающие заданную последовательность букв (например, выписать однокоренные слова).

15. Задан список группы студентов (фамилия, пол, год рождения). Данные для различных студентов отделяются запятыми, различные сведения об одном студенте — пробелами. Составить список студентов (мужского пола) заданного года рождения.

Указания к решению задач II уровня.

1. Ввести русские буквы (в алфавитном порядке) в массив размером 32. Предусмотреть числовой массив того же размера для частот повторения каждой буквы. Просматривать текст и для каждой буквы прибавлять 1 к соответствующему элементу массива частот. Подсчитать также общее количество букв. После окончания просмотра текста каждый элемент массива частот заменить результатом деления этого элемента на общее число букв. Напечатать массив букв и массив частот их появления.

2. Обычный и перемешанный алфавит хранить в двух символь-

ных переменных. Каждую букву текста найти в обычном алфавите и заменить на букву, стоящую в той же позиции в перемешанном.

3. Результат получить в виде символьного массива размером 4, каждый элемент которого — перемешанный алфавит, сдвинутый относительно исходного на заданное число позиций. Сдвиг достигается сочленением в нужном порядке частей исходного текста.

4. Выделять каждое слово из текста (см. п. 8 теоретического введения) и пересылать его в новый (закодированный) текст в обратном порядке. Символы между словами пересылать в прямом порядке.

5. Исходный текст задать в виде символьного массива. Результат получить в виде символьного массива, каждый элемент которого содержит не более 50 символов.

6. Определив длину исходного текста, найти, сколько пробелов нужно вставить. Определить число пробелов (N) и их позиции в исходном тексте. Позиции пробелов поместить в массив P. Удвоить все пробелы, если нужно добавить больше чем N пробелов. Если число добавляемых пробелов меньше, можно удваивать пробелы в исходном тексте в следующем порядке:

— удвоить средний пробел (расположенный в позиции $K=P\left(\left[\frac{N+1}{2}\right]\right)$);

— удвоить пробелы в позициях $K1=P\left(\left[\frac{K+1}{2}\right]\right)$ и $K2=P\left(\left[\frac{K+N}{2}\right]\right)$ и т. д.

7. Знаками препинания будем считать символы, не являющиеся буквами и пробелами.

8. Текст задать в одномерном символьном массиве размером не менее 4 (длина одного элемента $\leqslant 255$ символов). Если слово целиком не помещается в элемент массива, то помещать его в следующий элемент. Выделять слова (см. п. 8 теоретического введения), и первую букву каждого слова (если такая буква еще не встречалась) помещать в массив размером 30 (по числу букв, которыми может начинаться слово). В числовом массиве размером 30 осуществлять подсчет числа появления различных букв в начале слов.

После окончания просмотра текста оба массива упорядочить по убыванию элементов числового массива. После упорядочения символьный массив напечатать.

9. Сравнивать по очереди каждый символ с «(» и с «)». Если очередной символ «(», то к числовой переменной (до начала просмотра ее нужно положить равной 0) прибавить 1, если символ «)», то прибавить —1. Если закрывающая скобка появилась раньше, чем открывающая, то значение числовой переменной станет отрицательным — скобки не сбалансированы, просмотр можно прекратить.

Если текст просмотрен до конца и значение переменной равно 0, скобки сбалансированы.

10. Задача сводится к определению количества гласных в одном слове (см. п. 9 теоретического введения). В числовом массиве будем накапливать сумму слов, содержащих 1 слог (первый элемент массива), 2 слога (второй элемент), 3 слога (третий элемент) и т. д.

11. Каждую новую букву, на которую начинается слово, будем заносить в символьный массив. Количество повторений букв суммируем в числовом массиве. После окончания просмотра текста нужно определить максимальный элемент числового массива и напечатать соответствующий элемент символьного (хранящего буквы).

12. См. указание к задаче 1.

13. Исходные данные задать в виде символьного массива, один элемент которого содержит сведения об одном студенте. Далее для каждой строки массива нужно извлекать символ, расположенный после каждой звездочки, преобразовывать в числовую форму и прибавлять к сумме оценок одного студента. Разделить полученное число на 5 (количество оценок одного студента), и полученный результат, преобразовав в символьную форму, поместить в исходный массив (после последней оценки). Для удобства вывода на печать можно средний балл располагать, начиная с одной и той же позиции элемента символьного массива.

14. Задать текст, как указано в задаче 8. Определить длину L заданной последовательности букв. Выделять слова и сравнивать цепочки букв длиной L с заданной последовательностью. Если длина выделенного слова $<L$, сравнение не производить и переходить к следующему слову. См. также указание к задаче 15 уровня I (второй способ).

15. Задать текст, как указано в задаче 13. Пол задать буквой М. Символ после первого пробела для каждого студента сравнивать с М. В случае совпадения четыре символа после второго пробела (год рождения) сравнивать с заданным. В случае совпадения вывести на печать фамилию — цепочку символов, расположенную после запятой до первого пробела. Для этого позицию предыдущей запятой нужно запоминать. Чтобы фамилию первого студента в случае необходимости выделять по общему правилу, можно добавить в текст фиктивную запятую (перед первой фамилией) и до начала просмотра текста считать ее позицию равной 0.

Задание III уровня. Это реальные задачи, возникающие в различных областях практической деятельности человека. Их решение требует творческого подхода, а также применения приемов и методов, используемых при выполнении заданий I и II уровней. При выполнении задания III уровня необходимо также выполнить все пункты заданий I и II уровней.

Размер исходного текста, как правило, не ограничен 255 символами. Его нужно размещать в символьном массиве, элементы которого — части исходного текста по 255 символов (кроме, возможно, последнего элемента).

*Варианты задач III уровня.*

1. Проанализировать зашифрованный текст, чтобы выяснить, с помощью одного или нескольких различных алфавитов произведено шифрование. Для этого вычислить «индекс совпадения» (ИС) для всех букв по формуле:

$$\text{ИС} = \sum_{i=1}^{32} \frac{f_i (f_i - 1)}{N(N-1)},$$

где $f_i$ — количество появлений $i$-й буквы, N — общее количество букв. Если все буквы зашифрованы при помощи одного алфавита, то ИС для всех букв лежит в пределах от 0,045 до 0,065.

2. Разделить заданный текст (не более 1000 символов) на строки, содержащие не более 50 символов. (Перенос осуществлять на месте пробела.) Добавить равномерные пробелы, чтобы каждая строка содержала ровно 50 символов.

3. Пара скобок (открывающая и закрывающая) задана последовательностями символов A1◯ и A2◯ (их длины в общем случае не совпадают). Проанализировать заданный текст с целью обнаружения и локализации ошибок в использовании скобок. Возможно три типа ошибок:

1) несоответствие скобок A1◯ и A2◯ по количеству;

2) закрывающая скобка расположена до открывающей;

3) отсутствует содержание между скобками.

Результатом работы программы должно быть сообщение о типах допущенных ошибок и их месте в тексте (если возможно).

4. Составить программу, зашифровывающую заданный текст с использованием $n$ перемешанных алфавитов. В качестве исходных данных заданы: исходный алфавит, $n$ — число алфавитов, ($m_1$, ..., $m_n$) — количество букв, на которые сдвигается каждый алфавит относительно исходного.

5. Для хранения текста в сжатом виде найти часто повторяющиеся последовательности из двух букв и заменить их кодом. В качестве кода использовать символы, не встречающиеся в тексте. Составить также таблицу кодов.

6. В задаче 5 написать программу, декодирующую текст, хранящийся в сжатом виде, используя заданную таблицу кодов.

7. Составить программу «Меню». Машина задает вопрос о наличии основных продуктов, их количестве и требуемом числе порций, сообщает те блюда, которые можно приготовить, предлагает выбрать из них желаемое и сообщает его рецепт.

8. Составить программу назначения студентов на стипендию по результатам сессии, используя следующие правила:

1) если все оценки 5, назначается повышенная стипендия;

2) если оценки 4 и 5, назначается обычная стипендия;

3) если есть оценки 3, стипендия не назначается.

В результате работы программы должен быть напечатан список группы с оценками и средним баллом каждого студента и два списка фамилий (назначенных на повышенную и обычную стипендии).

9. В задаче 8 назначается повышенная стипендия с надбавкой 50%, если все оценки 5, с надбавкой 25%, если оценки 4 и 5, и обычная стипендия, если есть оценки 3. В результате выполнения программы должны быть напечатаны три списка фамилий студентов, назначенных на три различные стипендии.

10. Расшифровать текст, если известно, что он зашифрован при помощи одного перемешанного алфавита.

11. Список фамилий, разделенных запятыми, задан в произвольном порядке. Упорядочить его по алфавиту.

12. Список студентов (фамилия, имя, отчество) ввести в одномерный массив, отделяя запятыми сведения об отдельных студентах. Напечатать полученный массив. В отдельном массиве сформировать список фамилий по алфавиту.

13. В заданном тексте длиной $\leqslant 1000$ найти слова, которые встречаются более трех раз, закодировать их и сжать текст, заменив слова кодами. Составить таблицу кодов.

14. Считая, что в памяти ЭВМ хранится таблица кодов часто встречающихся слов, ввести текст в массив, заменяя слова кодами после ввода. Распечатать текст в исходном виде, т. е. заменяя коды словами.

15. Текст, передаваемый по телеграфу, содержит слова, разделенные пробелами (знаки препинания передаются сочетаниями букв). Считая, что таблица соответствия сочетаний букв и знаков препинания хранится в памяти ЭВМ, распечатать текст в привычном виде.

Указания к решению задач III уровня.

1. Исходный текст задать в виде символьного массива (один его элемент содержит не более 255 символов). См. также указание к решению задачи 1 уровня II.

2. См. указания к решению задач 5, 6 уровня II.

3. Для результата предусмотреть три переменные, в которых будет суммироваться количество ошибок каждого типа, и два массива размером 5 (максимальное предполагаемое число ошибок каждого типа) для хранения позиций ошибок 2-го и 3-го типа. Позиции ошибок 1-го типа указать нельзя (если только это не следствие ошибок 2-го типа).

4. Решение задачи состоит из следующих этапов:

— ввод данных;

— подготовка $n$ перемешанных алфавитов (см. указание к решению задачи 3 уровня II);

— просмотр исходного текста и получение зашифрованного текста (см. указание к решению задачи 2 уровня II), используя по очереди $n$ подготовленных алфавитов.

5. Выделять слова. В каждом слове выделять сочетания по 2 символа и не встречавшиеся ранее сочетания помещать в символьный массив. Если найдено повторяющееся сочетание, то в числовом массиве соответствующее количество увеличить на 1 (см. указание к решению задачи 11 уровня II). При этом вторую букву этого сочетания нужно пропустить при выборе следующей пары букв. Для подготовки кодов использовать исходный массив всех символов и массив символов, из которых состоит текст (см. задачу 12 уровня II). Оба массива перед формированием массива кодов упорядочить. Если число повторяющихся сочетаний больше, чем число кодов, то кодировать нужно наиболее часто повторяющиеся сочетания. Для их выявления нужно упорядочить массив частот повторений пар букв с одновременной перестановкой пар букв. Для облегчения получения нового закодированного текста в массиве сочетаний букв нужно запоминать в соответствующем элементе (например, через *) номера позиций этих сочетаний в исходном тексте. Получить новый закодированный текст, используя операцию сочленения. Нужно учесть, что формировать новый текст можно только добавляя новые части в порядке возрастания позиций.

6. Подход к решению этой обратной задачи следует из прямой задачи (см. задачу 5).

7. Данные можно разместить следующим образом:

— название одного блюда, необходимые основные продукты, требуемое их количество на одну порцию — одна строка символьного массива;

— рецепт приготовления одного блюда — две строки (510 символов) другого символьного массива.

Программа должна работать в диалоговом режиме.

8. Исходную информацию задать в виде одномерного символьного массива. Один элемент массива содержит информацию об одном студенте (фамилия и 5 экзаменационных оценок, разделенных *). Предусмотреть вычисление среднего балла каждого студента (разместить после последней оценки через *). Использовать функции VAL и STR$\bigcirc$ для преобразования оценок из символьной формы в числовую и обратно. В результате работы программы должны быть сформированы и напечатаны два массива — в одном фамилии студентов, назначенных на повышенную стипендию, в другом — на обычную.

9. См. указание к задаче 8.

10. Задача состоит из двух этапов:

— выявление перемешанного алфавита,

— расшифровка текста с использованием этого алфавита.

Для выполнения первого этапа нужно определить частоту появления различных букв в зашифрованном тексте, а затем, используя таблицу известных частот появления букв в тексте, выявить соответствие между буквами обычного алфавита и буквами перемешанного, использованного при шифровании.

Таблица частот букв русского алфавита

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| О | .0940 | П | .0421 | Б | .0197 | Ц | .0087 |
| А | .0896 | М | .0417 | З | .0193 | Ж | .0064 |
| Е | .0856 | В | .0400 | У | .0179 | Ю | .0063 |
| И | .0739 | Л | .0358 | Г | .0153 | Щ | .0048 |
| Н | .0662 | К | .0322 | Ь | .0125 | Ф | .0034 |
| Т | .0611 | Л | .0280 | Ч | .0118 | Э | .0033 |
| Р | .0561 | Я | .0243 | Й | .0094 | Ш | .0032 |
| С | .0554 | Ы | .0225 | Х | .0093 | Ъ | .0002 |

После получения перемешанного алфавита выполнение второго этапа трудностей не представляет.

11. Возможность решения задачи упорядочением числовых значений кодов символов исключается, так как для русских букв порядок возрастания кодов не совпадает с алфавитным порядком.

Для решения задачи можно русским буквам в алфавитном порядке поставить в соответствие последовательность следующих подряд символов в порядке возрастания их кодов. Далее закодировать русский текст с использованием выбранных следующих подряд символов, выделить фамилии в одномерный массив, упорядочить этот массив по возрастанию и перекодировать его с использованием составленной ранее таблицы кодов.

12. См. указание к задаче 11.

13. Выделить первое слово из текста. Выделяя последующие слова из текста, сравнивать с первым, запоминая в числовом массиве М позиции, в которых находятся совпадающие слова. Если после окончания просмотра количество совпадений, равное текущему значению индекса массива М, больше трех, то занести слово в таблицу кодов и заменить слово кодом, начиная с конца. Повторить процедуру, описанную выше. Символы, которые могут быть использованы для кодирования (не буквы), целесообразно предварительно поместить в массив и использовать их по мере необходимости. Если символ используется для кодирования, то в соответствующий элемент второго массива нужно поместить слово, которое он заменяет.

14. Таблицу кодов ввести как два массива (слов и их кодов). Текст вводить отдельными словами, используя вспомогательную переменную V$. Каждое введенное слово (значение V$) сравнивать со словами, которые подлежат кодированию (элементами массива слов). Если введенное слово найдено в таблице кодов, то вместо слова в текст внести код. При формировании текста использовать операции сочленения.

15. См. указание к задаче 14.

Вопросы для самопроверки.

1. Способы задания текстов. Использование символьных переменных и символьных массивов. При какой длине текста его можно задать как символьную переменную? В каком случае при вводе текста оператором INPUT необходимо использовать кавычки (или апострофы)?

2. В чем сходство способов обработки текстов и обработки массивов?

3. Какие стандартные функции определены для символьных величин? Как выполняется операция сочленения?

4. Как определить позицию заданного символа (цепочки символов) в тексте, как выделить символы, расположенные, начиная с заданной позиции?

5. Как удалить из текста последовательность символов, вставить в текст последовательность символов?

6. Как разделить текст на строки, когда а) конец строки отмечен ограничителем, б) задана длина строки?

7. Как выделить слова из текста? Почему последнее слово нельзя выделить по общему правилу?

8. Как выделить число из текста? Как выполнить арифметическую операцию с числом, представленным в символьном виде?

9. Как определить, является буква гласной или согласной?

10. Какие типовые алгоритмы обработки массивов могут быть использованы при обработке текстов?

11. Как определить, какие символы и сколько раз встречаются в тексте?

## *Работа 10.* ПРОСТЕЙШИЕ ПРИЕМЫ РАБОТЫ С ВНЕШНИМИ УСТРОЙСТВАМИ

Теоретическое введение. Перед выполнением работы следует изучить средства бейсика для работы с внешними устройствами, изложенные в п. 13 главы 3.

Можно выделить две основные формы использования внешних устройств при работе на мини- и микро-ЭВМ.

1. Использование внешних устройств при работе пользователя со своей программой, а именно: запоминание и сохранение текста программы на ВЗУ с целью ее многократного использования и сохранения для последующего сеанса, т. е. создание так называемого программного файла, использование ПУ для распечатки программы.

Для этого применяются специальные команды: OLD, RUN, SAVE, REPLACE, UNSAVE (см. п. 13.1, 13.2 главы 3).

2. Использование внешних устройств при работе (выполнении) самой программы пользователя *). Например, считывание с ВЗУ исходных данных для выполнения программы, запись результатов работы программы на ВЗУ.

На ВЗУ данные хранятся в виде файлов.

В бейсике можно организовывать файлы последовательного доступа и файлы прямого доступа (см. п. 13.2 главы 3). Как правило, файлы последовательного доступа используют для хранения данных, которые предполагается обрабатывать поочередно (всегда в одном и том же порядке!) одинаковым образом, а также когда заранее неизвестно общее количество данных, которые предполагается разместить в файле (при формировании файла) или которые содержатся в файле (при считывании из файла). Файлы прямого доступа применяются для хранения заранее известного количества данных, причем в процессе их обработки можно обращаться непосредственно сразу к нужному данному, зная его местоположение в файле, без предварительного просмотра предыдущих данных.

В работе рассматривается организация файлов на диске (дискете) (см. п. 13.2 главы 3). На устройствах такого типа возможна организация файлов как последовательного, так и прямого доступа.

Пример 1. Сформировать на ВЗУ файл последовательного доступа с именем "STUD1", содержащий фамилии и года рождения всех студентов одного курса. Используя сформированный файл, вывести на экран дисплея фамилии студентов, которым исполнится 21 год в 1989 году.

Решение.

1. **Первый этап** состоит в формировании на ВЗУ файла последовательного доступа с именем "STUD1". Файл должен иметь следующую структуру: $F_1G_1F_2G_2\ldots K$, где $F_i$, $G_i$ ($i=1, 2, \ldots$) — фамилия и год рождения очередного студента, $K$ — признак конца файла.

Фамилию и год рождения каждого студента будем вводить по очереди в оперативную память в символьную переменную F$ и числовую G и пересылать их в файл "STUD1", используя оператор PRINT #$N$, $N$ — номер канала (программа 4.78).

Пояснения к программе. В строке 10 открывается файл с именем "STUD1" для записи информации.

В программе можно было бы предусмотреть ввод имени файла в символьную переменную и использовать далее эту переменную в операторе OPEN. Для этого нужно было бы использовать дополни-

*) Применение ПУ для вывода результатов работы программы рассмотрено в работе 7.

тельные строки, например:

```
1 PRINT "ВВЕДИТЕ ИМЯ ФАЙЛА-НЕ БОЛЕЕ 6 СИМВОЛОВ"
2 INPUT N$
3 IF LEN (N$)>6 GO TO 1
```

и заменить строку 10 на

```
10 OPEN N$ FOR OUTPUT AS FILE #1
```

При выполнении операторов INPUT (строки 30 и 60) фамилия и год рождения очередного студента вводятся в оперативную память (в переменные F$ и G). Операторы в строках 70, 80 заносят эту информацию в файл "STUD1".

При окончании списка студентов (ввод пробела в строке 30) выполняется оператор 100, и файл "STUD1" закрывается. При этом автоматически формируется признак конца файла.

**Программа 4.78**

```
10 OPEN "STUD1" FOR OUTPUT AS FILE #1
20 PRINT "ВВЕДИТЕ ФАМИЛИЮ СТУДЕНТА , ДЛЯ ОКОНЧАНИЯ ВВОДА - ПРОБЕЛ"
30 INPUT F$
40 IF F$=" " GO TO 100
50 PRINT "ВВЕДИТЕ ГОД РОЖДЕНИЯ - 4 ЦИФРЫ"
60 INPUT G
70 PRINT #1,F$
80 PRINT #1,G
90 GO TO 20
100 CLOSE #1
110 STOP
```

2. Второй этап — вывод на экран дисплея списка студентов, которым исполнится 21 год в 1989 году.

Решим эту задачу, используя данные о студентах (фамилия и год рождения), хранящиеся на ВЗУ в файле с именем "STUD1". Для этого откроем файл "STUD1" для считывания и будем считывать по очереди сведения о каждом студенте в оперативную память (в переменные F$ — фамилия и G — год рождения), сравнивать G с 1968 и при их равенстве выводить на экран фамилию, содержащуюся в переменной F$ (программа 4.79).

**Программа 4.79**

```
10 PRINT "СПИСОК СТУДЕНТОВ" \ PRINT
20 OPEN "STUD1" FOR INPUT AS FILE #5
30 IF END #5 GO TO 70
40 INPUT #5,F$,G
50 IF G()1968 GO TO 30
60 PRINT F$ \ GO TO 30
70 PRINT "КОНЕЦ ДАННЫХ"
80 CLOSE #5
90 STOP
```

Пояснения к программе. В строке 20 файл с именем "STUD1" открывается для считывания.

Оператор в строке 40 считывает из файла в оперативную память фамилию студента (переменная F$) и год его рождения (переменная G). При появлении признака конца файла (строка 30) управление будет передано на конец программы (к строке 70). В строке 80 файл данных закрывается.

Пример 2. Соревнования по тяжелой атлетике проводятся по двум упражнениям (рывок и толчок). В соревнованиях принимают участие 20 спортсменов. Протокол соревнований в виде фамилий участников и их результатов в двух упражнениях в порядке стартовых номеров необходимо сохранить на ВЗУ для последующей обработки.

Далее обработать результаты соревнований, хранящиеся на ВЗУ, для определения победителя по сумме результатов в двух упражнениях.

Решение.

1. При проведении соревнований на ВЗУ сформируем два файла прямого доступа: с именем "FAM" для хранения символьной информации (фамилий) и "RES" для хранения числовой информации (результатов соревнований). Результаты сформируем в числовой матрице размером 20×2 (первый столбец — результаты в упражнении «рывок», второй столбец — в упражнении «толчок») (программа 4.80).

**Программа 4.80**

```
10 DIM #1,F$(20)*25
20 OPEN "FAM" AS FILE #1
30 DIM #2,R(20,2)
40 OPEN "RES" AS FILE #2
50 FOR I=1 TO 20
60 PRINT "ВВЕДИТЕ ФАМИЛИЮ";I;"-ГО УЧАСТНИКА (НЕ БОЛЕЕ 25 СИМВОЛОВ)"
70 INPUT A$
80 F$(I)=A$
90 PRINT "ВВЕДИТЕ РЕЗУЛЬТАТ В РЫВКЕ И ТОЛЧКЕ"
100 INPUT B,C
110 R(I,1)=B \ R(I,2)=C
120 NEXT I
130 CLOSE #1,#2
140 STOP
```

Пояснения к программе. В строках 10—40 описываются и открываются файлы прямого доступа "FAM" и "RES". Фамилии участников, которые будут заноситься в файл "FAM" в соответствии с описанием (строка 10), не должны содержать больше 25 символов.

В цикле по I (стартовый номер участника) вводится фамилия очередного спортсмена в переменную A$ (строка 70) и его результаты в двух упражнениях — в переменные B и C (строка 100). Фамилия заносится в I-й элемент файла "FAM" (строка 80), результаты — в I-ю строку файла "RES" (строка 110). После выхода из

цикла (окончания формирования файлов) файлы закрываются (строка 130).

2. Второй этап состоит в определении победителя соревнований по сумме двух упражнений, результаты которых хранятся на ВЗУ в файле прямого доступа "RES".

После описания и открытия файлов нужно организовать цикл по номеру участника и, считывая по очереди результаты из файла "RES" в оперативную память в переменные B и C, запоминать номер K участника с лучшей суммой (B+C). После просмотра результатов всех участников в переменной K будет номер (индекс) участника, имеющего наилучший результат.

Из файла "FAM" далее считывается в оперативную память K-й элемент (фамилия победителя), из файла "RES" — его результаты и выводятся на печать (программа 4.81).

**Программа 4.81**

```
10 DIM #1,F$(20)=25
20 OPEN "FAM" AS FILE #1
30 DIM #2,R(20,2)
40 OPEN "RES" AS FILE #2
50 L=-1
60 FOR I=1 TO 20
70 B=R(I,1) \ C=R(I,2)
80 L1=B+C
90 IF L1)L THEN L=L1 \ K=I
100 NEXT I
110 A$=F$(K)
120 PRINT "ПОБЕДИТЕЛЬ СОРЕВНОВАНИЙ-";A$
130 PRINT "ЕГО РЕЗУЛЬТАТ В АВОЕБОРЬЕ-";L;"КГ"
140 L1=R(K,1) \ L2=R(K,2)
150 PRINT "В РЫВКЕ-";L1;"КГ ,В ТОЛЧКЕ-";L2;"КГ"
160 CLOSE #1,#2
170 STOP
```

Пояснения к программе. В строках 10—40 описываются и открываются на ВЗУ файлы "FAM" и "RES".

В строках 60—100 организован цикл для определения участника, имеющего лучший результат в сумме двух упражнений.

Для определения лучшего результата используется алгоритм, приведенный в теоретическом введении к работе 8 (рис. 4.13 и программа 4.64), с некоторой модификацией.

Для каждого участника с номером I результаты пересылаются из файла "RES" в оперативную память в переменные B и C (строка 70). Сумма по двум упражнениям вычисляется в переменной L1 (строка 80). Лучшая из вычисленных сумм пересылается в переменную L. В этом случае изменяется и номер лучшего спортсмена K (строка 90). Если условие в строке 90 не выполняется (L1≤L), то сразу осуществляется переход к следующей строке (к оператору NEXT I).

До входа в цикл L полагается равным отрицательному числу, так что первый спортсмен заведомо превзойдет этот результат и после первого прохождения цикла будет L=B+C (для первого участника) и K=1.

В строках 110—150 организован вывод на экран дисплея фамилии победителя (в строке 110 она считывается в оперативную память в переменную A$ из K-го элемента файла "FAM") и его результатов.

В строке 160 закрываются файлы "FAM" и "RES", обмен с которыми осуществлялся по каналам 1 и 2.

З а д а н и е I у р о в н я. Предназначено для выработки навыков создания, корректировки, сохранения на ВЗУ программных файлов.

В задании I уровня следует выполнить следующую последовательность действий:

1) командой NEW (NEW *имя*) задать имя программы;

2) набрать на экране дисплея предлагаемый текст программы;

3) записать программу на ВЗУ под выбранным именем командой SAVE (SAVE *имя*);

4) очистить оперативную память командой SCR;

5) вызвать в оперативную память программу с ВЗУ командой OLD (OLD *имя*);

6) запустить программу на счет командой RUN;

7) сравнить полученный ответ с приведенным в задании. Если ответ неверен, перейти к п. 8, иначе к п. 12;

8) высветить текст программы на экране дисплея командой LIST;

9) найти и исправить ошибки в программе;

10) записать новую версию программы на ВЗУ командой REPLACE (REPLACE *имя*);

11) перейти к п. 4;

12) распечатать на ПУ текст программы командой SAVE LP;

13) модифицировать программу с целью вывода результатов работы программы на ПУ;

14) запустить программу на счет командой RUNNH (отметить различия в выполнении команд RUN и RUNNH);

15) уничтожить на ВЗУ свою программу командой UNSAVE (UNSAVE *имя*);

16) очистить оперативную память командой SCR. Варианты задач приведены ниже.

З а м е ч а н и я. 1. Во всех пунктах задания имя программы не должно изменяться.

2. В задании приведены команды применительно к ЭВМ «Электроника-60» (отличия для СМ-4 и ДВК-2 приведены в приложении).

*Варианты задач I уровня.*

**1.**
```
10 T=1
20 PRINT T,T*E
30 T=T+2
40 IF T<=10 GO TO 20
50 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 1 | 1 |
| 3 | 9 |
| 5 | 25 |
| 7 | 49 |
| 9 | 81 |

**2.**
```
10 F=9
20 PRINT 2*F,F*F
30 F=F+3
40 IF F<=15 GO TO 20
50 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 0 | 0 |
| 6 | 9 |
| 12 | 36 |
| 18 | 81 |
| 24 | 144 |
| 30 | 225 |

**3.**
```
10 H=1
20 PRINT H,H+1
30 H=H+1
40 IF H<=7 GOTO 20
50 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| 6 | 7 |
| 7 | 8 |

**4.**
```
10 J=4
20 PRINT J,J*7
30 J=J+2
40 IF J<=20 GOTO 30
50 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 4 | 28 |
| 6 | 42 |
| 8 | 56 |
| 10 | 70 |
| 12 | 84 |
| 14 | 98 |
| 16 | 112 |
| 18 | 126 |
| 20 | 140 |

**5.**
```
10 A=1
20 PRINT A,A+10
30 A=A+2
40 IF A<=6 GOTO 20
50 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 1 | 11 |
| 2 | 12 |
| 3 | 13 |
| 4 | 14 |
| 5 | 15 |
| 6 | 16 |

**6.**
```
10 P=1
20 M=2
30 P=P+M
40 PRINT M,P
50 M=M+1
60 IF M<=6 GOTO 30
70 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 2 | 2 |
| 3 | 6 |
| 4 | 24 |
| 5 | 120 |
| 6 | 720 |

7.
```
10 S=0
20 I=1
30 S=S+I*I
40 PRINT I,S
50 I=I+1
60 IF I<=7 GOTO 30
70 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 1 | 1 |
| 2 | 3 |
| 3 | 6 |
| 4 | 10 |
| 5 | 15 |
| 6 | 21 |
| 7 | 28 |

8.
```
10 P=0
20 K=1
30 P=P*K
40 PRINT K,P
50 K=K+1
60 IF K<=5 GOTO 30
70 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 1 | 1 |
| 2 | 5 |
| 3 | 14 |
| 4 | 30 |
| 5 | 55 |

9.
```
10 X=2
20 Y=X*X*X
30 PRINT X,Y
40 X=X+2
50 IF X<=12 GOTO 20
60 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 2 | 8 |
| 4 | 64 |
| 6 | 216 |
| 8 | 512 |

10.
```
10 C=0.27
20 H=50
30 PRINT H,H*C
40 H=50+50
50 IF H<250 GOTO 30
60 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 50 | 13.5 |
| 100 | 27 |
| 150 | 40.5 |
| 200 | 54 |
| 250 | 67.5 |

11.
```
10 S=0
20 I=1
30 A=1
40 A=A+2
50 S=S+A
60 PRINT I,A,S
70 I=I+2
80 IF I<=5 GOTO 40
90 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | | |
|---|---|---|
| 1 | 2 | 2 |
| 2 | 4 | 6 |
| 3 | 8 | 14 |
| 4 | 16 | 30 |
| 5 | 32 | 62 |

12.
```
10 H=1
20 PRINT H,0.4*H
30 H=H+1
40 IF H>=6 GOTO 20
50 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | |
|---|---|
| 1 | 0.4 |
| 2 | 0.8 |
| 3 | 1.2 |
| 4 | 1.6 |
| 5 | 2 |
| 6 | 2.4 |

13.
```
10 I=1
20 S=0
30 A=-1
40 S=S+A*I*I
41 PRINT I;A;S
50 A=-A-A
60 I=I+1
70 IF I<=6 GOTO 40
80 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

| | | |
|---|---|---|
| 1 | -1 | -1 |
| 2 | 1 | 3 |
| 3 | -1 | -6 |
| 4 | 1 | 10 |
| 5 | -1 | 15 |
| 6 | 1 | 21 |

14.
```
10 P=1
20 PRINT P;"*12=";P*12
30 P=P+1
40 IF P<=6 GO TO 30
50 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

```
1*12=12
2*12=24
3*12=36
4*12=48
5*12=60
6*12=72
```

15.
```
10 X=7
20 P=1
30 P=P*X
40 PRINT X
50 X=X+1
60 IF X<=10 GOTO 30
70 STOP
```

ПРАВИЛЬНЫЕ ОТВЕТЫ

```
 7     7
 8    56
 9   504
10  5040
```

Задание II уровня. Предназначено для приобретения навыков работы с файлами последовательного доступа.

Для выполнения задания следует составить две программы: первая программа — для создания файла последовательного доступа, вторая — для обработки данных, записанных в файле последовательного доступа. Если не указано в задании, имя файла задать самостоятельно. Подготовить тесты. Проверить работу программ на ЭВМ.

При выполнении заданий использовать, если необходимо, типовые алгоритмы работы с массивами (см. введение к работам 4, 8).

*Варианты задач II уровня.*

1. Протокол лыжных гонок записать на ВЗУ в файл "SK1". Для каждого участника вводится фамилия, время старта (часы, минуты, секунды), время финиша. Используя сформированный файл, вывести на экран дисплея фамилии участников, выполнивших норму ГТО.

2. Сформировать файл "BOOK" из фамилий любимых писателей учеников класса (ученики по очереди вводят по три фамилии писателей). Используя сформированный файл, напечатать фамилии пяти наиболее популярных писателей.

3. Сформировать файл «EXAM» по результатам сдачи сессии (три экзамена) группой студентов. Информация об одном студенте вводится в символьном виде в следующем порядке: *фамилия*$*N_1*N_2*N_3$, где $N_i (i=1, 2, 3)$ — оценки.

Используя сформированный файл, результаты сессии вывести на экран в виде таблицы. В шапке таблицы вывести названия предметов.

4. Задана разреженная матрица (размером не более чем 10××10), состоящая из нулей и единиц. Сформировать файл "MLIST", в котором запомнить матрицу в следующем виде: количество строк, количество столбцов матрицы, далее номера строк и столбцов (I, J), на пересечении которых находятся ненулевые элементы. Используя сформированный файл, вывести матрицу на экран в привычном виде.

5. В задаче 4 ненулевые элементы матрицы могут иметь любые значения. При формировании файла на ВЗУ запоминать для каждого ненулевого элемента также его значение.

6. Сформировать файл "FREND" из фамилий и дат (день, месяц и год) рождения Ваших друзей. Используя сформированный файл, вывести фамилии тех, кто родился летом (июнь, июль, август).

7. Сформировать файл "NAME", в котором хранится список имен. Используя сформированный файл, вывести на экран дисплея имена, начинающиеся с буквы *А*.

8. Сформировать файл "STUD", имеющий следующую структуру: фамилия студента, пол (одна буква *М* или *Ж*), год рождения. Напечатать список студентов мужского пола с указанием их возраста. В конце списка напечатать средний возраст.

9. Сформировать файл "CAR", содержащий информацию об автолюбителях: фамилия, марка автомобиля, цвет. Используя сформированный файл, вывести на экран дисплея сведения об автолюбителях, имеющих автомобиль «Жигули» зеленого цвета.

10. Сформировать файл о студентах одного курса (фамилия, пол, год рождения, месяц рождения). Используя сформированный файл, вывести на экран дисплея фамилии студентов мужского пола, родившихся летом (июнь, июль, август).

11. Сформировать файл, содержащий информацию о поездах, отправляющихся с Ленинградского вокзала г. Москвы (номер поезда, станция назначения, время отправления, время в пути). Используя сформированный файл, вывести на экран дисплея информацию о поездах, отправляющихся в Ленинград от 21 до 24 часов.

12. В задаче 11 вывести информацию о поездах, следующих в г. Ленинград и находящихся в пути менее 8 часов.

13. Сформировать файл, содержащий информацию о бытовых магнитофонах: марка магнитофона, его стоимость. Используя сформированный файл, вывести информацию о магнитофонах стоимостью менее 200 руб.

14. Сформировать файл, содержащий данные о книгах по программированию Вашей личной библиотеки (фамилия автора и его инициалы, название книги, название издательства, год издания). Используя сформированный файл, вывести на экран дисплея фамилии авторов и названия книг, выпущенных издательством «Мир».

15. Сформировать файл, содержащий сведения о магазинах Вашего города (района): название магазина (универмаг, продукты, ткани и т. п.), его номер, адрес. Используя сформированный файл, вывести на экран дисплея информацию обо всех магазинах «Книги».

Задание III уровня. Предназначено для приобретения навыков работы с файлами прямого доступа.

Для выполнения заданий следует составить две программы: первая — для формирования файла (файлов), вторая — для обработки данных, хранящихся в файле (файлах). Имена файлов задать самостоятельно. При выполнении заданий использовать типовые алгоритмы, приведенные в работах 4,8. Подготовить тесты. Проверить работу программ на ЭВМ.

*Варианты задач III уровня.*

1. В шахматном турнире принимает участие 10 шахматистов. Сформировать файл фамилий участников и файл результатов (в виде матрицы): победа — 1, ничья — 0,5, проигрыш — 0 очков. Используя сформированные файлы, вывести на экран фамилии участников турнира и количество набранных ими очков в порядке занятых мест.

2. В чемпионате по футболу принимает участие 16 команд. Сформировать файл названий команд и файл результатов игр (в виде матрицы): выигрыш — 2, ничья — 1, проигрыш — 0 очков. Используя сформированный файл, обработать результаты чемпионата и вывести на экран дисплея названия трех лучших команд (по числу очков) и количество побед каждой команды.

3. В классном журнале (в классе не более 20 учеников) каждый день по каждому предмету отмечается пропуск занятия или выставляется оценка за ответ. Сформировать на ВЗУ копию классного журнала по одному предмету за одну учебную четверть (два месяца) в виде двух файлов: фамилий и оценок (пропусков). Файл оценок формировать в виде матрицы. Наличие пропуска можно кодировать цифрой 6. Используя сформированные файлы, вывести на экран дисплея фамилии учеников, не имеющих за четверть ни одной оценки 2.

4. В задаче 3, используя сформированные файлы, выставить оценки за четверть, усредняя оценки, полученные каждым учеником, и округляя их до ближайшего целого. Если оценок нет, выставить 6 — не аттестован.

5. В задаче 3, используя сформированные файлы, вывести на экран дисплея фамилии учеников, имеющих пропуски занятий в порядке убывания количества пропусков.

6. К 20 спортивным журналистам обратились с просьбой назвать трех лучших хоккеистов сезона. Информация от каждого журналиста поступает независимо от других, вводится в файл прямого доступа. Используя сформированный файл, определить лучшего хоккеиста сезона по сумме очков (за первое место — 3, за второе место — 2, за третье место — 1 очко).

7. Сформировать два файла прямого доступа по результатам подписки на газеты и журналы (всего 10 наименований): файл наименований изданий и числовой файл в виде матрицы $2\times10$ (первая строка — стоимость подписки на год, вторая — количество подписчиков). Используя сформированные файлы, выдать справку об общем количестве подписчиков и общей сумме полученных денег.

8. Для задачи 7 напечатать, какое издание пользуется наибольшей популярностью у подписчиков.

9. Заданная разреженная матрица размером $20\times20$ состоит из нулей и натуральных чисел. Запомнить эту матрицу на ВЗУ в виде файла прямого доступа (матрица $20\times20$). Используя сформированный файл, вывести матрицу на экран дисплея в виде таблицы: первая графа — значение ненулевого элемента, вторая — номер строки, третья — номер столбца, на пересечении которых находится ненулевой элемент.

10. Во время баскетбольной игры формируется файл прямого доступа — одномерный массив (номер элемента — номер игрока). В массив заносится количество очков, набранных каждым игроком одной команды. В команде всего 12 игроков. Их фамилии хранятся в другом файле прямого доступа, формируемом перед началом игры. После окончания игры, используя сформированные файлы, вывести на экран фамилию лучшего игрока и количество набранных им очков.

11. В задаче 10, используя сформированные файлы, вывести на экран фамилии игроков в порядке убывания набранных ими очков.

12. На ВЗУ формируется файл прямого доступа (матрица $5\times100$) с результатами проведенных тиражей «Спортлото: 5 из 36» (см. теоретическое введение к работе 12, программа 5.11). После проведения очередного тиража результаты вводятся в очередной столбец файла. Используя сформированный файл, вывести на экран дисплея частоту появления каждого из 36 номеров во всех проведенных тиражах.

13. Сформировать файл, содержащий результаты сессии студентов одной группы в виде матрицы, в каждой строке которой хранятся результаты сессии (пять экзаменов) для одного студента. В группе 25 студентов, их фамилии хранятся в отдельном файле. Используя сформированный файл, вывести на экран фамилии отличников.

14. Для задачи 13 сформировать дополнительный файл названий дисциплин. Используя сформированные файлы, вывести на экран фамилии студентов, имеющих двойки (указать, по каким предметам).

15. В задаче 14, используя сформированные файлы, вывести на экран название дисциплины, по которой больше всего студентов получили двойки.

Вопросы для самопроверки.

1. Как задать имя программы, которую предполагается вводить в оперативную память? Какое имя имеет программа, если ее имя не задано? Как изменить имя программы, находящейся в оперативной памяти?
2. Как сохранить программу на диске? Под каким именем программа хранится на диске?
3. Как изменить текст программы, хранящейся на диске? Как стереть программу с диска?
4. Как распечатать текст программы на ПУ?
5. В чем различие в выполнении команд RUN и RUNNH?
6. Что такое файл данных последовательного доступа? Как его организовать? Что значит открыть файл?
7. Как осуществляется формирование файла последовательного доступа? Что такое запись? Что означает закрыть файл? Когда формируется признак конца файла?
8. Как осуществляется считывание данных из файла последовательного доступа? Что такое переход по концу файла и когда он применяется?
9. Что такое файл данных прямого доступа? Как организовать файл данных прямого доступа? Как осуществляется формирование файла, как осуществляется считывание данных из файла прямого доступа?
10. В чем сходство и отличие в выполнении операторов CLOSE #N, CLOSE и END? Каковы функции этих операторов? Когда и где их нужно использовать?

# Глава 5. ПРАКТИКА ПРОГРАММИРОВАНИЯ НА БЕЙСИКЕ

## *Работа 11.* СПЕЦИАЛЬНЫЕ ПРИЕМЫ РАБОТЫ С ЦЕЛЫМИ ЧИСЛАМИ

Теоретическое введение. В работе рассматриваются некоторые типовые задачи обработки целых чисел, приводятся алгоритмы решения ряда типовых задач, а также программы, реализующие эти алгоритмы.

1. *Определение четности.* Требуется определить, является заданное целое число N четным или нечетным. Результат («чет» или «нечет») присваивается символьной переменной S□.

Для решения задачи можно использовать следующее соображение: если число N четное, то оно кратно двум и значение N/2 совпадает со значением [N/2] — целая часть N/2 (программа 5.1).

*Список используемых переменных.* Исходные данные: N — целое число.

Результат: S□ — символьная переменная, которой присваивается значение «чет» или «нечет».

**Программа 5.1. Определение четности**

```
10 INPUT N
20 S$="НЕЧЕТ"
30 IF (N/2)=INT(N/2) THEN S$="ЧЕТ"
40 PRINT S$
50 STOP
```

*2. Определение суммы цифр числа.* Требуется найти сумму цифр заданного натурального числа N. Результат присвоить переменной S.

Для получения суммы цифр целого числа N будем выделять цифры этого числа, начиная с младшей, и накапливать их сумму в переменной S. Для выделения самой правой цифры числа вычислим N—[N/10]*10. Для выделения следующей цифры рассмотрим новое число N=[N/10], в котором следующая по порядку цифра исходного числа является младшей, и применим к этому числу описанную выше процедуру. Процесс будем продолжать до тех пор, пока очередное новое значение N не будет равно нулю.

При нахождении суммы цифр отрицательного числа нужно предварительно переменной N присвоить значение |N|.

В приведенной ниже программе 5.2, составленной в соответствии с этим алгоритмом, определяется также количество цифр числа N.

*Список используемых переменных.* Исходные данные: N — целое число.

Результат: S — сумма цифр числа N, I — количество цифр числа N.

**Программа 5.2.** Определение суммы цифр числа

```
10 INPUT N
20 N=ABS(N)
30 S=0 \ I=0
40 IF N=0 GO TO 90
50 S=S+N-(INT(N/10))*10
60 I=I+1
70 N=INT(N/10)
80 GO TO 40
90 PRINT "В ИСХОДНОМ ЧИСЛЕ ВСЕГО ";I;" ЦИФР"
100 PRINT "СУММА ЦИФР S=";S
110 STOP
```

Пояснения к программе. При выполнении программы 5.2 исходное число N портится. Если требуется сохранить его для дальнейших вычислений, необходимо сразу после ввода переслать его значение в другую переменную, используя, например, оператор

15 N1=N

При определении суммы цифр числа значение переменной I изменяется на единицу (строка 60), как только выделена очередная цифра числа N.

3. *Нахождение делителей числа.* Требуется найти и напечатать все делители целого числа N, включая 1.

Число A является делителем числа B, если B делится на A нацело. Будем проверять, являются ли делителями исходного числа N числа начиная с двух и до [N/2], так как единица является делителем любого целого числа. Проверку будем осуществлять в цикле. Значение каждого найденного делителя будем печатать. Для проверки того, является ли число I делителем, используем алгоритм, аналогичный алгоритму определения четности (программа 5.3).

*Список используемых переменных.* Исходные данные: N — целое число, делители которого нужно найти и напечатать.

Результат: значения I, для которых выполняется условие N/I=[N/I].

**Программа 5.3.** Нахождение делителей числа

```
10 INPUT N
20 C=INT(N/2)
30 PRINT "ДЕЛИТЕЛИ ЧИСЛА ";N
40 PRINT 1;
50 FOR I=2 TO C
60 IF (N/I)<>INT(N/I) GO TO 80
70 PRINT I;
80 NEXT I
90 PRINT N
100 STOP
```

Пояснения к программе. Чтобы исключить необходимость многократного вычисления выражения [N/2] при проверке условия выхода из цикла, в программе значение этого выражения вычисляется до входа в цикл и присваивается переменной C, которая и используется далее в операторе FOR.

Определяемые в программе значения делителей числа N не сохраняются. Можно видоизменить программу так, чтобы делители сохранялись в памяти. Для этого необходимо предусмотреть массив (например, D) достаточного размера (размер массива определяется ограничением на N) и после получения очередного делителя числа N пересылать его в очередной элемент массива, добавив, например, следующие дополнительные строки в программу:

```
5 DIM D(100)
6 J=1 \ D(1)=1
65 J=J+1 \ D(J)=I
```

Программа, дополненная указанными операторами, будет выполняться только для таких N, число делителей которых не больше 100.

4. *Нахождение простых чисел.* Требуется найти все простые числа в интервале от 1 до M и напечатать их.

Алгоритм «Решето Эратосфена». Метод, известный под названием «Решето Эратосфена», заключается в следующем. В последовательности натуральных чисел от 1 до M вычеркнем (заменим на нуль) каждое второе число, начиная с 2 (кроме самого числа 2). Затем найдем первое ненулевое число в последовательности. Это будет число 3. Вычеркнем (заменим на нуль) каждое третье число в последовательности, начиная с числа 3 (кроме самого числа 3) и т. д. В итоге невычеркнутыми (ненулевыми) числами в последовательности будут только простые числа от 1 до M.

Для реализации алгоритма на ЭВМ сформируем одномерный массив A, в который поместим целые числа от 1 до M (значение элемента массива равно индексу элемента). Размер массива A (и соответственно максимальное значение M) определяется объемом оперативной памяти ЭВМ, на которой реализуется указанный алгоритм.

Далее просматриваем элементы массива, начиная с I=2, до M. Если A(I)≠0, то его значение выводится на печать, а весь «хвост» массива преобразуется. Это преобразование заключается в замене нулями элементов с индексами I+I, I+2*I и т. д. до конца массива, т. е. при I=2 обнуляем каждый второй элемент массива, начиная с четвертого, при I=3 — каждый третий, начиная с шестого и т. д. После этого I увеличивается на 1, и ищется следующий элемент массива, отличный от нуля (программа 5.4).

*Список используемых переменных.* Исходные данные: M — натуральное число (верхняя граница интервала), M⩽8000 (такое

ограничение определяется объемом оперативной памяти ЭВМ «Электроника-60», на которой была реализована приведенная программа).

Результат: ненулевые элементы массива A, начиная с A(2).

Вспомогательные переменные: A — массив размером 8000, каждый элемент массива равен значению индекса этого элемента; I, C — управляющие переменные внешнего и внутреннего циклов.

**Программа 5.4.** «Решето Эратосфена»

```
10 DIM A(8000)
20 PRINT "ВВЕДИТЕ M<=8000"
30 INPUT M
40 IF M>8000 GO TO 20
50 FOR I=1 TO M \ A(I)=I \ NEXT I
60 PRINT "ПЕЧАТАЕМ ПРОСТЫЕ ЧИСЛА"
70 FOR I=2 TO M
80 IF A(I)=0 GO TO 110
90 PRINT A(I); \ C=I+I
100 FOR J=C TO M STEP I \ A(J)=0 \ NEXT J
110 NEXT I
120 PRINT
130 STOP
```

Пояснения к программе. Оператор в строке 40 осуществляет контроль вводимых исходных данных.

Операторы в строке 50 заполняют массив A целыми числами от 1 до M.

Операторы в строках 70—110, образующие внешний цикл по I, осуществляют преобразование исходного массива к виду, когда все элементы, не являющиеся простыми числами, заменяются на нули (вычеркиваются).

Во внутреннем цикле (строка 100) осуществляется замена нулями всех элементов, кратных I, кроме самого I-го элемента массива, если очередной (I-й) элемент массива не нулевой (преобразование «хвоста» массива).

При работе программы найденные простые числа сразу выводятся на печать (строка 90) и сохраняются в памяти ЭВМ как ненулевые элементы массива A. При необходимости можно переслать найденные простые числа в отдельный массив (см. пояснения к программе 5.3).

Рассмотренный метод имеет ограниченное применение. Например, его использование нецелесообразно

— для определения, является ли натуральное число N простым;

— для поиска простых чисел в интервале от A до B;

— для определения простых чисел при больших значениях M из-за ограниченного объема оперативной памяти ЭВМ.

Другой алгоритм нахождения простых чисел. Требуется найти все простые числа в интервале от A до B. Воспользуемся тем, что если целое число I не имеет ни одного

делителя в интервале от 2 до I, то это число I — простое. Для уменьшения объема вычислений будем искать простые числа в интервале от A до B только среди нечетных чисел, если A>2. По этой же причине в качестве делителей числа I (I∈[A, B]) будем проверять только нечетные числа, начиная с 3. Кроме того, легко убедиться в том, что проверку достаточно осуществлять до $[\sqrt{I}]$, а не до I (см. программу 5.5).

*Список используемых переменных.* Исходные данные: A, B — границы интервала поиска простых чисел.

Результат: Значения I, для которых не найдено делителей.

Вспомогательные переменные: $C=[\sqrt{I}]$; I, J — управляющие переменные внешнего и внутреннего циклов.

**Программа 5.5.** Нахождение простых чисел в интервале от A до B

```
10 PRINT "ВВЕДИТЕ А И В (0 <А < В)"
20 INPUT A,B
30 IF A<=0 GO TO 10
40 IF A>=B GO TO 10
50 PRINT "ПРОСТЫЕ ЧИСЛА ОТ";A;" ДО ";B
60 IF A>2 GO TO 100
70 IF A=2 GO TO 90
80 PRINT 1; \ A=A+2
90 PRINT 2;
100 IF A/2<>INT(A/2) GO TO 120
110 A=A+1
120 FOR I=A TO B STEP 2
130 C=INT(SQR(I))
140 FOR J=3 TO C STEP 2
150 IF (I/J)=INT(I/J) GO TO 170
160 NEXT J \ PRINT I;
170 NEXT I
180 PRINT
190 STOP
```

Пояснения к программе. Операторы в строках 30—40 осуществляют контроль вводимых данных.

Операторы 60—90 проверяют, равно ли A единице или двойке. Если A=1, то печатаются простые числа 1, 2; если A=2, то печатается простое число 2.

Операторы в строках 100—110 проверяют, является ли левая граница интервала A четным числом. Если A — четное (т. е. не простое) число, то его значение увеличивается на единицу.

Во внешнем цикле в интервале от A до B (строки 120—170) просматриваются только нечетные числа из заданного интервала.

Во внутреннем цикле в качестве делителей числа I проверяются только нечетные числа от 3 до $[\sqrt{I}]$.

Оператор PRINT в строке 180 осуществляет возврат каретки.

*5. Обработка многоразрядных чисел.* Диапазон чисел, которые можно представить в ячейках памяти ЭВМ, ограничен. Для работы с числами, лежащими вне этого диапазона, требуется применение специальных приемов для размещения таких чисел в памяти и их обработки.

Для представления многоразрядного числа в памяти ЭВМ можно каждую его цифру хранить как элемент массива: самую младшую цифру — количество единиц — в первом элементе массива, количество десятков — во втором и т. д. Старшие незанятые элементы массива следует заполнить нулями *). Рассмотрим типовые задачи обработки многоразрядных чисел.

*5.1. Сложение.* Требуется сложить два целых $n$-разрядных числа ($n \leqslant 20$). Каждое число считается представленным в виде одномерного массива. Результат сложения также будем получать в одномерном массиве.

Сложение будем осуществлять аналогично сложению двух чисел «столбиком», т. е. будем складывать одноразрядные цифры двух слагаемых, начиная с младшего разряда. Из полученной суммы двух цифр каждый раз будем выделять цифру соответствующего разряда суммы чисел и формировать перенос в следующий разряд. По окончании суммирования значение переноса (если он не равен нулю) считаем за самую старшую цифру суммы. Следует иметь в виду, что при сложении двух $n$-разрядных чисел сумма может оказаться $n+1$-разрядным числом (программа 5.6).

*Список используемых переменных.* Исходные данные: A, B — массивы для хранения цифр слагаемых размером 20; N — количество цифр в большем слагаемом ($N \leqslant 20$), после выполнения суммирования — количество цифр суммы.

Результат: C — массив для хранения цифр суммы размером 21.

Вспомогательные переменные: D — значение суммы двух I-х цифр, P — значение переноса в старший разряд, I — счетчик разрядов — управляющая переменная цикла.

Пояснения к программе. Операторы 40, 50 осуществляют контроль вводимого значения N.

При одном выполнении оператора INPUT (строка 80) вводятся две цифры одного разряда двух слагаемых. Причем цифры нужно вводить, начиная с младшего разряда. Если в одном слагаемом цифр меньше, то в качестве недостающих старших цифр следует вводить нули.

В строке 90 значению переноса в старший разряд P присваивается 0 для того, чтобы вычисление суммы младших цифр слагаемых (при первом выполнении строки 110) осуществлялось так же,

---

*) В одном элементе массива можно хранить и несколько цифр, но в этом случае алгоритмы обработки чисел усложняются.

**Программа 5.6.** Сложение многоразрядных чисел

```
10 DIM A(20),B(20),C(21)
20 PRINT "КОЛИЧЕСТВО РАЗРЯДОВ В БОЛЬШЕМ СЛАГАЕМОМ ? (НЕ БОЛЬШЕ 20)"
30 INPUT N
40 IF N>20 GO TO 20
50 IF N<=0 GO TO 20
60 PRINT "ВВЕДИТЕ ПООЧЕРЕДНО ЦИФРЫ МНОГОРАЗРЯДНЫХ ЧИСЕЛ,"
70 PRINT "НАЧИНАЯ С МЛАДШЕЙ"
80 FOR I=1 TO N \ INPUT A(I),B(I) \ NEXT I
90 P=0
100 FOR I=1 TO N
110 D=A(I)+B(I)+P
120 P=INT(D/10)
130 C(I)=D-P*10
140 NEXT I
150 IF P=0 GO TO 170
160 N=N+1 \ C(N)=P
170 FOR J=N TO 1 STEP -1
180 PRINT C(J);
190 NEXT J
200 PRINT
210 STOP
```

как и для всех остальных цифр, когда значение переноса в старший разряд формируется в строке 120.

В цикле по I (строки 100—140) формируется массив — сумма (C). Если после сложения старших цифр слагаемых перенос в старший разряд не равен нулю, то значение переноса заносится в старший разряд суммы (количество цифр суммы — элементов массива C — становится на 1 больше, чем количество цифр в наибольшем из слагаемых (строка 160)).

Чтобы напечатать результат в привычном виде, вывод элементов массива C осуществляется, начиная с конца — со старшего разряда (строки 170—190).

Замечание. Для ввода многоразрядных чисел можно использовать также прием, применяемый при обработке данных, количество которых заранее не известно (см. п. 6 главы 2), используя для окончания ввода, например, какое-либо отрицательное число. Этот способ демонстрируется далее в программе 5.7.

*5.2. Умножение.* Требуется умножить $n$-разрядное число ($n \leqslant 20$) на целое число K ($0 < K \leqslant 100$).

Считаем, что $n$-разрядное число представлено в виде одномерного массива.

Умножение будем осуществлять аналогично умножению «столбиком». После перемножения $i$-й цифры $n$-разрядного числа и числа K прибавляем к результату перенос из предыдущего разряда и выделяем из полученной суммы количество единиц. Этот и будет $i$-я цифра произведения. После умножения $n$-й цифры $n$-разрядного числа на число K и выделения $n$-й цифры произведения оставшийся перенос записываем поразрядно в массив, предназначенный для про-

изведения, начиная с $(n+1)$-го элемента, выделяя цифры согласно алгоритму «Сумма цифр числа» (см. п. 2 и программу 5.7).

*Список используемых переменных.* Исходные данные: A — массив для хранения цифр заданного многоразрядного числа, размером 20; K — целое число, сомножитель ($K \leqslant 100$).

Результат: B — массив — произведение размером 22.

Вспомогательные переменные: N — количество разрядов исходного числа, F — переменная для ввода очередной цифры многоразрядного числа, P — значение переноса в старший разряд, C — значение суммы произведения $i$-й цифры $n$-разрядного числа на число K и переноса в $i$-й разряд, I, J — управляющие переменные циклов.

**Программа 5.7.** Умножение многоразрядного числа на число.

```
10 DIM A(20),B(22)
20 REM ВВОД МНОГОРАЗРЯДНОГО ЧИСЛА
30 I=0
40 PRINT "ВВОДИТЕ ЧИСЛО , НАЧИНАЯ С МЛАДШЕЙ ЦИФРЫ"
50 IF I)20 THEN PRINT "КОЛ-ВО ЦИФР ПРЕВЫШАЕТ РАЗМЕР МАССИВА" \ STOP
60 PRINT "ВВЕДИТЕ ОЧЕРЕДНУЮ ЦИФРУ , ДЛЯ ОКОНЧАНИЯ"
70 PRINT "ВВОДА ИСПОЛЬЗУЙТЕ ОТРИЦАТЕЛЬНОЕ ЧИСЛО"
80 INPUT F
90 IF F<0 GO TO 120
100 I=I+1 \ A(I)=F
110 GO TO 50
120 N=I
130 PRINT "ВВЕДИТЕ K НЕ БОЛЬШЕЕ 100"
140 INPUT K
150 REM НАЧАЛО ЦИКЛА УМНОЖЕНИЯ
160 P=0
170 FOR I=1 TO N
180 C=A(I)*K+P
190 P=INT(C/10)
200 B(I)=C-P*10
210 NEXT I
220 REM ЗАНЕСЕНИЕ В МАССИВ В ОСТАВШИХСЯ ЦИФР ПЕРЕНОСА
230 J=N
240 IF P=0 GO TO 290
250 J=J+1
260 B(J)=P-(INT(P/10))*10
270 P=INT(P/10)
280 GO TO 240
290 FOR L=J TO 1 STEP -1
300 PRINT B(L);
310 NEXT L
320 PRINT
330 STOP
```

Пояснения к программе. Ввод многоразрядного числа в массив A осуществляется в цикле, образованном операторами 30—110. Каждая цифра вводится в переменную F и после проверки условия окончания ввода (строка 90) заносится в соответствующий элемент массива A. После окончания ввода количество введенных цифр содержится в переменной I, значение которой пересы-

дается в переменную N (строка 120). Количество разрядов числа($N$,) таким образом, не является в приведенной программе исходным данным, а определяется как результат выполнения цикла ввода.

Перед началом выполнения цикла умножения значение переноса в старшие разряды устанавливается равным нулю для того, чтобы умножение младшей цифры на число K выполнять так же, как и умножение других цифр, когда перенос может возникнуть в результате умножения предыдущей цифры. Оператор 200 заносит в I-й элемент произведения младшую цифру результата умножения I-й цифры многоразрядного сомножителя на число K.

Операторы в строках 230—280 заносят цифры последнего переноса в старшие разряды произведения, начиная с (N+1)-го. После завершения цикла значение J определяет число цифр произведения.

Печать массива-произведения (строки 290—310) осуществляется, начиная со старшей цифры произведения (J-й элемент массива), в обратном порядке (используется отрицательный шаг).

Перед пересылкой P в формируемый массив-произведение B (строки 240—280) количество (N) заполненных к этому моменту элементов массива B пересылается в переменную J (строка 230), чтобы не портить последующими операторами значение N, равное количеству цифр исходного многоразрядного числа.

Пример 1. Определить, каким днем недели было 12 апреля 1961 года, когда был впервые осуществлен полет человека в космос.

Решение. Задачу будем решать в общем виде. Требуется по дате, заданной как день месяца $d$, номер месяца $n$ и год G, определить, на какой день недели выпала эта дата.

Закодируем дни недели при помощи чисел (0 — воскресенье, 1 — понедельник, 2 — вторник, 3 — среда, 4 — четверг, 5 — пятница, 6 — суббота).

Введем также следующие обозначения:

$m$ — приведенный номер месяца, определяемый как

$$m=\begin{cases} n-2, & \text{если } n>2, \\ n+10, & \text{если } n\leqslant 2, \end{cases}$$

т. е. $m=1$ — соответствует марту, $m=2$ — апрелю и т. д. Январь и февраль являются 11-м и 12-м месяцами предыдущего года;

P — приведенный год, определяемый как

$$\mathrm{P}=\begin{cases} \mathrm{G}, & \text{если } n>2, \\ \mathrm{G}-1, & \text{если } n\leqslant 2; \end{cases}$$

C — полное число столетий, прошедших к заданному дню;

Y — номер года в текущем столетии.

Таким образом, можно записать $\mathrm{P}=100\cdot\mathrm{C}+\mathrm{Y}$.

Для 12 апреля 1961 года: $d=12$, $n=4$, $m=2$, $\mathrm{G}=1961$, $\mathrm{P}=1961$, $\mathrm{C}=19$, $\mathrm{Y}=61$.

Можно показать (см. О. Оре. Приглашение в теорию чисел.— М., Наука, 1980), что номер дня недели определяется как остаток от деления $w$ на 7, где

$$w = d + [(13m - 1)/5] + Y + [Y/4] + [C/4] - 2C,$$

(квадратные скобки означают взятие целой части; $w$ — количество дней, прошедших с некоторого известного дня, который пришелся на воскресенье).

Составим программу так, чтобы после ввода даты в привычных обозначениях определялись все параметры формулы ($m$, Y, C), вычислялось $w$ и определялся остаток от деления $w$ на 7.

Для повышения наглядности можно перед выводом на печать определять название дня недели. Для этого можно использовать массив размером 7, в котором хранятся названия дней недели (программа 5.8).

*Список используемых переменных.* Исходные данные: D — номер дня в месяце, N — номер месяца в году (1 — январь, 2 — февраль и т. д.), G — год.

Результат: P() — день недели, на который выпала заданная дата.

Вспомогательные переменные: S() — символьный массив размером 7, содержащий названия дней недели; P — приведенный год, первый месяц которого — март; M — приведенный номер месяца (1 — март, 2 — апрель и т. д.); C — количество полных столетий, прошедших к заданному дню; Y — год, заданный двумя младшими цифрами; W — результат вычисления по формуле (см. выше); K — номер дня недели для заданной даты.

Пример 2. Вычислить P=N! при N⩾12.

Решение. При больших N значение N! не может быть представлено точно в ячейке памяти, предназначенной для одного числа. Вспомним (см. п. 3.1 главы 3), что точность представления чисел в бейсике (для рассматриваемых ЭВМ) не более шести значащих цифр.

Уже 12!=538876800 содержит семь значащих цифр и в памяти ЭВМ будет представлено приближенно как $5.38877 \cdot 10^8$.

Чтобы вычисления N! для N⩾12 проводить точно, нужно значение факториала формировать в массиве как многоразрядное число. Вычисление факториала (N!) состоит в повторении умножения (I—1)!·I при I, изменяющемся от 2 до N. Так как (I—1)! представляется каждый раз как многоразрядное число, то составной частью алгоритма вычисления факториала большого числа является алгоритм умножения многоразрядного числа на число, рассмотренный в п. 5.2. Этот алгоритм нужно выполнять во внешнем цикле по I (I=2, . . ., N). В отличие от алгоритма, рассмотренного в п. 5.2, произведение многоразрядного числа на число будем получать в том же массиве, в котором находилось число — сомножитель.

**Программа 5.8.** Определение дня недели

```
10 DIM S$(7)
20 FOR I=1 TO 7
30 READ S$(I) \ NEXT I
40 DATA "ВОСКРЕСЕНЬЕ","ПОНЕДЕЛЬНИК","ВТОРНИК","СРЕДА","ЧЕТВЕРГ
50 DATA "ПЯТНИЦА","СУББОТА"
60 PRINT "ВВЕДИТЕ ЧИСЛО, НОМЕР МЕСЯЦА, ГОД"
70 INPUT D,N,G
80 P=G
90 M=N-2
100 IF N)2 GO TO 120
110 P=P-1 \ M=M+12
120 C=INT(P/100)
130 Y=P-C*100
140 W=D+INT((13*M-1)/5)+Y+INT(Y/4)+INT(C/4)-2*C
150 K=W-7*INT(W/7)
160 PRINT D;".";N;".";G;" - ";S$(K+1)
170 STOP

RUNNH

ВВЕДИТЕ ЧИСЛО, НОМЕР МЕСЯЦА, ГОД
? 12,4,1961
 12 . 4 . 1961  - СРЕДА

STOP AT LINE 170
```

В процессе вычислений может возникнуть возможность переполнения массива, выделенного под факториал. В этом случае вычисления необходимо прекратить и напечатать соответствующее сообщение.

Алгоритм вычисления факториала большого числа реализован в программе 5.9.

*Список используемых переменных.* Исходные данные: N — целое число, факториал которого необходимо вычислить.

Результат: A — массив размером 100 для размещения цифр вычисляемого факториала.

Вспомогательные переменные: I — управляющая переменная внешнего цикла (изменяется от 2 до N); J — текущее число заполненных элементов массива; K — текущий индекс элементов массива — управляющая переменная внутреннего цикла, в котором выполняется умножение многоразрядного числа (I—1)! на число I (изменяется от 1 до J). После завершения цикла K — управляющая переменная цикла вывода на печать; C — результат умножения K-й цифры многоразрядного числа на число I; P — значение переноса в старшие разряды

П о я с н е н и я  к  п р о г р а м м е. До входа в цикл вычисления N! начальное значение факториала устанавливается равным 1 (1!). Это соответствует засылке 1 в первый элемент массива A. Количество элементов массива A устанавливается равным 1 (строка 50).

Программа 5.9. Вычисление N!

```
10 DIM A(100)
15 FOR I=1 TO 100 \ A(I)=0 \ NEXT I
20 PRINT "ВВЕДИТЕ ЧИСЛО N , ДЛЯ КОТОРОГО ВЫЧИСЛЯЕМ N!"
30 INPUT N
40 REM ЦИКЛ ВЫЧИСЛЕНИЯ N!
50 A(1)=1 \ J=1
60 FOR I=2 TO N
70 REM УМНОЖЕНИЕ (I-1)! НА I
75 REM (I-1)! ХРАНИТСЯ В МАССИВЕ A (A(1),...,A(J))
80 P=0
90 FOR K=1 TO J
100 C=A(K)*I+P
110 P=INT(C/10)
120 A(K)=C-P*10
130 NEXT K
140 IF P=0 GO TO 200
150 J=J+1
160 IF J)100 THEN PRINT "НЕ ХВАТАЕТ МЕСТА В МАССИВЕ" \ STOP
170 A(J)=P-INT(P/10)*10
180 P=INT(P/10)
190 GO TO 140
200 NEXT I
205 REM ПЕЧАТЬ РЕЗУЛЬТАТА
210 PRINT TAB(10);N;"!=";
220 FOR K=J TO 1 STEP -1
230 PRINT A(K);
240 NEXT K
250 PRINT
260 STOP
```

Строки 80—190 повторяют строки 160—280 программы 5.7, составленной для умножения многоразрядного числа на число, за исключением строки 230 (J=N). В рассматриваемой программе границей изменения индекса K во внутреннем цикле является переменная J — количество сформированных к данному моменту элементов массива A, и сохранять ее после выхода из цикла по K не нужно. При пересылке в массив A цифр переноса P (строки 140—190) значение J каждый раз увеличивается на 1. Так что при повторном выполнении цикла по K значение J имеет тот же смысл — количество сформированных элементов массива A (или количество цифр в числе (I—1)! при текущем значении I).

После каждого изменения J на 1 проверяется условие переполнения массива. Если J стало больше 100, т. е. для следующей цифры нет места, печатается соответствующее сообщение и выполнение программы прекращения (выполняется оператор STOP) (строка 160).

Вывод результатов на печать осуществляется так же, как в программах 5.6 и 5.7.

Задание I уровня. Требует использования типовых алгоритмов работы с целыми числами, представляемыми как простые переменные (см. пп. 1—4 теоретического введения).

Определить, какой типовой алгоритм (алгоритмы), описанный во введении, может быть использован для решения задачи. Составить алгоритм решения задачи. Типовой алгоритм оформить, как подпрограмму на бейсике. Описать входные и выходные переменные подпрограммы. Составить программу решения задачи с использованием составленной подпрограммы. Подготовить тесты. Проверить работу программы на ЭВМ.

*Варианты задач I уровня.*

1. Для натурального числа $N$ ($N<10^6$) определить:

а) кратно ли оно 3. Использовать следующее свойство: если сумма цифр числа кратна 3, то и число кратно 3;

б) кратно ли оно 5. Использовать следующее свойство: если в числе $N$ последняя цифра 0 или 5, то число кратно 5.

2. Найти все числа в интервале от 1 до 1000, которые совпадают с последними разрядами своих квадратов, например: $5^2=25$, $25^2=625$.

3. Для двух натуральных чисел $P$ и $Q$ ($P$, $Q<10^6$) определить, являются ли они взаимно простыми. Два числа называются взаимно простыми, если они не имеют общих делителей, кроме 1.

4. В интервале от 1 до 1000 найти все парные простые числа. *Парными простыми* числами называют два простых числа, разность между которыми равна 2, например: 3 и 5, 11 и 13, 17 и 19.

5. *Число Армстронга* — такое число из $K$ цифр, для которого сумма $K$-х степеней его цифр равна самому числу. Например: $153=1^3+5^3+3^3$.

Найти все числа Армстронга из двух, трех, четырех цифр.

6. *Палиндром* — такое сочетание цифр, которые читаются одинаково слева направо и справа налево. Например: 121, 55, 4884.

Найти все палиндромы, для которых:

а) их квадраты — также палиндромы (в интервале от 1 до 1000);

б) их кубы — также палиндромы (в интервале от 1 до 100).

7. Напечатать график распределения простых чисел по сотням с указанием их количества в сотне для интервала от 1 до 1000.

8. Счастливым будем считать такое число из шести цифр, в котором сумма левых трех цифр равна сумме правых трех цифр. Например: 457961 : $4+5+7=9+6+1=16$.

Найти все счастливые числа в интервале от $A$ до $B$ и подсчитать их количество. Подсчитать, сколько счастливых чисел — палиндромов находится в интервале от $A$ до $B$.

9. Найти минимальную и максимальную суммы цифр числа в интервале от $A$ до $B$ и построить гистограмму распределения чисел по сумме их цифр ($0<A<B<10^6$).

10. Написать программу нахождения всех совершенных и дружественных чисел в интервале от 1 до $10^6$. *Совершенным* называется такое число, которое равно сумме всех своих делителей, за исклю-

чением самого числа, например, 28=1+2+4+7+14. *Дружественными* числами называется такая пара натуральных чисел $M$ и $N$, для которых сумма всех делителей числа $M$ (кроме числа $M$) равна $N$, а сумма всех делителей числа $N$ (кроме самого числа $N$) равна $M$. Например, числа 220 и 284 дружественные:

сумма делителей 220:

1+2+4+5+10+11+20+22+44+55+110=284,

сумма делителей 284:

1+2+4+71+142=220.

11. Написать программу разложения числа $A$ ($A<10^6$) на простые делители, подсчитывая, сколько раз встречается один и тот же простой делитель, а также вычислить сумму этих делителей.

12. Известно, что любое натуральное число $P$ ($P>7$) можно представить в виде $P=A*3+B*5$. Написать программу нахождения всех пар $A$ и $B$ для числа $P$.

13. В память ЭВМ вводится строка символов вида: *dd.mm.nnnn*, где *dd* — день, *mm* — номер месяца, *nnnn* — год. В программе определить: а) порядковый номер дня в году; б) день недели, на который пришлась эта дата.

14. В обращении имеются монеты 1-, 2-, 3-, 5-, 10-, 15-, 20-, 50-копеечного достоинства. Написать программу, определяющую сдачу наименьшим количеством монет.

15. Заданную дробь, представленную в виде $M/N$ ($M$, $N<10^6$ — натуральные числа), сократить до несократимой дроби.

Указания к решению задач I уровня.

3. Найти все делители числа $P$ и проверить, является ли хотя бы один из них делителем числа $Q$.

4. Найти все простые числа в заданном интервале. Каждое новое простое число сравнивать с предыдущим на парность.

6. Рекомендуем выделенные цифры числа запоминать в массиве, а затем сравнивать попарно 1-ю и последнюю цифры, 2-ю и предпоследнюю и т. д. до $[N/2]$-й цифры, где $N$ — число цифр в числе. После нахождения числа — палиндрома определять, является ли его квадрат (куб) палиндромом.

7. После нахождения простых чисел определить, сколько их входит в каждую сотню, и запомнить эти количества в массиве. В качестве точки графика можно печатать само количество простых чисел в сотне.

8. б) См. указание к решению задачи 6.

9. См. указание к задаче 7.

10. Для каждого числа $N$ сначала найти все его делители (кроме самого $N$), найти их сумму $M$. Если $M\neq N$, то число $N$ не совершенное. Далее найти сумму делителей числа $M$ (кроме самого $M$).

Если сумма делителей числа $M$ равна $N$, то $M$ и $N$ — дружественные числа.

12. Следует организовать полный перебор всех пар $A$ и $B$, при этом $A$ должно изменяться от $[P/3]$ до 0, а $B$ — от 0 до $[P/5]$.

13. При помощи стандартных функций для работы с символьными переменными (см. п. 6 гл. 3) выделить в разные переменные день месяца, номер месяца и номер года. Далее см. пример 1.

14. Для определения размера сдаваемой сдачи сначала определить, сколько требуется для этого монет 50-копеечного достоинства, затем 20-, 15-, 10- и т. д., пока необходимая сумма не будет набрана.

15. Найти все простые делители $M$, а затем каждый делитель с учетом его частоты проверять, является ли он делителем $N$.

Задание II уровня. Содержит более сложные задачи, требующие использования нескольких типовых алгоритмов, описанных во введении. Типовые алгоритмы оформить как подпрограммы. Написать программу решения предложенной задачи. Подготовить тесты. Проверить работу программы на ЭВМ. Проанализировать возможность повышения эффективности программы (уменьшения количества выполняемых операций).

*Варианты задач II уровня.*

1. Написать программу нахождения 10 наименьших натуральных чисел таких, что $X^3=A^3+B^3+C^3$, где $A$, $B$, $C$ — натуральные числа, $A\neq B\neq C$.

2. Найти все натуральные 4-х значные числа, не превосходящие заданного $N$, цифры в которых образуют строго возрастающую последовательность $10^3\leqslant N<10^4$.

3. Для данного натурального числа $A$ ($A<10^6$), заданного в десятичной системе счисления, найти его представление в восьмеричной и двоичной системах счисления.

4. Найти все такие простые числа, не превосходящие заданного числа $N$ ($N<10^6$), двоичная запись которых представляет палиндром.

5. Найти 10 таких натуральных чисел, каждое из которых можно представить как сумму квадратов двух неотрицательных чисел, представленную двумя способами, т. е. $M=I^2+J^2=K^2+L^2$, $I$, $J$, $K$, $L$ — разные числа.

6. Найти все трехзначные числа, сумма цифр которых равна $N$ ($1\leqslant N\leqslant 27$).

7. Построить гистограмму частот выпадания числа 13 на разные дни недели за период с 1920 по 1986 годы.

8. Определить, сколько раз на каждый день недели выпадал ваш день рождения.

9. Построить гистограмму выпадания дня рождения на различные дни недели.

10. Написать программу, печатающую календарь на заданный месяц заданного года.

11. Написать программу, печатающую календарь для заданного года.

12. Написать программу построения графика биоритмов на заданный интервал времени.

13. Студент должен записаться на сдачу экзамена в один из четырех предложенных дней. Написать программу, которая на основании биоритмов поможет ему выбрать нужный день сдачи экзамена.

14. Для выполнения ответственного задания сроком на неделю комиссия решила выбрать одного из пяти претендентов на основе графика их биоритмов. Составьте программу, которая поможет комиссии осуществить выбор.

15. Тренер футбольной команды решил выставить на матч, назначенный на определенный день, команду с учетом графика биоритмов футболистов. В команде 18 футболистов: 2 вратаря, 6 защитников, 5 полузащитников, 5 нападающих. На матч тренер хочет выставить: 1 вратаря, 4 защитников, 3 полузащитников, 3 нападающих.

Указания к решению задач II уровня.

1. Рекомендуется для каждой суммы $S=A^3+B^3+C^3$ находить число $D=\sqrt[3]{S}$, а затем проверять, являются ли числа $[D]$ и $[D]+1$ искомым $X$.

2. Требуется организовать четыре вложенных цикла по каждой из цифр числа.

3. Для перевода числа $A$, заданного в десятичной системе счисления, в систему счисления по основанию $P$ требуется найти коэффициенты $c_i$ в разложении $A$:

$$A=c_n\cdot P^n+\ldots+c_2P^2+c_1P+c_0,$$

где для всех $c_i$ справедливо: $0\leqslant c_i\leqslant P-1$.

Эти коэффициенты определяются как остатки при делении нацело на основание системы счисления $P$ исходного числа $A$ и последующих частных от деления $q_i$

$$q_i=q_{i+1}\cdot P+c_i,\quad i=0,\ 1,\ \ldots,\ n;\quad q_0=A.$$

Деление продолжается до тех пор, пока очередное частное не будет равно 0.

Для ускорения перевода чисел в двоичную систему можно пользоваться следующим свойством двоичной и восьмеричной систем счисления: каждая двоичная триада может быть заменена одной восьмеричной цифрой. Например:

$$(1971)_{10}=(3663)_8=(\underset{3}{11}\ \underset{6}{110}\ \underset{6}{110}\ \underset{3}{011})_2.$$

Поэтому при переводе чисел из десятичной системы счисления в двоичную рекомендуется переводить сначала число в восьмеричную систему, а затем заменять каждую цифру восьмеричного числа ее двоичным эквивалентом.

6. Для решения задачи требуется организовать цикл по двум цифрам ($I$ и $J$) числа $IJK$, а третью цифру ($K$) вычислять как разность $K=N-I-J$. Если $K$ — цифра, то число $IJK$ удовлетворяет условию.

12. Физическое, эмоциональное и умственное состояние изменяются со дня рождения циклически с периодом 23, 28 и 33 дня соответственно. Состояние для $D$-го дня со дня рождения определяется по формуле

$$Y=\mathrm{SIN}(X1),$$

где $X1=(D/X-[D/X])\cdot 2\pi$ ($X=23$, 28 или 33 для физического, эмоционального и умственного состояния).

Для получения графика, начиная со дня, определяемого заданной датой, на $S$ дней нужно определить, каким днем со дня рождения является начальный день (с учетом того, что каждый четвертый год — високосный), и далее организовать цикл по номеру дня от 1 до $S$ от заданной даты, печатая с учетом масштаба каждую точку трех графиков.

13. Необходимо проанализировать умственное состояние за четыре дня. См. также указания к задаче 12.

14. Для каждого из претендентов следует определить значения функции его биоритмов на неделю и рекомендовать комиссии кандидата, имеющего в среднем наилучшее состояние.

15. Для каждого игрока команды вычислить значение функции его физического цикла и рекомендовать тренеру выставить на игру тех игроков, у которых вычисленное значение функции наибольшее.

Задание III уровня. Требует использования алгоритмов работы с многоразрядными числами, приведенными в п. 5 теоретического введения. Необходимо разработать алгоритм решения задачи, используя типовые алгоритмы. Составить программу решения задачи. Подготовить тесты. Проверить работу программы на ЭВМ.

*Варианты вадач* III *уровня.*

1. Написать программу перемножения двух многоразрядных чисел, первое из $M$ цифр, второе — из $N$ ($M$, $N\leqslant 10$).

2. Написать программу вычисления разности двух многоразрядных чисел, уменьшаемое — из $M$ цифр, вычитаемое — из $K$ ($M>K$, $M$, $K\leqslant 50$).

3. Написать программу умножения двух многоразрядных двоичных чисел, первое из $M$ цифр, второе — из $N$ ($M$, $N\leqslant 20$).

4. Написать программу деления нацело двух многоразрядных чисел, делимое — из $M$ цифр, делитель — из $K$ ($M>K$, $M$, $K\leqslant 30$).

5. Написать программу, выполняющую по выбору сложение, вычитание, умножение или деление нацело двух восьмеричных чисел (количество цифр в числе $\leqslant 30$).

6. Дано множество из $N$ натуральных чисел ($N \leqslant 30$). Найти такие две группы из этих чисел, для которых модуль разности между суммами элементов групп минимален. Каждое число множества должно принадлежать какой-нибудь группе.

7. Для каждого ученика класса известен его вес в килограммах. Надо найти все варианты состава двух команд по перетягиванию каната таких, что модуль разности суммарного веса членов команд минимален.

8. Задано 10 произвольных натуральных чисел. Найти все группы по 5 чисел, сумма которых равна заданному числу $M$.

9. На причале ждут посадки $N$ пассажиров вместе с багажом. Для каждого пассажира известен его вес вместе с багажом. Капитан хочет посадить на пароход как можно больше пассажиров. (Сортировку по весам не производить.)

10. Определить частоту появления в числе $N!$ цифры $K$ ($N \leqslant 100$).

11. Найти такое минимальное число $N$, что в числе $N!$ сумма цифр — палиндром.

12. Определить частоты вхождения в число $N!$ ($N \leqslant 100$) цифр, из которых состоит $N!$.

13. Написать программу, выполняющую по выбору сложение или вычитание двух многоразрядных двоичных чисел (количество разрядов $\leqslant 30$).

14. Найти все такие $N$ ($N \leqslant 100$), что у числа $N!$ сумма цифр — простое число.

15. Найти все такие $N$ ($N \leqslant 100$), что у числа $N!$ сумма цифр — квадрат целого числа.

Указания к решению задач III уровня.

5. В программе использовать переход к блокам умножения, сложения, вычитания и деления нацело с помощью оператора ON GO TO.

6. Для решения этой задачи требуется найти все подмножества множества из $n$ элементов. Всего таких подмножеств $2^n$. Но так как нас не интересуют полное и пустое подмножества, то всего необходимо найти $2^n-2$ подмножеств. Так как разбиение осуществляется на две группы, то выделив одну группу, автоматически получаем другую (как дополнение до полного множества). Значит, найдя $2^{n-1}-1$ подмножеств, мы получим всевозможные разбиения исходного множества на 2 группы.

Исходные числа разместить в массиве (например, A). Подмножества, составляющие первую группу, можно выделять, используя двоичные ($n-1$) разрядные числа, начиная с 1. Общее количество таких чисел $2^{n-1}-1$ равно числу возможных разбиений на 2 группы

исходного множества (см. выше). Каждому разряду двоичного числа поставим в соответствие элемент массива А. Если разряд содержит 1, то будем считать, что соответствующий элемент массива А относится к первой группе (иначе ко второй). Для каждого из возможных двоичных ($n$—1) разрядных чисел нужно вычислить сумму S1 соответствующих элементов массива А и определить модуль разности между суммами элементов, относящихся к разным группам, как ABS (S—2*S1), где S — сумма всех чисел, принадлежащих множеству (ее можно вычислить в начале программы). Разбиение, улучшающее критерий, нужно запоминать. Двоичные числа хранить, как многоразрядные числа (одна цифра в одном элементе массива). Каждое следующее число получать, прибавляя 1 к предыдущему двоичному числу.

7. См. указание к задаче 6.

8. 10 чисел, заданных в качестве исходных данных, разместить в массиве (например, А). Построить все пятизначные десятичные числа, цифры которых образуют строго возрастающую последовательность (см. указание к задаче 2 уровня II). Считая каждую цифру (увеличенную на 1) в этих числах индексом элемента массива А, суммировать его элементы, соответствующие цифрам каждого пятизначного числа, и получающуюся сумму сравнивать с М.

9. См. указание к задаче 6.

13. В одном элементе массива хранить одну двоичную цифру. См. также указание к задаче 5.

Вопросы для самопроверки.

1. Какие этапы нужно выполнить при решении задачи на ЭВМ? Как разработать алгоритм решения сложной задачи? Что такое метод пошаговой детализации?

2. Какие простые приемы можно использовать для повышения эффективности программы? Каких правил нужно придерживаться, чтобы программа была удобной для восприятия?

3. Как выделить цифры натурального числа? Каким при этом должно быть условие выхода из цикла?

4. Алгоритм нахождения делителей числа. Объяснить, почему управляющая переменная цикла изменяется до [N/2]? Как обеспечить сохранение найденных делителей в памяти ЭВМ?

5. Алгоритм нахождения простых чисел. Решето Эратосфена. Как нужно задать исходные данные для работы алгоритма? В чем ограниченность применения этого метода?

6. Алгоритм определения, является ли число простым.

7. Алгоритм нахождения простых чисел в заданном интервале. Почему шаг просмотра равен 2? Почему просмотр начинается с первого нечетного числа и как это организуется в программе?

8. Какое число называется многоразрядным? Как представляется многоразрядное число в памяти ЭВМ? Как осуществляется ввод многоразрядного числа, если количество его цифр а) известно перед выполнением программ, б) произвольно?

9. Алгоритм сложения двух многоразрядных чисел.

10. Как организовать вывод многоразрядных чисел, если количество его цифр задано переменной?

11. Алгоритм умножения многоразрядного числа на число.

## *Работа 12.* ПРОГРАММИРОВАНИЕ ИГР

Теоретическое введение. Диалоговый режим работы на ЭВМ при использовании бейсика позволяет реализовывать игровые программы, где ЭВМ генерирует случайные числа, необходимые практически в любой игре, а также может выступать в роли партнера. Разработка игровых программ дает большой эффект при обучении программированию, так как вызывает интерес обучаемых и требует использования разнообразных приемов программирования и средств языка.

В качестве вариантов задач в работе предлагаются игры, сравнительно легко поддающиеся программированию, такие как «Морской бой», «Быки и коровы», «Карусель-лото» (детская «рулетка»), «Набери сумму», «Бросание костей» и некоторые другие. Чтобы избежать громоздких программ, условия некоторых игр упрощены по сравнению с существующими правилами.

Во введении рассматриваются способы получения случайных чисел на ЭВМ.

*1. Получение случайных чисел из интервала (0, 1).* Получение случайных чисел осуществлятся в бейсике при помощи функции RND. Использование этой функции в программе позволяет получать случайные числа, равномерно распределенные в интервале (0; 1). Например, при выполнении оператора

```
10 PRINT RND,RND,RND
```

на экране появится

0.0407319  0.0528293  0.803172

При помощи функции RND выбираются числа из таблицы случайных чисел, заранее составленной и введенной в память ЭВМ. При первом появлении функции RND в программе выбирается первое число из таблицы, при каждом последующем использовании функции RND выбирается следующее число.

Многократное выполнение программы, содержащей строку 10 (см. выше), приведет к одинаковым результатам, так как при каждом новом выполнении программы таблица случайных чисел просматривается с начала.

В некоторых случаях (например, в играх с ЭВМ) желательно при каждом новом выполнении программы иметь другую последовательность случайных чисел. Для этого можно использовать опе-

ратор RANDOMIZE, который выбирает случайным образом точку входа в таблицу случайных чисел, например,

```
10 RANBOMIZE
20 PRINT RND,RND,RND
```

При неоднократном выполнении этой программы на экране будут появляться различные последовательности из трех случайных чисел. Оператор RANDOMIZE размещается в программе перед первым появлением функции RND.

*2. Преобразование интервала изменения случайных чисел (0; 1) в интервале* (A, B). Для получения случайных чисел, принадлежащих открытому интервалу (A, B), можно использовать следующее выражение:

(B—A)*RND+A.

Например, программа

```
10 FOR I=1 TO 5
20 PRINT 2*RND+5
30 NEXT I
```

при выполнении печатает случайные числа, заключенные в интервале (5,7):

```
5.08146
6.05659
6.60634
5.12878
5.31569
```

*3. Получение случайных целых чисел в пределах от* K *до* L. Пусть, например, требуется случайным образом получать целые числа 1, 2, 3, 4, 5 (т. е. K=1, L=5). Для этого каждому целому числу нужно поставить в соответствие 1/5 часть интервала (0;1). Проще всего это сделать, растянув интервал (0;1) в пять раз и сдвинув на 1, т. е. преобразовать интервал (0;1) в интервал (1;6) и применить далее функцию INT (нахождение целой части).

Растянуть интервал (0;1) в пять раз и сдвинуть его на 1 можно при помощи преобразования

5*RND+1.

Когда RND<0.2, первое слагаемое меньше 1 и сумма меньше 2, т. е. применяя к этому значению функцию INT, будем в качестве результата иметь 1. Если 0.2≤RND<0.4, то первое слагаемое больше или равно 1, но меньше 2, и сумма больше или равна 2, но меньше 3, т. е. при применении функции INT мы получим в качестве результата 2 и т. д. И, наконец, если 0.8≤RND<1, то первое

слагаемое больше или равно 4, но меньше 5, и сумма больше или равна 5, но меньше 6. Для произвольных K, L нужно использовать преобразование

INT((L—K+1)*RND+K).

В качестве примера составим программу получения случайных чисел для игры «Спортлото: 6 из 49» (см. программу 5.10).

Программа 5.10

```
10 RANDOMIZE
20 FOR I=1 TO 6
30 PRINT INT(49*RND+1)
40 NEXT I
50 STOP
```

З а м е ч а н и е. При составлении программы не учитывалась возможность появления совпадающих чисел.

*4. Получение* M *различных случайных целых чисел в пределах от* K *до* L. Получаемые случайные целые числа будем пересылать в массив, например, A. Числа, начиная со второго, сравниваем с полученными ранее числами (элементами массива A). Если такое число уже имеется в массиве, то будем генерировать новое случайное число, пока не получится в итоге требуемое количество различных случайных целых чисел.

П р и м е р. Генерация пяти различных случайных чисел для игры в «Спортлото: 5 из 36» (см. программу 5.11).

*Список используемых переменных.* Исходные данные: нет.

Результат: A — массив размером 5 для размещения полученных случайных чисел.

Вспомогательные переменные: P — случайное целое число из интервала [1, 36]; C — имеет значение 0 или 1; C=1, если в массиве есть число, равное вновь полученному; I, J — управляющие переменные циклов; M — количество уже полученных случайных чисел — элементов массива A.

Программа 5.11

```
10 RANDOMIZE
20 DIM A(5)
30 A(1)=INT(36*RND+1)
40 M=1
50 FOR I=2 TO 5
60 P=INT(36*RND+1) \ C=0
70 FOR J=1 TO M
80 IF P<>A(J) GO TO 100
90 C=1 \ J=I
100 NEXT J
110 IF C=1 GO TO 60
120 A(I)=P \ M=M+1
130 NEXT I
140 FOR I=1 TO 5 \ PRINT A(I); \ NEXT I
150 STOP
```

Пояснения к программе. Если вновь сгенерированное целое число P уже имеется среди элементов массива A (условие в строке 80 не выполняется), полагаем C=1 (строка 90), переменной цикла J присваиваем значение (J=I), превосходящее верхнюю границу внутреннего цикла M (M на единицу меньше текущего значения I), для его завершения.

Пример 1. *Игра Баше.* В игре участвуют двое. Условия игры: имеется N предметов. Соперники ходят по очереди. За каждый ход игрок может взять 1, 2, ..., K предметов. Проигрывает тот, кто вынужден взять последний предмет.

Рассмотрим частный случай игры Баше, когда N=15, K=3. Нетрудно видеть, что в этом частном случае алгоритм выигрыша для первого игрока следующий:

1. взять 2 предмета;

2. второй и последующий ходы делать так, чтобы количество предметов, взятых вместе с соперником за очередной ход, составляло в сумме 4.

Этот алгоритм приводит к выигрышу также при N=7, 11, 15, 19....

Составим программу в соответствии с этим алгоритмом для случая, когда первый игрок — ЭВМ, т. е. ЭВМ делает первый ход (программа 5.12). В программе используются следующие переменные: N — количество предметов (в начале игры N=15), L — число предметов, взятых за 1 ход поочередно каждым игроком, I — код участника (1 — ЭВМ, 2 — ее партнер).

Замечание. Ветвь, которая реализуется при выполнении условия I=1 в строке 190 (вывод сообщения «Вы выиграли»), никогда не будет выполняться, так как программа составлена в соответствии с алгоритмом, гарантирующим выигрыш ЭВМ. Однако наличие этой ветви позволит легко перейти к более общему случаю, когда начальное количество предметов произвольно и, следовательно, не всегда гарантирован выигрыш ЭВМ.

Пояснения к программе. Первый ход выполняет ЭВМ (при I=1 выполняется ветвь, начинающаяся строкой 120). После очередного хода код игрока I изменяет значение (см. строки 100, 160), и если игра не закончена (N≠1), то ход передается партнеру.

Рассмотрим более общий случай игры, когда N произвольно. Проведем предварительный анализ. Выигрыш гарантирован первому игроку, если после его первого хода количество предметов равно 4*M+1, где M — целое (число ходов), чтобы за M ходов, беря в сумме с партнером по 4 предмета, он мог оставить после своего последнего хода 1 предмет. Т. е. своим первым ходом первый игрок должен, если это возможно, привести игру к этой ситуации. Если M определено, то первый ход первого игрока должен быть

**Программа 5.12**

```
10 N=15
20 L=2
30 I=1
40 IF N=1 GO TO 190
50 IF I=1 GO TO 120
60 REM ХОД ВТОРОГО ИГРОКА
70 PRINT "ВАШ ХОД . СКОЛЬКО ПРЕДМЕТОВ БЕРЕТЕ";
80 INPUT L
90 N=N-L
100 I=1
110 GO TO 40
120 REM ХОД ЭВМ
130 L=4-L
140 N=N-L
150 PRINT "МОЙ ХОД";L;"ОСТАЛОСЬ";N;"ПРЕДМЕТОВ"
160 I=2
170 GO TO 40
180 REM ПЕЧАТЬ РЕЗУЛЬТАТА ИГРЫ
190 IF I=1 GO TO 220
200 T$="ВЫ ПРОИГРАЛИ"
210 GO TO 230
220 T$="ВЫ ВЫИГРАЛИ"
230 PRINT T$
240 STOP
```

(N—4*M—1) предметов. Для определения M нужно найти целое от деления N—1 на 4.

Итак, первый ход первого игрока должен быть

$$L=N-4*INT((N-1)/4)-1.$$

Легко видеть, что из четырех последовательных значений N (например, 14, 15, 16, 17) для одного получается L=0, т. е. недопустимое по правилам игры значение. Любой ход первого игрока в этом случае приведет к выигрышу второго игрока (если, конечно, тот не будет ошибаться).

Если окажется L=0, то машина делает первый ход, выбирая число предметов L (1, 2 или 3) случайным образом с использованием функции RND. После каждого хода партнера машина в надежде на его ошибку будет проверять, не появился ли у нее шанс на выигрыш, вычисляя L и сравнивая его с нулем.

Кроме введенных переменных (N, L, I), используем еще вспомогательную переменную A, которой присвоим значение 1, если $L\neq 0$, и значение 0, если L=0. Если окажется A=1, то машина начинает следовать выигрышной стратегии (см. программу 5.12) и выигрывает. Если L=0, то после очередного хода партнера машина снова вычисляет свой ход в надежде на ошибку партнера. И если такая ошибка будет совершена, то машина сразу же использует свой шанс.

Программа 5.13 реализует изложенный алгоритм.

Программа 5.13

```
10 RANDOMIZE
20 PRINT "ВВЕДИТЕ ЧИСЛО ПРЕДМЕТОВ";
30 INPUT N
40 I=1 \ A=0
50 IF N=1 GO TO 270
60 IF I=1 GO TO 140
70 REM ХОД ВТОРОГО ИГРОКА
80 PRINT "ВАШ ХОД . СКОЛЬКО ПРЕДМЕТОВ БЕРЕТЕ";
90 INPUT L
100 N=N-L
110 I=1
120 GO TO 250
130 REM ХОД ЭВМ
140 IF A<>0 GO TO 210
150 L=N-4*INT((N-1)/4)-1
160 IF L<>0 GO TO 190
170 L=INT(3*RND+1)
180 GO TO 200
190 A=1
200 GO TO 220
210 L=4-L
220 N=N-L
230 PRINT "МОЙ ХОД";L;" ОСТАЛОСЬ";N;"ПРЕДМЕТОВ"
240 I=2
250 GO TO 50
260 REM ПЕЧАТЬ РЕЗУЛЬТАТОВ ИГРЫ
270 IF I=1 GO TO 300
280 T$="ВЫ ПРОИГРАЛИ"
290 GO TO 310
300 T$="ВЫ ВЫИГРАЛИ"
310 PRINT T$
320 STOP
```

З а м е ч а н и е. Лишние переходы в строках 120, 180 можно не использовать. Однако это в некоторой степени нарушит правильную структуру программы (см. п. 3 главы 1).

Приведенную программу легко далее обобщить на случай произвольного значения K, т. е. верхнего предела числа предметов, которые могут быть взяты за один ход.

П р и м е р 2. *Игра «Быки и коровы».* ЭВМ генерирует четырехзначное число, в котором все цифры различны. Игрок должен отгадать это число, делая несколько попыток (вводя числа в ЭВМ). После ввода очередного числа ЭВМ проводит анализ и сообщает о степени совпадения введенного числа с исходным, т. е. числа «быков» и «коров». «Корова» — это цифра в числе игрока, совпадающая по разряду с такой же цифрой в отгадываемом числе. «Бык» — цифра в числе игрока, не совпадающая по разряду с такой же цифрой в отгадываемом числе. Если, например, задано число 6482, то число 5428 содержит 1 «корову» и 2 «быка».

Программа этой игры должна содержать следующие части (см. программу 5.14):

— формирование случайного четырехзначного числа, в котором все цифры различны;

— ввод и анализ числа игрока, т. е. определение числа «быков» и «коров».

В программе нужно использовать счетчик числа ходов (введенных чисел) игрока.

Игра заканчивается, когда число полностью отгадано или игрок признает себя побежденным.

При формировании четырехзначного числа A сначала нужно получить 4 различных целых числа из интервала [0,9] и поместить их в массив A1. Далее сформировать число A, считая A1 (1) — числом единиц, A1 (2) — числом десятков и т. д., т. е.

$$A=A1(4)*1000+A1(3)*100+A1(2)*10+A1(1).$$

После ввода числа игрока B его нужно сравнить с A, и если $B\neq A$, то из числа B нужно выделить цифры в массив B1. Для выделения цифры младшего разряда (количества единиц) нужно вычислить

$$P=B-INT(B/10)*10$$

и положить $B1(1)=P$. Далее положить $B=INT(B/10)$ и применить ту же процедуру для выделения следующей цифры (числа десятков) и т. д.

Для определения числа «коров» и «быков» в числе B нужно определить, сколько раз выполняются равенства

а) $A1(I)=B1(I)$, $I=1, 2, 3, 4$;

б) $A1(I)=B1(J)$, $I\neq J$; $I, J=1, 2, 3, 4$.

Так как все цифры числа различны, то после выполнения равенства б) следует сразу завершить цикл по J и перейти к следующему I, увеличив число «быков» на 1.

*Список используемых переменных.* Исходные данные: нет.

Результат: N — число ходов игрока.

Вспомогательные переменные: A1 — массив размера 4 — случайные различные цифры, A — генерируемое четырехзначное число, B — число игрока, B1 — массив размера 4 — цифры числа игрока, K — число «коров», L — число «быков», I, J — индексы — управляющие переменные циклов, B2 — копия числа игрока, используемая при выделении его цифр.

Задание I уровня. Задачи I уровня предназначены для приобретения навыков работы с датчиком случайных чисел (функцией RND) и требуют составления программ, используемых для получения случайных чисел при программировании игр (см. задачи II и III уровней). Для каждой задачи требуется разработать

Программа 5.14

```
10 DIM A1(4),B1(4)
20 RANDOMIZE
30 REM ЭВМ ЗАГАДЫВАЕТ ЧИСЛО
40 A1(1)=INT(10*RND)
50 FOR I=2 TO 4
60 A1(I)=INT(10*RND)
70 FOR J=1 TO I-1
80 IF A1(I)=A1(J) GO TO 60
90 NEXT J
100 NEXT I
110 A=A1(4)*1000+A1(3)*100+A1(2)*10+A1(1)
120 N=0
130 PRINT "ВАШ ХОД. ЕСЛИ НОЛЬ - ВЫ СДАЛИСЬ"
140 INPUT B
150 IF B=0 GO TO 510
160 REM 510 - ПЕРЕХОД К СТРОКЕ С НОМЕРОМ 510, ВЫ СДАЛИСЬ
170 N=N+1
180 K=0
190 REM K - КОЛИЧЕСТВО КОРОВ
200 L=0
210 REM L - КОЛИЧЕСТВО БЫКОВ
220 IF A=B GO TO 480
230 REM 480 - ПЕРЕХОД К СТРОКЕ С НОМЕРОМ 480, ВЫ УГАДАЛИ
240 IF B<10000 GO TO 280
250 PRINT "ВВЕДИТЕ ЧЕТЫРЕХЗНАЧНОЕ ЧИСЛО"
260 N=N-1
270 GO TO 130
280 I=1 \ B2=B
290 C=INT(B2/10) \ B1(I)=B2-C*10
300 B2=C \ I=I+1
310 IF I<=4 GO TO 290
320 REM СЧИТАЕМ КОЛИЧЕСТВО КОРОВ И БЫКОВ
330 FOR I=1 TO 4
340 FOR J=1 TO 4
350 IF I=J GO TO 390
360 IF B1(I)<>A1(J) GO TO 420
370 L=L+1
380 GO TO 430
390 IF B1(I)<>A1(J) GO TO 420
400 K=K+1
410 GO TO 430
420 NEXT J
430 NEXT I
440 REM ПЕЧАТАЕМ РЕЗУЛЬТАТЫ ХОДА
450 PRINT N;"- ИЙ ХОД"
460 PRINT "ВАШЕ ЧИСЛО - ";B;" В НЕМ ";K;" КОРОВЫ, ";L;" БЫКА"
470 GO TO 130
480 PRINT "В ВАШЕМ ЧИСЛЕ 4 КОРОВЫ - НА ";N;"-ОМ ХОДУ ВЫ УГАДАЛИ"
490 PRINT "ВАШЕ ЧИСЛО - ";B;" = ";A
500 STOP
510 PRINT "НА ";N;" -ОМ ХОДУ ВЫ СДАЛИСЬ"
520 PRINT "ЗАГАДАННОЕ ЧИСЛО - ";A
530 PRINT "ИГРА ОКОНЧЕНА"
540 STOP
```

алгоритм и применить один из способов получения случайных чисел, приведенных во введении к работе. Программу проверить на ЭВМ.

*Варианты задач I уровня.*

1. Получить четырехзначное число, все цифры которого различны. Нуль может быть старшей цифрой числа.

2. Получить 6 целых случайных чисел в интервале от 1 до 6.

3. Задано целое число A (A ∈ [1,36]). Получить случайное целое число B из интервала [1,36] и напечатать сообщение, равно ли число B числу A.

4. Провести тираж «Спортлото: 6 из 49».

5. Задана символьная матрица A□(10×10), в каждый элемент которой занесен символ «.» (точка). Случайным образом выбрать один элемент матрицы A и занести в него символ * (звездочка).

6. Определить, попало ли случайное целое число C из интервала [0,36] в лежащий внутри него интервал [A, B]: A и B — целые числа, A>0. Если C=0, то напечатать сообщение: «ВЫПАЛ НУЛЬ».

7. В символьной матрице A□(10×10), каждому элементу которой присвоен символ «.» (точка), выбрать случайным образом 5 различных элементов и присвоить им символ * (звездочка).

8. В матрице A (10×10) значения всех элементов равны нулю. Выбрать случайным образом 7 различных элементов и присвоить им значение 3.

9. Подсчитать, сколько раз встретилось целое число L среди K случайных целых чисел из интервала [1,6] (L ∈ [1,6]).

10. Получить M различных случайных чисел из интервала [K, L]; M<L—K.

11. Определить, сколько случайных целых чисел из интервала [1,6] нужно получить, чтобы заданное число L (L ∈ [1,6]) встретилось среди них ровно 5 раз.

12. Напечатать последовательность 10 различных случайных целых чисел из интервала [1,20].

13. Подсчитать, сколько раз встретилось целое число P (P ∈ [K, L]) среди M случайных целых чисел в интервале [K, L]; M<L—K.

14. Получить четырехзначное целое число. Цифры могут совпадать. Нуль не может быть старшей цифрой.

15. Получить пять различных целых чисел из интервала [1,20]. В массив B размером 20 поместить число 7 в элемент, индекс которого является первым из полученных случайных чисел, затем поместить число 5 в элемент, индекс которого является вторым случайным числом, 3 — третьим, 2 — четвертым и 1 — пятым (остальные элементы — нулевые).

Указания к решению задач I уровня.

1. Сначала получить 4 различных целых числа из интервала [0,9], помещая их в массив, например, D. Далее получить число A, считая, что D (1) — количество единиц в числе A, D (2) — количест-

во десятков, D(3) — количество сотен, D(4) — количество тысяч. Т. е.

A=D(4)*1000+D(3)*100+D(2)*10+D(1).

4. Задача сводится к получению 6 различных целых чисел из интервала [1,49].

5. Следует случайным образом получить 2 целых числа I и J из интервала [1,10], а затем присвоить элементу символьной матрицы A○(I, J) значение *.

7. Получить два целых случайных числа I, J из интервала [1,10]. Элементу матрицы A○(I, J) присвоить значение *. Для каждой новой пары чисел I, J проверять значение элемента A○(I, J). Если A○(I, J)="*", то получить новую пару чисел, пока не будет выбрано в итоге 5 различных элементов матрицы A○.

8. См. указания к задаче 7.

Задание II уровня. Задачи II уровня требуют составления программ простых игр или фрагментов игр, используемых в задачах III уровня. При их решении для получения случайных чисел следует использовать программы, составленные для решения задач I уровня.

Для каждой задачи требуется разработать алгоритм, используя, если необходимо, метод пошаговой детализации, составить программу. В программе предусмотреть вывод соответствующих сообщений для организации диалога с ЭВМ. Проверить работу программы на ЭВМ.

*Варианты задач II уровня.*

1. Написать программу набора суммы очков S из слагаемых, являющихся целыми числами, каждое из которых принадлежит интервалу [K, L] и генерируется случайным образом. Сумму набирать до тех пор, пока S<N—5 (N задано). Если значение S окажется больше N, напечатать соответствующее сообщение (фрагмент игры «Набери сумму», набор очков ЭВМ).

2. В задаче 1 набрать сумму S, как можно более близкую к N. Перед получением очередного слагаемого на экране дисплея должен высвечиваться вопрос о том, хочет игрок прибавить очередное слагаемое или закончить набор суммы (фрагмент игры «Набери сумму», набор очков человеком — партнером ЭВМ).

3. Получить случайным образом четырехзначное число A, цифры которого могут совпадать, нуль не может быть старшей цифрой числа (число A на экран не выводится). Ввести в память ЭВМ другое четырехзначное число B. Определить:

а) сколько цифр числа B совпали по разряду с цифрами числа A;

б) сколько цифр числа B имеется в числе A (цифры не совпадают по разряду).

(Фрагмент игры «Быки и коровы» — определение результатов одного хода игры.)

4. Генерируется случайное целое число A из интервала [0,36]. Если A=0, напечатать соответствующее сообщение. Если A≠0, то определить и напечатать:

— четное число A или нечетное;

— в какой из интервалов попало A: [1,18], [19,36];

— в какую дюжину попало A: [1,12], [13,24], [25,36];

— в какую из девяток попало A: [1,9], [10,18], [19,27], [28,36];

— в какую из шестерок попало A: [1,6], [7,12], ..., [31,36];

— в какую из троек попало A: [1,3], [4,6], ..., [34,36];

— в какую из двоек попало A: [1,2], [3,4], ..., [35,36];

— равно ли A заданному целому числу B (B∈[1,36]).

(Фрагмент игры «Карусель-лото» — определение результатов одного хода.)

5. Генерируется случайным образом целое число из интервала [1,10] 50 раз. Подсчитать, сколько раз выпало каждое целое число, построить гистограмму частот выпадания чисел из интервала [1,10].

6. Построить гистограмму из 10 строк, длина каждой строки задается случайным образом в пределах от K до L символов.

7. Написать программу определения K чисел из интервала [1,36], имеющих наибольшую частоту выпадения в 30 тиражах «Спортлото: 5 из 36» (K≤15). Построить гистограмму частот появления этих чисел.

8. Определить, какое число (числа) чаще всего выпадало в 50 тиражах «Спортлото: 6 из 49», напечатать частоту появления этого числа и само число.

9. В символьной матрице A□(10×10) все элементы имеют значение «.» (точка). Случайным образом выбрать один элемент матрицы, присвоить ему значение X, а окружающим его элементам присвоить значение _ (пробел) (фрагмент игры «Морской бой» — уничтожение катера при очередном ходе).

10. В матрице A (10×10) случайным образом выбрать 4 элемента так, чтобы они не соприкасались друг с другом (фрагмент игры «Морской бой» — расстановка на поле 10×10 четырех катеров).

11. Определить, сколько раз встретилось целое число M (M∈∈[1,6]) среди шести случайных целых чисел из интервала [1,6]. Если M встретилось менее 5 раз, то получить еще (6—K) случайных чисел, где K — количество появлений числа M среди первых 6 чисел. Продолжать получение случайных чисел, пока число не встретится 5 раз.

12. В числовой матрице A (10×10) выбрать случайным образом 4 элемента матрицы A, расположенных подряд по вертикали или горизонтали (фрагменты игры «Морской бой» — установление на поле 10×10 четырехклеточного корабля, расположенного на одной линии).

13. В числовой матрице A (10×10) выбрать случайным образом 4 соседних элемента матрицы A, расположенных произвольно по отношению друг к другу, и заполнить их числом 5. Остальные элементы нулевые. (Фрагмент игры «Морской бой» — установление на поле 10×10 четырехклеточного корабля, расположенного произвольно.)

14. Написать программу, определяющую, сколько раз сумма из пяти слагаемых, генерируемых случайным образом, превысила заданное число N. Слагаемые выбираются из целых чисел из интервалов [K, L]. Сумму набирать 20 раз.

15. Написать программу игры в кости по следующим правилам. Играющий называет (вводит в ЭВМ) целое число A от 1 до 6. Затем генерируется три случайных целых числа из интервала [1,6] («бросаются кости»). Если среди трех сгенерированных чисел число A встретилось один раз — игрок получает 5 очков, два раза — 10 очков, три раза — 15 очков, не встретилось ни разу — 0 очков.

Количество играющих до 5, кости для каждого игрока бросаются по очереди.

Указания к решению задач II уровня.

1. После получения каждого нового слагаемого печатать слагаемое и сумму.

2. Перед определением нового слагаемого следует печатать значение набранной суммы и задавать вопрос: «Будете продолжать набор?». Для окончания набора можно вводить нуль.

3. См. теоретическое введение к работе (пример 2). Так как цифры числа могут совпадать, то цикл по J при сравнении различных цифр чисел A и B нужно выполнять до конца.

4. Для определения номера дюжины, в которую попало случайное число A, номера девятки, номера шестерки, номера тройки, номера двойки, вычислить C по формуле:

$$C = INT((A-1)/K),$$

где C — номер комбинации (дюжины, девятки и т. д.), K — количество цифр, входящих в интервал.

Например, для K=12 (дюжина) значением C (номер дюжины) может быть: 0 — число A в интервале [1,12], 1 — число A в интервале [13,24] или 2 — число A в интервале [25,36]. По значению C можно вычислить границы интервала [D, E]:

$$D = C*K+1, \quad E = (C+1)*K$$

и напечатать их. Так, при K=12 и C=0: D=1, E=12.

Аналогично для случаев K=9, K=6, K=3, K=2.

5. В массиве B размером 10 установить первоначальные значения элементов равными нулю. При получении очередного случайного целого числа I ($I \in [1,10]$) значение B (I) увеличить на единицу.

В итоге значение каждого элемента B(I) будет соответствовать частоте появления числа I. Значения элементов массива B представить в виде гистограммы.

7. См. указания к задаче 5 уровня II.

8. См. указания к задаче 5 уровня II. После проведения всех тиражей определить по значениям элементов массива B число, имеющее наибольшую частоту.

9. См. указания к задаче 5 уровня I. Преобразования рекомендуется производить сначала на числовой матрице A, заполненной нулями. Выбранному случайным образом элементу матрицы A присвоить значение 5, соседним клеткам — другое значение, например, 3. При печати матрицы произвести замену: нулю поставить в соответствие символ «.» (точка), 5 — символ X, 3 — символ —.

10. См. указания к задаче 7 уровня I, к задаче 9 уровня II. В числовой матрице присваивать выбранным элементам значение 5, соседним 3 (первоначально все элементы равны нулю). Если значение вновь выбранной клетки не равно нулю, то либо эта клетка уже встречалась, либо она является соседней по отношению к ранее выбранной. Тогда следует выбрать новую клетку.

11. Среди 6 случайных чисел определить, сколько раз встретилось число M. Это значение присвоить переменной K. Если K<5, то выработать (6—K) случайных чисел и определить, сколько раз среди них встречается число M, при этом значение K увеличить на количество вновь выпавших чисел M и процедуру повторить. В переменной C считать, сколько раз осуществлялась генерация групп по шесть случайных чисел.

12. Выбрать один элемент матрицы случайным образом, присвоить ему значение 5 (см. указания к задаче 5 уровня I и к задаче 9). Затем определить случайное целое число I в интервале [0;1]. Если I=0, то корабль строить по горизонтали, иначе — по вертикали. Затем определить новое случайное целое число I из интервала [0;1]. Если I=0, то строим корабль влево (для вертикали — вниз), иначе — вправо (для вертикали — вверх). Продолжаем построение продвижением в выбранном направлении до полного построения корабля. Если в данном направлении строить корабль нельзя (граница массива), строим его в противоположном направлении.

13. См. указания к задаче 12 уровня II, но вместо числа I из интервала [0,1] генерируем случайное целое число J из интервала [—1,1]. Полученное значение J складываем с первой координатой первого выбранного элемента матрицы A и генерируем новое число J. Новое значение J складываем со второй координатой первого выбранного элемента матрицы A. Если получились координаты старого элемента или новые координаты вышли за пределы массива, то повторяем процедуру еще раз. После нахождения нового элемен-

та матрицы A повторяем процедуру, считая исходными координаты нового элемента, пока не построим четырехклеточный корабль.

15. Очки, набранные игроками, поместить в массив. Для определения победителя найти максимальный элемент этого массива и его индекс.

Задание III уровня. Задачи III уровня требуют составления законченной программы игры, используя фрагменты программ, подготовленные при решении задач I и II уровней.

Требуется составить схему игры, продумать организацию диалога между человеком и ЭВМ. Выбрать способ представления данных. Разработать алгоритм, используя, если требуется, метод пошаговой детализации. Составить программу. В программе предусмотреть наглядный вывод результатов и соответствующих сообщений по ходу игры и после ее окончания. Проверить работу программы на ЭВМ. Для каждой задачи даются указания по ее решению.

*Варианты задач III уровня.*

1. Игра «Скачки» (вариант 1). В игре участвуют 10 наездников; на каждый тур игры каждый из них продвигается вперед на расстояние от 1 до 5 км случайным образом. Длина дистанции — 50 км. Всего проводится 5 заездов, победителю каждого заезда начисляется 5 очков. Просмотр наездников в туре осуществлять последовательно. Победителем считается наездник, набравший наибольшее количество очков во всех заездах. Перед началом заездов участник игры выбирает номер наездника, с которым он будет идентифицироваться во время игры. Количество участников игры не превышает 10, в программе предусмотреть возможность случайного распределения номеров наездников.

2. Игра «Скачки» (вариант 2). В каждом туре с вероятностью 0,1 каждый наездник может упасть, т. е. продвинуться за этот тур на ноль *км*.

3. Игра «Трек». В велогонке на треке участвуют 20 спортсменов, велогонка длится 20 кругов. На финише каждого круга занявшему 1-е место начисляется 7 очков, 2-е место — 5 очков, 3-е место — 3 очка, 4-е место — 2 очка, 5-е место — 1 очко. Победителем (победителями) считается велосипедист, набравший наибольшее количество очков или победивший в 5 кругах. Перед велогонкой участники игры случайным образом разыгрывают номера, под которыми они выступают.

4. Игра «Гонка с выбыванием». В мотокроссе участвуют 15 спортсменов, участники должны преодолеть 14 кругов. После каждого круга участник, занимающий последнее место, снимается с соревнований. Перед началом соревнований участники игры случайным образом разыгрывают номера, под которыми они выступают. Количество участников не более 15.

5. Игра «Набери сумму» (вариант 1). Два участника по очереди

набирают сумму из слагаемых, которые могут иметь целые значения от K до L и задаются случайным образом. Задача заключается в том, чтобы вовремя прекратить набор суммы, когда она достаточно близка к заданному по условиям игры числу N (или равна ему), но не превосходит его. Выигравшим считается участник, набравший сумму, более близкую к N. Если сумма какого-либо игрока превзойдет N, то он сразу проигрывает, и игра прекращается. В случае равенства сумм у обоих игроков победителем считается набравший сумму вторым. Одним из партнеров является ЭВМ, другим — человек. ЭВМ набирает сумму S, пока $S<N-5$. Второй партнер может принять решение о прекращении набора перед получением очередного слагаемого.

6. Игра «Набери сумму» (вариант 2). Первым сумму набирает человек. ЭВМ при наборе суммы использует следующую стратегию. Она стремится набрать сумму, равную или большую, чем сумма первого игрока. Если суммы равны, выигрыш присуждается ЭВМ.

7. Игра «Набери сумму» (вариант 3). Играют два партнера. ЭВМ лишь генерирует случайным образом слагаемые суммы и определяет результат игры. Предусмотреть возможность повторного проведения игры и определения общего счета после ее окончания.

8. Игра «Карусель-лото» (вариант 1). За один ход ЭВМ генерирует случайное целое число в интервале [0,36]. Перед этим участники заказывают одну комбинацию из следующих возможных (стараясь угадать число или интервал, в который число попадает):

а) выпадет четное или нечетное число;

б) число попадет в интервал [1,18] или [19,36];

в) число попадет в одну из трех дюжин [1,12], [13,24], [25,36];

г) число попадет в одну из четырех девяток: [1,9], [10,18], [19,27], [28,36];

д) число попадает в одну из шестерок: [1,6], [7,12], [13,18], [19,24], [25,30], [31,36];

е) число попадет в одну из троек: [1,3], [4,6], ..., [34,36];

ж) число попадет в одну из двоек: [1,2], [3,4], ..., [35,36];

з) выпадет K, где $K \in [1,36]$.

За угадывание комбинации а) или б) игрок получает 1 очко; в) — 2 очка; г) — 3 очка; д) — 5 очков; е) — 11 очков; ж) — 17 очков; з) — 35 очков. Генерирование числа осуществляется заданное количество раз и определяется условиями игры. В случае выпадения нуля ни один из участников не получает ни одного очка. Побеждает набравший большее количество очков.

9. Игра «Карусель-лото» (вариант 2). Перед игрой каждый из участников имеет 8 очков. Перед началом партии участник как бы «ставит» очко на выбранную комбинацию. В случае успеха он получает назад свое очко плюс приз в очках (см. условие задачи 8), в случае неудачи — теряет очко. При выпадении нуля теряются все

поставленные в партии очки. Участник может пропускать партию, ставить несколько очков на одну комбинацию (в этом случае они считаются независимыми). Если у участника кончились очки, он выбывает из игры.

10. Игра «Морской бой» (вариант 1). Игра происходит между двумя участниками, один из которых ЭВМ, на поле размером 10×10 клеток. Каждый из участников расставляет на поле 10 одноклеточных катеров (катера не могут касаться друг друга). При попадании в катер при очередном ходе он считается уничтоженным. Побеждает уничтоживший первым катера соперника. Если участник во время очередного хода уничтожил катер противника, то он ходит еще раз. Ход в клетку, соседнюю с уничтоженным катером, считается недействительным и повторяется. ЭВМ расставляет катера случайным образом. Ее партнер вводит координаты катеров с клавиатуры.

11. Игра «Морской бой» (вариант 2). На поле каждый участник расставляет эскадру, состоящую из одного 4-хклеточного линкора, двух трехклеточных крейсеров, трех двухклеточных эсминцев и четырех одноклеточных катеров. Все многоклеточные корабли расположены линейно, т. е. только горизонтально или вертикально.

12. Игра «Морской бой» (вариант 3). Корабли могут располагаться произвольно, но не должны касаться друг друга (находиться в соседних клетках).

13. Игра в «кости» (вариант 1). Кость — кубик с размеченными гранями, на грани нанесены 1, 2, ..., 6 точек (очков). Каждый игрок сначала бросает 6 костей сразу. После первого хода игрок выбирает какую-либо из выпавших граней и откладывает кости, которые выпали на эту грань (эти кубики выбывают из игры). Далее игрок бросает оставшиеся кости и снова откладывает кости, выпавшие на выбранную после первого броска грань, и т. д., пока не будет отложено не менее пяти костей. Далее кости бросает второй игрок. Побеждает игрок, отложивший не менее пяти костей за меньшее число бросков. При одинаковом числе бросков победителем считается игрок, выбравший грань с большим числом очков.

14. Игра в «кости» (вариант 2). Закончив выбрасывание костей на первую выбранную грань M1 (отложив не менее пяти костей), игрок опять бросает шесть костей, выбирает вторую грань M2≠M1 и снова повторяет броски, пока не будет отложено не менее пяти костей, выпавших на грань M2, и т. д., пока не будут выбраны все шесть граней. Далее кости бросает второй игрок.

15. Игра «Быки и коровы». Условия игры приведены в теоретическом введении (пример 2). Цифры в числе могут совпадать. Нуль не может быть старшей цифрой числа.

Указания к решению задач III уровня.

1. Перед очередным заездом массив A обнуляется. Для каждого наездника (от 1 до 10) в каждом туре генерируется случайное целое

число из интервала [1; 5] и прибавляется к соответствующему элементу массива А. Как только один из участников наберет 50 *км*, заезд останавливается и победителю начисляется 5 очков. Очки для каждого наездника суммируются в элементах массива В (размером 10). После пяти заездов определяется победитель (номер максимального элемента массива В).

2. В каждом туре для каждого наездника генерировать случайное целое число из интервала [1,10]. Если выпадает заранее определенное число, например, 1 (вероятность его появления равна 0,1), то соответствующие элементы массивов А и В не изменяются в текущем туре. См. также указания к задаче 1 уровня III.

3. После каждого круга определять 5 разных случайных чисел от 1 до 20. Первое число определяет номер участника, занявшего первое место, второе — занявшего второе место и т. д. Значения соответствующих элементов массива А увеличивать на указанное в правилах игры количество очков. В массиве В отмечать победы I-го участника, увеличивая значения В(I) на 1. Перед каждым новым кругом проверять, выполняется ли условие В(I)=5 для какого-либо I (I=1, ..., 20). Если выполняется, то победил спортсмен с номером I. После каждого круга печатать номер круга и таблицу результатов (номера элементов массива В — номера участников, содержимое элементов массива В — набранные участниками очки). После прохождения всех кругов найти победителя (определить номер максимального элемента массива В).

4. После очередного круга I-му элементу массива В, где I — номер последнего на этом круге спортсмена, присваивается значение нуль и далее эти элементы не рассматриваются (число элементов массива В уменьшается на 1).

5. См. указания к задачам 1, 2, а также задачу 14 уровня II.

6. См. указания к задаче 5. При наборе суммы ЭВМ набор прекращается, если эта сумма равна или больше суммы, набранной первым игроком.

7. См. указания к задаче 2 уровня II. Предусмотреть возможность повторения игры, добавив внешний цикл. Для окончания игры можно использовать специальное значение, например, нуль.

8. Программа должна содержать следующие части:

— ввод ставок участников игры. Сведения о комбинациях, на которые ставят участники, удобно представить в виде матрицы А, где номер строки — номер участника, а номера столбцов — номера комбинаций. Наличие нуля в столбце ($a_{ij}=0$) означает, что на эту комбинацию игрок не поставил. Если

$$a_{i1}=\begin{cases}1-\text{выбран нечет},\\ 2-\text{выбран чет};\end{cases}$$

$$a_{i2}=\begin{cases}1-\text{первая половина},\\ 2-\text{вторая половина}.\end{cases}$$

В $a_{i3}$, $a_{i4}$, ..., $a_{i7}$ указывается номер набора чисел, на который ставит игрок; в $a_{i8}$ указывается конкретное выбранное участником число;

— генерация случайного числа из интервала [0,36];

— определение, в какие интервалы попало число. Информацию об этом целесообразно разместить в массиве F, аналогично одной строке матрицы A. Для определения номеров интервалов, в которые попало число, вычислять C по формуле

C=INT((A—1)/K),

где C — номер комбинации (дюжины, девятки и т. д.), K — количество чисел, входящих в интервал. Например, для K=12 (номер дюжины) значением C может быть: 0 — число A попало в интервал [1,12], 1 — [13,24] или 2 — [25,36];

— цикл по строкам матрицы A (игрокам). Каждая строка сравнивается с массивом F. Сумма очков каждого игрока запоминается в соответствующем элементе массива D;

— определение победителя (определение индекса максимального элемента массива D).

9. См. указания к задаче 8. Перед игрой в каждый элемент массива результатов участников поместить 8. По результатам партии из общей суммы очков каждого игрока вычесть по 1 за каждую неугаданную комбинацию, на которую поставил игрок. За каждую угаданную комбинацию прибавить к сумме число очков, заданное условиями игры (см. задачу 8). Участники с итоговой суммой, равной нулю, выбывают из игры.

10. Перед расстановкой катеров две числовые матрицы размером 10×10 заполнить нулями. ЭВМ расставляет катера случайным образом. Для этого генерируется по 2 целых числа из интервала [1, 10] и элементу матрицы A(I, J) присваивается какое-либо значение, например, 5. Все соседние клетки заполнить, например, единицами, и при получении каждой следующей пары чисел I, J проверять условие A(I, J)=0. Если оно не выполняется, то получить новую пару чисел, пока не будет расставлено 10 катеров, не касающихся друг друга. См. также указание к задаче 10 уровня II. Координаты катеров второго партнера вводить с клавиатуры. Матрицу заполнить аналогично описанному выше способу. Во время игры все преобразования осуществлять с числовыми матрицами, используя, например, следующие значения: 6 — катер поражен, 1 — сделан ход. При поражении корабля следует значения всех соседних клеток заменить на 1. Если ход сделан в клетку со значением 1, то его нужно повторить. После каждого хода числовые матрицы преобразовать в символьные и вывести на экран. Преобразование можно осуществлять по следующим правилам: 1 заменить на —, 6 — на X, вместо остальных значений вывести «.».

11. См. указание к задаче 10. В примерный перечень значений следует добавить: 4 — корабль ранен, а после полного поражения корабля (это определяется программным путем) во все соседние клетки занести 1. После каждого попадания в корабль определить по значениям соседних клеток, убит или ранен корабль, и напечатать соответствующее сообщение. См. также указание к задаче 12 уровня II.

12. См. указание к задаче 10 и к задаче 13 уровня II.

13. См. указание к задаче 11 уровня II.

14. См. указание к задаче 11 уровня II.

Вопросы для самопроверки.

1. На чем основана возможность составления игровых программ на бейсике?
2. Получение случайных чисел из интервала (0; 1). Функция RND.
3. Как получить различные последовательности случайных чисел? Оператор RANDOMIZE и его место в программе.
4. Получение случайных чисел, принадлежащих интервалу (A, B).
5. Получение случайных целых чисел в пределах от K до L.
6. Получение различных случайных целых чисел в пределах от K до L.
7. Получение случайного целого числа, содержащего заданное количество различных цифр.
8. Может ли ЭВМ выступать в роли одного из партнеров в игре и как это реализуется в программе?
9. Какие функции может выполнять в игре ЭВМ, если она не является ни одним из партнеров?
10. Выбор случайным образом заданного количества различных элементов матрицы: а) соседних друг по отношению к другу, б) не соприкасающихся.
11. Какие этапы необходимо выполнить при решении задачи на ЭВМ? Пояснить на примерах составления игровых программ. Показать связь между этапами выбора способа представления данных и разработкой алгоритма. Какие требования к программе особенно важны при программировании игр?
12. Что является исходными данными и результатами для игровых программ?

## *Работа 13.* РАЗРАБОТКА И ВЫПОЛНЕНИЕ ПРОГРАММ СЛОЖНОЙ СТРУКТУРЫ С ИСПОЛЬЗОВАНИЕМ ВНЕШНИХ УСТРОЙСТВ

Теоретическое введение. В данной работе рассматриваются способы организации подпрограмм как отдельных программных модулей, а также методы разработки и организации программ сложной структуры. Необходимые для этого средства языка бейсик (операторы COMMON, OVERLAY, CHAIN) излагаются в п. 14 главы 3. Рассмотрение проводится на конкретных примерах.

Пример 1. Разработка подпрограммы формирования файла последовательного доступа определенной структуры.

Требуется составить подпрограмму, формирующую файл последовательного доступа (см. п. 13 главы 3), имеющий следующую структуру: фамилия, дата рождения (дата рождения записывается как строка символов вида ДД.ММ.ГГГГ, где ДД — число, ММ — месяц, ГГГГ — год рождения).

Имя формируемого файла подпрограмма будет получать как обобщенную переменную (программа 5.15).

**Программа 5.15**

```
10 COMMON N$
20 OPEN N$ FOR OUTPUT AS FILE #1
30 PRINT "ВВЕДИТЕ ФАМИЛИЮ, ЕСЛИ ВВЕДЕТЕ 1 - КОНЕЦ ВВОДА"
40 INPUT F$
50 IF F$="1" GO TO 130
60 PRINT #1,F$
70 PRINT "ВВЕДИТЕ ДАТУ РОЖДЕНИЯ В ВИДЕ"
80 PRINT "ДД.ММ.ГГГГ , ГДЕ ДД - ДЕНЬ, ММ - МЕСЯЦ,"
90 PRINT "ГГГГ - ГОД РОЖДЕНИЯ"
100 INPUT R$
110 PRINT #1,R$
120 GO TO 30
130 CLOSE #1
140 STOP
```

Пояснение к программе. В строке 10 оператор COMMON объявляет обобщенную переменную N$, в которой будет передано в подпрограмму имя формируемого файла.

В строке 20 открывается файл с именем, переданным через обобщенную переменную N$.

В строках 30—110 в память ЭВМ поочередно вводятся данные. Информация сразу же записывается в файл. Условие окончания формирования файла — ввод вместо фамилии символа 1 (строка 50).

По окончании формирования файл закрывается (строка 130).

Пример 2. Разработка подпрограммы с организацией перекрытия (программа 5.16).

Требуется составить подпрограмму вычисления суммы типа

$$S=\sum_{i=\mathrm{K}}^{\mathrm{M}} a_i,$$

где $a_i$ — очередной член суммы, заданный формулой.

Для выполнения подпрограммы значения K, M будем передавать через обобщенные переменные, а формулу вычисления очередного члена суммы будем передавать через программный файл с именем, хранящимся в обобщенной переменной L$. Программный файл

должен состоять из одной строки вида 50 A=F, где F — арифметическое выражение, по которому вычисляется очередной член суммы $a_i$ в зависимости от номера $i$.

**Программа 5.16**

```
10 COMMON K,M,L$
20 OVERLAY L$
30 S=0
40 FOR I=K TO M
50 A=I
60 S=S+A
70 NEXT I
80 PRINT "ЗНАЧЕНИЕ СУММЫ S=";S
90 STOP
```

Пояснения к программе. В строке 10 оператор COMMON объявляет обобщенные переменные K, M, L○, в которых передаются в подпрограмму значения пределов изменения индексов при суммировании, и имя программного файла, в котором в виде строки программы записана формула вычисления значения очередного члена суммы в зависимости от его номера I.

В строке 20 оператор OVERLAY организует перекрытие программы, находящейся в оперативной памяти, и программного файла с именем, хранящимся в обобщенной переменной L○, с целью включения в программу 5.16 строки из указанного файла, в которой вычисляется очередной член суммы, на место строки 50.

В строках 30—80 вычисляется значение суммы и выводится на экран дисплея.

Пример 3. Составить программу сложной структуры, состоящую из нескольких модулей (программы 5.17, 5.18, 5.19 и 5.20). Программа состоит из четырех программных модулей. В первом модуле (имя "MODUL1") в память ЭВМ в обобщенный массив A из 500 элементов заносятся значения. Во втором модуле (имя "MODUL2") подсчитывается и выводится на экран дисплея значение суммы элементов обобщенного массива A. В третьем модуле (имя "MODUL3") на экран дисплея будут выведены наибольшее и наименьшее значения элементов обобщенного массива A. В четвертом модуле (имя "MODUL4") элементы обобщенного массива A записываются в файл последовательного доступа на ВЗУ.

**Программа 5.17. MODUL1**

```
10 COMMON A(500)
20 FOR I=1 TO 500
30 PRINT "ВВЕДИТЕ";I;"ДАННОЕ"
40 INPUT A(I)
50 NEXT I
60 CHAIN "MODUL2"
```

**Программа 5.18. MODUL2**

```
10 COMMON A(500)
20 S=0
30 FOR I=1 TO 500
40 S=S+A(I)
50 NEXT I
60 PRINT "СУММА ЭЛЕМЕНТОВ S=";S
70 CHAIN "MODUL3"
```

**Программа 5.19. MODUL3**

```
10 COMMON A(500)
20 B=A(1) \ M=A(1)
30 K=1 \ L=1
40 FOR I=1 TO 500
50 IF B<A(I) THEN B=A(I) \ K=I \ GO TO 70
60 IF M>A(I) THEN M=A(I) \ L=I
70 NEXT I
80 PRINT "НАИБОЛЬШИЙ ЭЛЕМЕНТ МАССИВА A =";B;" ЕГО НОМЕР =";K
90 PRINT "НАИМЕНЬШИЙ ЭЛЕМЕНТ МАССИВА A =";M;" ЕГО НОМЕР =";L
100 CHAIN "MODUL4"
```

**Программа 5.20. MODUL4**

```
10 COMMON A(500)
20 PRINT "СООБЩИ ИМЯ ФАЙЛА"
30 INPUT N$
40 OPEN N$ FOR OUTPUT AS FILE #1
50 FOR I=1 TO 500
60 PRINT #1,A(I)
70 NEXT I
80 CLOSE #1
90 STOP
```

Пояснения к программам. MODUL1. В этом модуле в строке 10 оператором COMMON объявляется обобщенный массив данных, который в данном модуле вводится в память ЭВМ (строки 20—50). В строке 60 оператор CHAIN организует вызов в оперативную память программного файла с именем "MODUL2", и управление передается в вызванный файл строке с наименьшим номером.

MODUL2. В этом программном модуле находится сумма элементов массива A, который описывается оператором COMMON (строка 10). Сумма элементов вычисляется (строки 20—50) и выводится на экран дисплея (строка 60). Последним оператором модуля является оператор CHAIN (строка 70), вызывающий в оперативную память программный файл с именем "MODUL3". Управление передается в первую строку файла "MODUL3".

MODUL3. В строке 10 оператор COMMON описывает обобщенный массив A. В строках 20—70 среди элементов массива находятся максимальный и минимальный, а также их местоположение в массиве. Найденные значения выводятся на экран дисплея. В строке 100 записан оператор CHAIN, вызывающий в оперативную память

программный файл с именем "MODUL4". Управление передается в первую строку файла "MODUL4".

MODUL4. В строке 10 оператором COMMON описывается обобщенный массив А. В память ЭВМ после запроса вводится имя файла (строки 20—30), файл открывается (строка 40) и в него вводятся значения элементов массива А (строки 50—70). По окончании записи элементов файл закрывается (строка 80), и выполнение программы заканчивается.

Задание I уровня. Предлагаемые задачи оформить как подпрограммы с передачей данных через обобщенные переменные и (или) через файл последовательного доступа на ВЗУ. Подготовить тесты. Проверить работу подпрограммы. Для проверки работы подпрограммы составить программу, в которой задаются входные параметры подпрограммы, осуществляется обращение к подпрограмме и печатаются результаты ее работы. При выполнении задания использовать алгоритмы, приведенные в теоретическом введении к работам 4, 7, 8, 11.

*Варианты задач I уровня.*

В задачах 1—9 предусмотреть в подпрограммах возможность обработки одномерных массивов размером до 100, матриц — до 10×10. Исходные массивы и их размеры получать через обобщенные переменные.

1. Суммирование элементов матрицы.
2. Представление матрицы в виде одномерного массива по строкам.
3. Поиск максимального и минимального элементов в массиве.
4. Сортировка элементов массива по возрастанию.
5. Инвертирование массива.
6. Циклический сдвиг элементов массива на К позиций влево или вправо. Значение К передавать через обобщенную переменную, знак К определяет направление сдвига (+ вправо, — влево).
7. Транспонирование матрицы.
8. Умножение двух квадратных матриц.
9. Сложение двух квадратных матриц.

В задачах 10—13 имя файла последовательного доступа передавать через обобщенную символьную переменную.

10. Создать файл последовательного доступа, записывая в него поочередно фамилию, имя, отчество сотрудника (три символьные величины), год рождения (числовая величина).
11. В задаче 10 вывести содержимое файла в виде таблицы на ПУ и на экран дисплея.
12. Файл последовательного доступа содержит информацию об абонентах в следующем порядке: фамилия абонента (символьная величина), номер телефона (числовая величина). По заданному номеру телефона найти фамилию абонента.

13. В задаче 12 по фамилии абонента найти номер его телефона.

14. Определить день недели для заданной даты. Дата задается как значение символьной переменной в виде ДД.ММ.ГГГГ, где ДД — число, ММ — месяц, ГГГГ — год.

15. Построить гистограмму для N чисел ($N \leqslant 20$). Значение N вводится через обобщенную переменную. Числа вводятся из файла последовательного доступа, имя которого передается через обобщенную символьную переменную.

Задание II уровня. Для решения задачи составить подпрограмму (п/п), оформив ее как отдельный программный модуль с передачей вида функции (в задачах 1—8) или исходных данных, заданных в операторах DATA (в задачах 9—15), при помощи организации перекрытий подпрограммы и специального программного файла, в котором записаны соответствующие программные строки. Составить инструкцию по работе с п/п. Проверить работу п/п на примерах. При выполнении задания использовать вычислительные методы, изложенные в теоретическом введении к работе 6.

*Варианты задач II уровня.*

1. Методом итераций найти корень уравнения $x=\varphi(x)$ на отрезке $[a, b]$.

2. Методом Ньютона найти корень уравнения $f(x)=0$ на отрезке $[a, b]$.

3. Методом половинного деления найти корень уравнения $f(x)=0$ на отрезке $[a, b]$.

4. Вычислить определенный интеграл методом трапеций.

5. Вычислить определенный интеграл методом Симпсона.

6. Вычислить определенный интеграл методом трапеций с заданной точностью.

7. Решить дифференциальное уравнение $y'=f(x, y)$ методом Эйлера.

8. Решить систему двух дифференциальных уравнений первого порядка методом Эйлера.

9. Выполнить линейное интерполирование. Данные размещаются в операторах DATA.

10. Осуществить контроль знаний. Вопрос, пять ответов и номер правильного ответа расположены в операторах DATA.

11. Определить средний балл группы по результатам пяти экзаменов. В группе не более 25 человек. Результаты экзаменов заданы в операторах DATA.

12. Определить средний балл студента за пять лет обучения. Сведения об оценках (не более 50) заданы в операторах DATA.

13. Упорядочить массив, содержащий не более 50 элементов, по возрастанию или по убыванию (по выбору). Исходный массив задан в операторах DATA. Строку — проверку условия — также принимать из перекрывающего программного файла.

14. Решить систему линейных алгебраических уравнений методом Гаусса. Порядок системы не более 10. Коэффициенты и свободные члены системы заданы в операторах DATA.

15. В задаче 14 использовать метод Гаусса с выбором главного элемента в столбце.

Задание III уровня. В предлагаемых задачах выделить самостоятельные подзадачи. Оформить алгоритмы их решения в виде отдельных модулей, организовав передачу информации между модулями либо с использованием обобщенных переменных, либо с использованием файлов последовательного (или прямого) доступа, расположенных на ВЗУ. (Формирование исходных файлов на ВЗУ предусмотреть в программе.) Разработать схему взаимодействия модулей. Составить инструкцию по работе с программой. Проверить работу программы на ЭВМ.

*Варианты задач III уровня.*

1. Составить программу «секретарь». Программа должна выполнять следующие функции по заданной дате:

— сообщать (т. е. печатать) перечень фамилий тех, кому нужно позвонить;

— сообщать перечень фамилий или имен тех, с кем надо встретиться;

— сообщать о всех важных делах, запланированных на указанный день;

— сообщать перечень фамилий тех, кого следует поздравить с днем рождения.

2. Составить программу — подсказку выбора вида транспорта и номера рейса из Москвы в город $N$ с указанием стоимости проезда. Программа по заданному названию города $N$ должна сообщать следующие сведения:

— расписание движения самолетов из Москвы в город $N$;

— расписание движения поездов из Москвы в город $N$;

— расписание движения водного транспорта из Москвы в город $N$;

— стоимость проезда из Москвы в город $N$ каждым из трех видов транспорта.

3. Составить программу обработки экспериментальных данных, число которых $N \leqslant 40$. Данные должны вводиться в память ЭВМ. Обработка включает в себя следующее:

— вычисление математического ожидания M и дисперсии D по формулам:

$$M=\frac{1}{N}\sum_{i=1}^{N} a_i, \quad D=\frac{1}{N}\sum_{i=1}^{N} a_i^2-M^2,$$

где $a_i$ — экспериментальные данные;

— определение наибольшего и наименьшего элемента выборки данных и его порядкового номера в выборке;

— печать гистограммы.

4. Составить программу обработки результатов сдачи сессии студентами группы. В память ЭВМ должны вводиться следующие данные: фамилия студента и его порядковый номер по журналу, результаты сдачи сессии (5 экзаменов), признак наличия или отсутствия у студента постоянной общественной работы, признак участия или не участия в научно-исследовательской работе. Обработка включает в себя:

— построение и вывод таблицы результатов сдачи сессии студентами с указанием среднего балла студента и группы (для группы — по каждому экзамену и по всей сессии в целом);

— вывод сведений отдельно об отличниках, учащихся на хорошо и отлично, имеющих одну тройку, имеющих двойки;

— вывод сведений о студентах, участвующих в общественной работе;

— вывод сведений о студентах, участвующих в научно-исследовательской работе;

— вывод сведений о студентах, участвующих в общественной и научно-исследовательской работе.

5. Составить программу обработки итоговой таблицы чемпионата по футболу. Игры проходили в один круг, каждая команда встречалась с другой один раз. В программе предусмотреть ввод исходных данных. Обработка должна включать в себя:

— построение итоговой таблицы чемпионата с распределением команд согласно занятым местам с подсчетом очков, набранных каждой командой (за победу присуждается два очка, за ничью — одно очко, за поражение — ноль очков);

— определение и вывод названия команды (команд): одержавшей наибольшее количество побед, забившей в чемпионате наибольшее количество мячей, имеющей лучшую разность забитых и пропущенных мячей, пропустившей наименьшее количество мячей, одержавшей наибольшее число побед с крупным счетом (победой с крупным счетом называется победа с разницей мячей больше двух).

6. Составить программу обработки итоговой таблицы шахматного турнира. В программе предусмотреть ввод исходных данных (фамилии шахматистов и результаты их встреч друг с другом). Турнир проходил в один круг. Обработка должна включать:

— формирование итоговой таблицы турнира с расстановкой участников согласно занятым местам с подсчетом набранных очков (за победу присуждается одно очко, за ничью — пол-очка, за поражение — ноль очков);

— определение и вывод фамилии участника (участников):

одержавшего наибольшее число побед, завершившего больше всех партий вничью, потерпевшего больше всех поражений.

7. Составить программу подготовки сведений о студентах группы для их распределения по местам работы. В память ЭВМ вводятся сведения: фамилия студента; пол (один символ *м* или *ж*); все его оценки за период обучения; признаки: проживания студента (один символ: *м* — москвич, *п* — живет в Подмосковье, *и* — иногородний), семейного положения студента (0 — холост, 1 — женат), наличия детей (0 — нет, 1 — есть). Выдавать по требованию следующие сведения:

— список студентов по убыванию среднего балла за период обучения;

— список студентов-москвичей по убыванию среднего балла;

— список студентов, живущих в Подмосковье, по убыванию среднего балла;

— список студентов-иногородних по убыванию среднего балла;

— список девушек по убыванию среднего балла;

— список семейных студентов, а также имеющих детей, по убыванию среднего балла;

— список юношей по убыванию среднего балла.

8. Составить программу, помогающую отобрать студентов для участия в стройотряде. В память ЭВМ вводятся следующие сведения: фамилия студента, пол, признаки: успеваемости (0 — успевает на тройки; 1 — на тройки и четверки; 2 — на тройки, четверки и пятерки; 3 — на четверки и пятерки; 4 — отличник), участия в общественной жизни (0 — нет, 1 — участвует), овладения строительной специальностью (0 — нет, 1 — повар, 2 — каменщик, 3 — бетонщик, 4 — плотник, 5 — стропальщик, 6 — шофер при наличии прав), в случае овладения несколькими специальностями в качестве признака фигурирует число, каждая цифра которого — соответствующая специальность. По требованию выдавать следующие сведения:

— список студентов-поваров, имеющих еще какую-нибудь строительную специальность;

— список студентов, владеющих как минимум двумя специальностями, кроме повара;

— список студентов, имеющих права на вождение автомобиля и еще одну строительную специальность, кроме повара;

— список студентов, имеющих хотя бы одну из специальностей: каменщик, плотник, стропальщик.

Примечания. 1). В стройотряд не берутся студенты, успевающие на одни тройки.

2). В каждом списке сначала перечислять студентов, участвующих в общественной жизни, а затем — не участвующих.

9. Составить программу назначения студентов на стипендию.

В память ЭВМ вводятся сведения: фамилия студента, его оценки в сессию (5 оценок), признаки: участия в общественной жизни (0 — нет, 1 — да), участия в научно-исследовательской работе (0 — нет, 1 — да). По требованию выдавать следующие сведения:

— список студентов, назначенных на повышенную (на 25%) стипендию за отличную учебу, активное участие в общественной жизни и научно-исследовательской работе;

— список студентов, назначенных на повышенную (на 15%) стипендию за хорошую и отличную учебу (не менее двух пятерок), активное участие в общественной жизни и научно-исследовательской работе;

— список студентов, назначенных на обычную стипендию, но имеющих не более одной тройки;

— список студентов, не назначенных на стипендию.

10. Составить программу, помогающую администратору гостиницы. В гостинице восемь этажей. На первом этаже — административные помещения и ресторан; на втором — восемь двухкомнатных двухместных номеров-люксов; на третьем и четвертом — по шестнадцать одноместных номеров; на остальных этажах — по шестнадцать двухместных однокомнатных номеров на каждом этаже. О каждом номере известна следующая информаци: номер свободен, номер забронирован, номер занят, в номере живет один человек (мужчина или женщина), в этом случае известна дополнительная информация: с правом подселения или нет (если номер двухместный), прибыл в командировку, прибыл в частном порядке. По требованию администратору выдавать информацию:

— о свободных номерах;

— о свободных мужских местах в номерах с указанием номера;

— о свободных женских местах в номерах с указанием номера;

— о гостях, приехавших в командировку с указанием номеров, в которых они живут;

— о гостях, приехавших в частном порядке;

— о забронированных номерах.

Примечание. Номера в гостинице имеют три цифры. Самая старшая (левая) цифра — номер этажа, оставшиеся две — порядковый номер на этаже.

11. Составить программу, помогающую читателю найти нужную книгу в библиотеке. В ЭВМ о каждой книге хранится следующая информация: автор, название, издательство, год издания. Сведения о книгах хранятся в файле последовательного доступа в символьном виде, данные о книгах разделены символом *. Сведения о книгах по программированию хранятся в файле с именем "PROGR", по математике — "MATHEM", по физике — "PHYSIC", по химии — "CHEMIS", по истории — "HISTOR". По требованию выдавать следующую информацию:

— сведения о всех книгах на заданную тематику;

— поиск заданной книги в заданном разделе (сообщаются сведения об авторе, название книги и раздела);

— сведения о книгах на заданную тематику заданного издательства;

— сведения о книгах, вышедших в свет в текущем году (т. е. сведения о новых поступлениях), по всем темам или по заданной;

— сведения о книгах заданного издательства по всем темам;

— сведения о книгах одного автора по всем темам или по заданной.

Примечание. Для упрощения обработки считать, что каждая книга имеет одного автора.

12. Составить программу, помогающую узнать сведения о магазине. В ЭВМ хранятся сведения о магазинах (в символьном виде): название улицы, номер дома, номер телефона (семь цифр). Данные хранятся в файле последовательного доступа, сведения о разных магазинах разделены символом * (звездочка). Файлы созданы из сведений об однотипных магазинах, всего типов магазинов — семь. Сведения о булочных хранятся в файле с именем "BAKER", сведения о гастрономах — "PROV", сведения об овощных магазинах — "VEG", сведения о книжных магазинах — "BOOK", сведения об автомагазинах — "AVTO", сведения об универмагах — "STORE", сведения о спортивных магазинах — "SPORT". По требованию выдавать следующую информацию:

— сведения о всех магазинах заданного типа;

— сведения о всех магазинах, находящихся на заданной улице;

— адрес магазина по заданному номеру телефона;

— сведения о всех магазинах, номер телефона которых содержит заданные три первые цифры.

13. Составить программу, помогающую выбрать рецепт приготовления пищи. В ЭВМ хранятся следующие сведения о рецептах: название блюда (например, борщ, салат мясной, шашлык по-карски и т. п., т. е. название может быть составным), состав компонентов (не более семи), рецепт приготовления в виде символьной переменной длиной не более 255 символов. Первым символом компонента обязательно является цифра (с целью отличия от рецепта). Данные хранятся в файле последовательного доступа, сведения о рецептах разделены символом — * (звездочка). Файлы созданы из рецептов по признаку очередности блюд, всего четыре разновидности. Сведения о закусках, салатах и т. п. хранятся в файле с именем "SALAT", сведения о первых блюдах — в файле "ONE", сведения о вторых блюдах — в файле "TWO", сведения о третьих блюдах — в файле "DESERT". По требованию выдавать следующую информацию:

— рецепт приготовления заданного блюда;

— компоненты для приготовления заданного блюда;
— всю информацию о блюдах заданного файла;
— наименования блюд из заданного файла.

14. Составить программу, помогающую сотрудникам ГАИ. В памяти ЭВМ в файле последовательного доступа хранится следующая информация об автомобилях: регистрационный номер автомобиля (записывается в символьном виде как БNNNNББ, где Б — буква, N — цифра), цвет автомобиля, год выпуска, адрес проживания владельца. Данные о разных автомобилях разделены символом *. Файлы созданы из сведений об автолюбителях, имеющих автомобиль заданной марки, всего три файла. Сведения о владельцах «Запорожцев» хранятся в файле "ZAP", о владельцах автомобиля «Жигули» — в файле "LADA", о владельцах автомобиля «Волга» — в файле "VOLGA". По требованию выдавать сведения об автолюбителях, имеющих:
— автомобиль заданной марки определенного цвета;
— автомобиль с заданным регистрационным номером;
— автомобиль заданной марки с известной цифровой частью регистрационного номера;
— автомобиль заданного цвета;
— автомобиль заданной марки заданного цвета с не полностью известными цифрами цифровой части регистрационного номера.

15. Составить программу, помогающую сообщать сведения о местах на авиарейсы из Москвы в Ленинград на календарный месяц (30 дней). В файле прямого доступа хранится информация о местах на рейс на заданное число в виде матрицы, в которой номер строки — ряд, номер столбца — номер места в ряду (всего 25 рядов, по шесть мест в ряду). Элемент матрицы может иметь следующие значения: 0 — место свободно, 1 — место продано, 2 — место забронировано. Считать, что в день имеется один рейс. Файлы имеют имена следующего типа: "DAYNN", где: NN — число, на которое хранится информация. Например, "DAY15" — в этом файле хранятся сведения о местах на рейс, отправляющийся 15 числа. По требованию сообщать следующую информацию:
— о количестве свободных мест на заданное число;
— о проданных местах на заданное число;
— о проданных местах на весь месяц;
— о брони на весь месяц;
— о брони на заданное число.

Вопросы для самопроверки.
1. Что такое модульное программирование?
2. Что такое структурное программирование?
3. Каким образом можно хранить отдельные модули в памяти ЭВМ?
4. Как объединить несколько различных модулей в одну программу? Команда APPEND. Что произойдет, если в загружаемом

модуле имеются строки, номера которых совпадают с номерами строк программы, находящейся в оперативной памяти? В каких случаях целесообразно использовать такой способ организации программ?

5. Как организовать поочередное выполнение модулей? Оператор CHAIN и порядок его выполнения. В каких случаях целесообразно использовать подобную организацию программы?

6. Каким образом может быть организована передача данных из одного модуля в другой при их поочередном выполнении? Оператор COMMON.

7. Каким образом можно заменить или добавить в программу строки в процессе ее выполнения? Программы с перекрытием. Оператор OVERLAY, отличие его от команды APPEND.

8. В каких случаях целесообразна организация программ с перекрытием? Какие преимущества и в каких случаях дает такая организация программ?

9. Как можно использовать оператор OVERLAY при составлении программ, реализующих численные методы решения дифференциальных уравнений, вычисления определенных интегралов и т. п. в общем виде?

10. Как организовать данные для обеспечения возможности их использования различными модулями? Перечислить различные способы передачи данных из одного модуля в другой при их поочередном выполнении.

## ДОПОЛНИТЕЛЬНЫЕ ЗАДАЧИ

1. Вычислить:

$1+\frac{1}{2}+\ldots+\frac{1}{10}$;

$5+8+11+\ldots+35$; $1+3+3^2+\ldots+3^{10}$;

$1+\frac{1}{2}+\frac{1}{2^2}+\ldots+\frac{1}{2^{10}}$; $\frac{2}{3}+\frac{3}{4}+\frac{4}{5}+\ldots+\frac{10}{11}$;

$4\cdot6\cdot8\cdot\ldots\cdot20$; $2+2^2+2^3+\ldots+2^{10}$; $15\cdot16\cdot17\cdot\ldots\cdot25$;

$y=\frac{1}{2}+2x^3$ при $x=-5, -4, \ldots, 5$;

$y=2x+x^2$ при $x=2, 4, \ldots, 20$;

$y=10x^2$ при $x=-2, -1.8, \ldots, 2$.

2. Получить и напечатать последовательность из $n$ натуральных чисел, образованную по следующему правилу: каждое число в последовательности, начиная с третьего, получается сложением двух предыдущих чисел. Первые два числа одинаковы и равны 1. (Такая последовательность называется числами Фибоначчи.)

3. Вычислить $n$-й член последовательности, образованный дробями: 1/1, 2/1, 3/2, ..., т. е. числитель (знаменатель) следующего члена последовательности получается сложением числителей (знаменателей) двух предыдущих членов. Числители первых двух членов равны 1 и 2, знаменатели — 1 и 1.

4. В задаче 3 вычислить член последовательности, который отличается от предыдущего члена не более чем на 0,001.

5. По древней легенде, мудрец, который изобрел шахматы, потребовал от персидского шаха такое количество пшеницы, чтобы им можно было покрыть шахматную доску, положив на первую клетку одно зерно, на вторую клетку — 2, на третью — 4 и т. д., т. е. помещая на каждую следующую клетку в два раза больше зерен, чем на предыдущую. Какое количество зерна может покрыть доску? (Считать, что в одном грамме 15 зерен.) С какой точностью получен результат?

6. Считая, что Земля — идеальная сфера с радиусом $R\approx6350$ км, определить расстояние до линии горизонта с высоты 1, 2, ..., 10 км.

7. При заданном $x$ вычислить приближенно сумму

$$S=e^x=\sum_{i=0}^{\infty}\frac{x^i}{i!}\quad (0!=1),$$

прекращая вычисления, когда очередной член по абсолютной величине будет меньше 0,0001.

8. При заданном $x$ вычислить сумму

$$S=1-\frac{2}{3}x+\frac{3}{4}x^2-\frac{4}{5}x^3+\ldots+\frac{11}{12}x^{10}.$$

9. При заданном $x$ вычислить сумму

$$S=\sin x=\sum_{i=0}^{\infty}(-1)^i\frac{x^{2i+1}}{(2i+1)!},$$

прекращая вычисления, когда очередной член суммы по абсолютной величине будет меньше 0,0001.

10. Вычислить

$$S=\cos x=\sum_{i=0}^{\infty}(-1)^i\frac{x^{2i}}{(2i)!}$$

при значениях $x$, изменяющихся в пределах от $\pi/10$ до $\pi/2$ с шагом $\pi/10$. Вычисления прекращать, когда очередной член по абсолютной величине станет меньше 0,0001.

11. Решить уравнение $x=0{,}5(\sin x^2-1)$, вычисляя $x_{n+1}=0{,}5(\sin x_n-1)$, $n=1,2,\ldots$, при $x_0=0{,}4$. Вычисления прекратить, когда будет выполнено условие $|x_{n+1}-x_n|\leqslant 0{,}0001$.

12. Для заданных $a$ и $b$ получить

$$c=\begin{cases}\max(a,\ b), & \text{если } a>0,\\ \min(a,\ b), & \text{если } a\leqslant 0.\end{cases}$$

13. Задано $n$ пар чисел $a$ и $b$. Получить $n$ чисел $c=\max(a,\ b)$.

14. Для заданных $a$, $b$, $c$ получить $z=\max(\min(a,\ b),\ c)$.

15. Для функций, заданных графически на рис. 5.1, определить значение $y$ при заданном значении $x$.

16. Определить, принадлежит ли заданная точка $(x,\ y)$ фигуре, изображенной на рис. 5.2.

17. Сформировать массив размером $10\times 19$ по следующему правилу. В первой строке средний (десятый) элемент равен 1, остальные элементы нулевые, $(i+1)$-й элемент каждой последующей строки получается сложением $i$-го и $(i+2)$-го элементов предыдущей строки. Ненулевые элементы полученного массива образуют так называемый треугольник Паскаля.

В задачах 18—33 используются термины: *вектор* — одномерный массив; *матрица* — двумерный массив; *главная диагональ матрицы* размером $n\times n$ — диагональ, на которой расположены элементы $a_{11}$, $a_{nn}$; *побочная главная диагональ матрицы* — диагональ, на которой расположены элементы $a_{1n}$, $a_{n1}$; *единичная матрица* — квадратная матрица, все элементы которой нулевые, за исключением элементов главной диагонали, равных 1; *периметр матрицы* включает элементы первого и последнего столбцов и первой и последней строк.

18. Задана матрица размером $N\times M$.

а) Просуммировать элементы каждой строки. Результат получить в виде вектора размером $N$.

б) Просуммировать элементы каждого столбца. Результат получить в виде вектора размера $M$.

19. Задана матрица размером $N\times M$.

а) Найти максимальный элемент каждой строки. Результат получить в виде вектора размером $N$.

б) Найти минимальный элемент каждого столбца. Результат получить в виде вектора размером $M$.

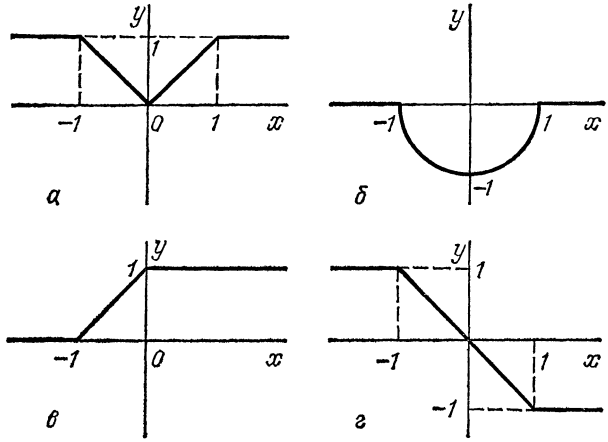



Рис. 5.1

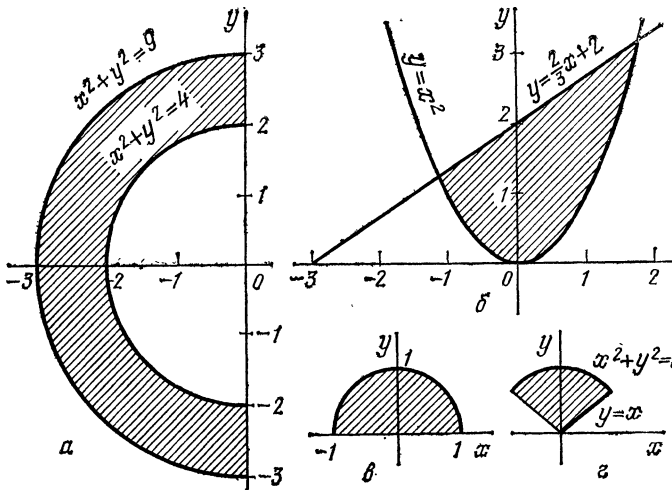



Рис. 5.2

20. Задана квадратная матрица размером $N \times N$. Переставить местами главную и побочную главную диагонали.

21. Сформировать матрицу размером $N \times 3N$, составленную из трех единичных квадратных матриц размером $N \times N$.

22. Просуммировать элементы, расположенные по периметру матрицы размером а) $N \times M$, б) $N \times N$.

23. Заполнить нулями элементы квадратной матрицы, расположенные по ее периметру (использовать один цикл).

24. Задана матрица размером $N \times N$. Просуммировать элементы, расположенные на всех диагоналях, параллельных главной, включая главную. Результат получить в виде вектора размером $2N-1$.

25. Из матрицы размером $N \times N$ сформировать одномерный массив по следующему правилу:

а) элементы первой строки — в порядке возрастания индексов столбцов, элементы второй строки — в порядке убывания индексов столбцов, элементы третьей строки — в порядке возрастания индексов столбцов, элементы четвертой строки — в порядке убывания индексов столбцов и т. д.;

б) элементы первого столбца — в порядке возрастания индексов строк, элементы второго столбца — в порядке убывания индексов строк, элементы третьего столбца — в порядке возрастания индексов строк, элементы четвертого столбца — в порядке убывания индексов строк и т. д.;

в) элементы $N$-й строки — в порядке убывания индексов столбцов, элементы $(N-1)$-й строки — в порядке возрастания индексов столбцов, элементы $(N-2)$-й строки — в порядке убывания индексов столбцов, элементы $(N-3)$-й строки — в порядке возрастания индексов столбцов и т. д.;

г) элементы $N$-го столбца — в порядке убывания индексов строк, элементы $(N-1)$-го столбца — в порядке возрастания индексов строк, элементы $(N-2)$-го столбца — в порядке убывания индексов строк, элементы $(N-3)$-го столбца — в порядке возрастания индексов строк и т. д.

26. В матрице $N \times N$ ($N$ — четное) поменять местами:

а) элементы первой и второй строк, элементы третьей и четвертой строк и т. д.;

б) элементы первого и второго столбцов, элементы третьего и четвертого столбцов и т.д.

27. В матрице $N \times N$ поменять местами:

а) элементы первой и последней строк, элементы второй и $(N-1)$-й строк и т. д. до средней строки (если $N$ — нечетное) или до строк с номерами $N/2$ и $N/2+1$ (если $N$ — четное);

б) элементы первого и последнего столбцов, элементы второго и $(N-1)$-го столбцов и т. д. до среднего столбца (если $N$ — нечетное) или до столбцов с номерами $N/2$ и $N/2+1$ (если $N$ — четное).

28. Матрицу $N \times N$ повернуть (без использования вспомогательных массивов):

а) на 90° вправо,

б) на 180° вправо, в) на 90° влево.

29. В матрице $N \times N$ поменять местами элементы:

а) симметрично относительно главной диагонали,

б) симметрично относительно побочной главной диагонали.

30. Просуммировать элементы квадратной матрицы размером $N \times N$:

а) расположенные в ее верхней четверти, ограниченной главной и побочной главной диагоналями, включая элементы, расположенные на диагоналях;

б) расположенные в нижней четверти, ограниченной диагоналями, включая диагонали;

в) расположенные в правой четверти, ограниченной диагоналями, включая диагонали;

г) расположенные в левой четверти, ограниченной диагоналями, включая диагонали.

31. Для матрицы размером $N \times N$ заполнить единицами:

а) нижнюю половину, включая среднюю строку, если $N$ — нечетное, за исключением элементов, расположенных справа от главной диагонали;

б) правую половину, включая средний столбец, если $N$ — нечетное, за исключением элементов, расположенных левее главной диагонали;

в) верхнюю половину, за исключением элементов, расположенных правее побочной главной диагонали;

г) верхнюю половину, за исключением элементов, расположенных левее главной диагонали;

д) левую половину, за исключением элементов, расположенных правее побочной главной диагонали.

32. В матрице $N \times N$ поменять местами элементы:

а) расположенные в верхней и нижней четвертях, ограниченных главной и побочной главной диагоналями (за исключением элементов, расположенных на диагоналях);

б) расположенные в левой и правой четвертях, ограниченных главной и побочной диагоналями (за исключением элементов, расположенных на диагоналях).

33. Сформировать вектор размером 4, элементами которого являются соответствующие суммы элементов матрицы размером $N \times N$:

а) первым элементом — сумма элементов матрицы над главной диагональю, вторым элементом — сумма элементов матрицы под главной диагональю, третьим элементом — сумма элементов матрицы над побочной главной диагональю, четвертым элементом — сумма элементов матрицы под побочной главной диагональю;

б) первым элементом — сумма элементов, расположенных в верхней четверти, ограниченной главной и побочной главной диагоналями (без элементов диагоналей), вторым — сумма элементов, расположенных в правой четверти, ограниченной главной и побочной главной диагоналями, третьим — сумма элементов, расположенных в нижней четверти, ограниченной главной и побочной главной диагоналями, четвертым — сумма элементов, расположенных в левой четверти, ограниченной главной и побочной главной диагоналями.

34. Заданы массивы $X$, $Y$ размером $N$. Массив $X$ содержит индексы от 1 до $N$ (все элементы различны). Элементы массива $Y$ переставить таким образом, чтобы элемент, находящийся на $I$-м месте, оказался на месте элемента, индекс которого указан в $X(I)$.

35. В массиве, заполненном наполовину, продублировать все элементы с сохранением порядка следования (например, задан массив $X=(3, 8, \ldots)$, получить массив $X=(3, 3, 8, 8, \ldots)$).

36. Из заданного массива удалить повторяющиеся элементы. Массив сжать.

37. Из массива удалить элементы, которые встречаются только один раз. Массив сжать.

38. Разреженную матрицу (содержащую незначительное количество отличных от нуля элементов) представить в виде трех одномерных массивов, элементами которых являются: номера строк, номера столбцов и значения ненулевых элементов.

39. Задан вектор размером $N$. Определить индекс элемента, наиболее близкого к среднему значению элементов этого вектора.

40. Задан вектор размером $N$. Определить среднее значение его элементов, отбросив предварительно минимальный и максимальный элементы.

41. В заданном массиве поменять местами четные и нечетные элементы (первый со вторым, третий с четвертым и т. д.).

42. В произвольной последовательности чисел (не более 100), заданной в виде одномерного массива, найти самую длинную подпоследовательность, упорядоченную по возрастанию.

43. Найти норму $S$ матрицы $A$ размером $N \times N$ по формуле:

$$S = \max_i \sum_{j=1}^{n} |a_{ij}|.$$

44. Задана последовательность $n$ чисел $a_1$, $a_2$, ..., $a_n$. Найти ее среднее геометрическое $\left(\sqrt[n]{a_1 \cdot a_2 \cdot \ldots \cdot a_n}\right)$.

45. Задана последовательность чисел. Упорядочить их таким образом, чтобы в начале были расположены все отрицательные числа с сохранением порядка их следования, а затем положительные (включая 0) также с сохранением порядка следования. Вспомогательный массив не использовать.

46. Составить программу — тест: «Легко ли Вы приобретаете друзей»? Участник должен выполнить три задания:

I. Ответить на вопросы (в скобках — очки: первое число — «ДА», второе — «НЕТ»): 1. Обманул ли Вас когда-нибудь друг? (15, 5); 2. Считаете ли Вы себя знатоком людей? (20, 2); 3. Предпочитаете ли Вы путешествовать в одиночестве? (4, 12); 4. Любите ли Вы бывать в обществе? (15,, 8); 5. Знаете ли Вы жильцов своего подъезда? (12, 6).

II. Указать номера верных утверждений (в скобках очки: первое — номер указан, второе — нет): 1. Дружбу не могут заменить другие увлечения (5, 15); 2. В беде тебя покинет и друг (3, 12); 3. Сначала подумай о друге, потом о себе (20, 1); 4. Не следует требовать от друзей слишком многого (20, 6); 5. Чем больше друзей, тем лучше (5, 10).

III. Укажите номера черт, которыми должен обладать друг: 1. Откровенный (30); 2. Верный (40); 3. Находчивый (10); 4. Красивый (6); 5. Ителлигентный (8); 6. Скромный (15); 7. Интересный (5).

Определение результатов тестирования:

1) до 140 очков — друзей Вы приобретаете с огромным трудом. Вы не верите в то, что можно встретить настоящего друга. Больше доверия к людям;

2) от 141 до 180 очков — Вы хорошо знаете людей. У Вас есть друзья. Они Вас не подведут;

3) от 181 очка — у Вас много друзей. Но настоящие ли это друзья? Вы их легко приобретаете, но можете легко обидеть и потерять.

47. В заданном тексте (слова разделяются одним пробелом) поменять местами первое и последнее слово.

48. В заданном тексте (слова разделяются одним пробелом) найти и, если есть, напечатать все слова — палиндромы (т. е. такие слова, которые одинаково читаются слева направо и справа налево).

49. Определить, является ли заданный текст палиндромом (пробелы между словами во внимание не принимать).

50. Осуществить все циклические сдвиги слов в заданном тексте влево (слова разделены одним пробелом). Каждый полученный при этом текст вывести на печать.

# СПИСОК АЛГОРИТМОВ *)

### Простейшие алгоритмы

Вычисление суммы $n$ слагаемых (126). Вычисление произведения $n$ сомножителей (128). Вычисление факториала (128, 170). Табулирование функции (128). Определение принадлежности точки треугольнику (139, задача 14).

### Обработка массивов

Суммирование элементов массива (152). Суммирование двух массивов (152). Вычисление следа матрицы (152). Транспонирование матрицы (153). Умножение матрицы на вектор (154). Умножение матрицы на матрицу (154). Удаление элемента из массива (155). Включение элемента в заданную позицию массива (155). Включение элемента в упорядоченный массив с сохранением упорядоченности (155). Удаление строки из матрицы (156). Включение строки в матрицу (156). Перестановка элементов в массиве (157). Перестановка строк в матрице (157). Поиск максимального (минимального) элемента в массиве (158). Преобразование матрицы в одномерный массив (159). Определение количества элементов массива, удовлетворяющих заданному условию (240). Суммирование элементов, удовлетворяющих заданному условию (240). Объединение двух массивов в один с чередованием элементов исходных массивов (241). Инвертирование массива (241). Поиск заданного элемента в массиве (242). Циклический сдвиг элементов массива (243). Упорядочение массива (244). Проверка массива на упорядоченность (245). Поиск заданного элемента в упорядоченном массиве (бинарный поиск) (246). Объединение двух упорядоченных массивов в один с сохранением упорядоченности (247). Сравнение двух упорядоченных массивов (248).

### Вычислительные методы

Метод половинного деления для нахождения корня уравнения (175). Линейное интерполирование (194). Квадратичное интерполирование (195). Метод итераций для нахождения корня уравнения (197). Метод Ньютона для нахождения корня уравнения (199). Метод трапеций для вычисления определенного интеграла (200).

*) В скобках указываются страницы. Если указан и номер задачи, см. также указание к ее решению.

Метод Симпсона для вычисления определенного интеграла (202). Метод Гаусса с выбором главного элемента в столбце для решения системы линейных уравнений (205). Метод Гаусса для вычисления определителя (204). Метод Гаусса — Жордана для решения системы линейных уравнений (207). Метод Эйлера для решения обыкновенных дифференциальных уравнений (207). Вычисление определенного интеграла с заданной точностью (209).

### Обработка текстов

Выделение из текста символа (или цепочки символов), расположенного в заданной позиции (271). Определение позиции, в которой располагается заданный символ или цепочка символов (271). Удаление из текста цепочки символов (271). Добавление в текст заданной цепочки символов (271). Выделение слова из текста (273). Разделение текста на строки (271, 272). Определение того, является буква гласной или согласной (274). Определение количества слогов в слове (282, задача 10). Расположение русских слов по алфавиту (286, задача 11).

### Работа с натуральными числами. Разные задачи

Определение четности числа (302). Выделение цифр числа (302). Нахождение делителей числа (303). Нахождение простых чисел (304, 305). Сложение многоразрядных чисел (307). Умножение многоразрядного числа на число (308). Вычисление факториала большого числа (311). Перевод числа из одной системы счисления в другую (316, задача 3). Определение дня недели по дате (310). Составление календаря (317, задачи 10, 11). Построение графика биоритмов (417, задача 12). Игра «Жизнь» (258, задача 13, 265, задача 10). Контроль знаний (139, задача 15, 260, задача 15, 265, задачи 12, 13). Получение случайных чисел из заданного интервала (322). Получение различных целых случайных чисел из заданного интервала (323). Игра Баше (324). Игра «Быки и коровы» (326). Игра «Набери сумму» (334, задачи 5, 6, 7). Игра «Карусель — лото» (335, задачи 8, 9). Игра «Морской бой» (336, задачи 10, 11, 12).

Приложение 1

## СИМВОЛЫ БЕЙСИКА В ПОРЯДКЕ СТАРШИНСТВА И ИХ СЕМИРАЗРЯДНЫЕ ВОСЬМЕРИЧНЫЕ КОДЫ

| Символ | Восьмеричный код | Символ | Восьмеричный код | Символ | Восьмеричный код |
|---|---|---|---|---|---|
| ␣ | 040 | > | 076 | ↑ | 136 |
| ! | 041 | ? | 077 | Ю | 140 |
| " | 042 | @ | 100 | А | 141 |
| # | 043 | A | 101 | Б | 142 |
| $ | 044 | B | 102 | Ц | 143 |
| % | 045 | C | 103 | Д | 144 |
| & | 046 | D | 104 | Е | 145 |
| ' | 047 | E | 105 | Ф | 146 |
| ( | 050 | F | 106 | Г | 147 |
| ) | 051 | G | 107 | Х | 150 |
| * | 052 | H | 110 | И | 151 |
| + | 053 | I | 111 | Й | 152 |
| , | 054 | J | 112 | К | 153 |
| - | 055 | K | 113 | Л | 154 |
| . | 056 | L | 114 | М | 155 |
| / | 057 | M | 115 | Н | 156 |
| 0 | 060 | N | 116 | О | 157 |
| 1 | 061 | O | 117 | П | 160 |
| 2 | 062 | P | 120 | Я | 161 |
| 3 | 063 | Q | 121 | Р | 162 |
| 4 | 064 | R | 122 | С | 163 |
| 5 | 065 | S | 123 | Т | 164 |
| 6 | 066 | T | 124 | У | 165 |
| 7 | 067 | U | 125 | Ж | 166 |
| 8 | 070 | V | 126 | В | 167 |
| 9 | 071 | W | 127 | Ь | 170 |
| : | 072 | X | 130 | Ы | 171 |
| ; | 073 | Y | 131 | З | 172 |
| < | 074 | Z | 132 | Ш | 173 |
| = | 075 | [ | 133 | Э | 174 |
| | | \ | 134 | Щ | 175 |
| | | ] | 135 | Ч | 176 |

Приложение 2

## СЛУЖЕБНЫЕ СЛОВА И СОКРАЩЕНИЯ

### Служебные слова бейсика

| | |
|---|---|
| APPEND | — присоединить |
| CHAIN | — сочленить |
| CLOSE | — закрыть |
| COMMON | — общий |
| DATA | — данные |
| DELETE | — вычеркнуть |
| END | — конец |
| FILE | — файл |
| FOR | — для |
| GO TO | — перейти к |
| IF | — если |
| INPUT | — ввести |
| LET | — пусть |
| LINE | — строка |
| LIST | — показать на экране текст программы |
| NAME | — имя |
| NEW | — новый |
| NEXT | — следующий |
| NONAME | — без имени |
| OLD | — загрузить |
| ON | — в зависимости от |
| OPEN | — открыть |
| OUTPUT | — вывести |
| OVERLAY | — перекрыть |
| PRINT | — напечатать |
| PRINT USING | — напечатать с использованием формата |
| READ | — читать |
| READY | — готов |
| RENAME | — переименовать |
| REPLACE | — заменить |
| RESTORE | — восстановить |
| RETURN | — возврат |
| RUBOUT | — стереть |
| RUN | — выполнить |
| RUNNH | — выполнить без вывода заголовка |
| SAVE | — сохранить |
| STEP | — шаг |
| STOP | — останов |
| STOP AT LINE *N* | — останов в строке *N* |
| THEN | — тогда |
| UNSAVE | — стереть с диска |

### Сокращения

| | |
|---|---|
| ASC — american standard code | — американский стандартный код |
| ATN — arctg | — арктангенс |
| CON — constant | — константа |
| CR — carriage return | — возврат каретки |
| CTRL — control | — управление |
| DEF FN — define function | — определить функцию |
| DEL — delete | — вычеркнуть |

| | |
|---|---|
| DIM — dimension | — размерность |
| GOSUB — go to subroutine | — перейти к подпрограмме |
| IDN — identical | — тождественный (MAT C = IDN — получение матрицы тождественного преобразования) |
| INT — integer | — целый |
| LEN — length | — длина |
| POS — position | — позиция |
| REM — remark | — примечание |
| RESEQ — resequence | — изменение последовательности |
| RND — random | — случайный |
| SCR — scratch | — стереть |
| SEG — segment | — часть |
| SGN — sign | — знак |
| SQR — square root | — квадратный корень |
| STR — string | — строка символов |
| SUB — subsequence | — последовательность |
| TAB — tabulation | — табулирование |
| TRM — trimming trailing blanks of a string | — удаление конечных пробелов из строки символов |
| VAL — value | — значение |
| ZER — zero | — нуль |

Приложение 3

## РАБОТА НА БЕЙСИКЕ НА ЭВМ СЕРИИ СМ В СИСТЕМЕ ОС РВ

Начало сеанса. После включения дисплея на его экране обычно появляется специальный символ, называемый курсором. Для начала сеанса следует нажать клавишу ВК. В ответ на экране дисплея появляется специальный символ >, называемый символом приглашения. После этого вы можете начать сеанс. Для этого следует набрать на клавиатуре HEL (HELLO) и нажать клавишу ВК.

Далее система запросит ваш личный номер:

ACCOUNT OF NAME?

Следует сообщить свой номер, а затем нажать ВК. После этого система запросит ваш пароль:

PASSWORD?

Следует набрать на клавиатуре свой пароль и нажать клавишу ВК. Символы пароля при этом не высвечиваются на экране. Если пароль набран верно, то выводится сообщение о готовности к работе. Для вызова бейсика следует набрать команду BAS (BASIC) и нажать клавишу ВК. На экране должно появиться READY, что означает готовность бейсика к работе.

Конец сеанса. При окончании работы с бейсиком следует набрать команду BYE и нажать ВК. На экране появится символ приглашения >. Для полного окончания сеанса следует еще раз набрать команду BYE.

Распечатка программных файлов и файлов данных. Распечатка текстов программ на бейсике (программные файлы) и результатов выполнения программ на ПУ при работе на ЭВМ серии СМ осуществляется следующим образом.

Печать текста программы на ПУ осуществляется после выхода из бейсика при помощи команды

```
>PIP  LI:=NAME.BAS
```

где NAME — имя программы.

Вывод результатов работы программы на ПУ осуществляется в два этапа:

1. Во время выполнения программы необходимо выводить результаты ее работы в файл последовательного доступа (см. п. 13.2 главы 3). При этом целесообразно для контроля выводить результаты также на экран дисплея.

2. После выхода из бейсика вывести на ПУ сформированный файл при помощи команды

```
>PIP  LI:=NAME.DAT
```

где NAME — имя файла данных.

Просмотр каталога файлов. Для просмотра каталога файлов пользователя применяется команда:

```
CAT
```

По этой команде на экран дисплея будут выведены все имена программ пользователя в виде:

```
NAME.BAS;N
```

где NAME — имя программы; BAS — спецификация программного файла на бейсике; N — номер версии программы.

Приложение 4

## РАБОТА НА БЕЙСИКЕ НА ЭВМ СЕРИИ «ЭЛЕКТРОНИКА» В СИСТЕМЕ ФОБОС (ФОДОС)

Начало сеанса. После включения ЭВМ и дисплея и запуска (старта) операционной системы ФОБОС (ФОДОС) на экране появится символ приглашения «.» (точка). Для вызова бейсика следует набрать одну из команд:

```
.RU BASIC
.R BASIC
.BASIC
```

Разрешается вместо BASIC использовать сокращенное имя BAS.

Появление на экране дисплея сообщения READY свидетельствует о готовности бейсика к работе.

Конец сеанса. Для окончания работы в бейсике следует набрать команду

```
BYE
```

Появление на экране дисплея символа приглашения «.» свидетельствует о выходе из бейсика.

Просмотр каталога файлов. Для просмотра каталога файлов в системе ФОБОС (ФОДОС) следует сначала выйти из бейсика, а затем, после появления символа приглашения, набрать команду:

```
.DIR
```

По этой команде на экране дисплея высветится список всех файлов пользователя, хранящихся на ВЗУ. Программные файлы — это файлы, имеющие спецификацию «.BAS»; файлы данных — спецификацию «.DAT».

Приложение 5

## СООБЩЕНИЯ ОБ ОШИБКАХ

Формат сообщения об ошибке:

? R AT LINE N

где: R — полное или сокращенное сообщение типа ошибки; N — номер строки, в которой допущена ошибка (если он может быть указан).

Типы ошибок.

1. ? ARGUMENT ERROR (?ARG)

Ошибка при записи аргумента функции.

2. ? ARRAYS TOO LARGE (?ATL)

Не хватает оперативной памяти для размещения массива.

3. ? BAD DATA READ (?BDR)

Ошибка при вводе данных

4. ? BAD DATA RETYPE FROM ERROR (?BRT)

Введены неправильные данные. Ввод данных требуется повторить.

5. ? BAD LOG (?BLG)

Значение аргумента стандартной функции LOG или LOG10 равно нулю или отрицательно. Предупреждение — программа продолжает выполняться, но значение функции равно нулю.

6. ? BUFFER STORAGE OVERFLOW (?BSO)

Не хватает оперативной памяти для размещения файла.

7. ? CANNOT DELETE FILE (?CDF)

Файл, указанный в команде UNSAVE, не может быть уничтожен.

8. ? CHANNEL ALREADY OPEN (?CAO)

Повторное использование оператора OPEN для открытия файла.

9. ? CHANNEL I/O ERROR (?CIE)

Допущена ошибка при считывании (записи) данных из файла (в файл).

10. ? CHANNEL NOT OPEN (?CNO)

Обращение к неоткрытому файлу.

11. ? COMMON OUT OF ORDER (?COO)

Несовпадение порядка переменных и массивов в операторе COMMON с порядком, заданным в предыдущем программном модуле.

12. ? CONTROL VARIABLE OUT OF RANGE (?CVO)

Значение переменной в операторе ON GOTO или ON GOSUB равно нулю или отрицательно.

13. ? DIVISION BY ZERO (?DVO)

Была предпринята попытка деления на нуль. Предупреждение — программа продолжает выполняться, но значение выражения равно нулю.

14. ? END NOT LAST (?ENL)

Оператор END является не последним в программе.

15. ? ERROR CLOSING CHANNEL (?ECC)

Ошибка при закрытии файла.

16. ? EXCESS INPUT IGNORED (?EII)

Количество введенных данных больше, чем требуется в операторе INPUT, предупреждение — программа продолжает выполняться, лишние данные игнорируются.

17. ? EXPONENTIATION ERROR (?ERR)

Попытка возведения отрицательного числа в дробную степень. Предупреждение — программа продолжает выполняться, но значение выражения равно нулю.

18. ? EXPRESSION TOO COMPLEX (?ETC)

Слишком сложное выражение в строке.

19. ? FILE ALREADY EXIST (?FAE)

Попытка создания уже созданного файла.

20. ? FILE NOT FOUND (?FNF)

Файл с указанным именем отсутствует.

21. ? FILE PRIVILEGE VIOLATION (?FPV)

Включение управления над ограниченными файлами.

22. ? FLOATING OVERFLOW (?FOV)

Полученное значение больше, чем $10^{38}$. Предупреждение — программа продолжает выполняться, но значение берется равным нулю.

23. ? FLOATING UNDERFLOW (?FUN)

Полученное значение меньше чем $10^{-38}$. Предупреждение — программа продолжает выполняться, но значение берется равным нулю.

24. ? FOR WITHOUT NEXT (?FWN)

Отсутствует оператор NEXT, закрывающий цикл, начатый оператором FOR.

25. ? FUNCTION ALREADY DEFINED (?FAD)

Повторное определение функции.

26. ? ILLEGAL CHANNEL NUMBER (?ICN)

Недопустимый номер объявленного канала для обмена данными с файлом.

27. ? ILLEGAL DEF (?IDF)

Ошибка в операторе DEF.

28. ? ILLEGAL DIM (?IDM)

Ошибка в операторе DIM или COMMON.

29. ? ILLEGAL FILE LENGTH (?IFL)

Длина файла, заданного в операторе OPEN, имеет недопустимое значение.

30. ? ILLEGAL FILE SPECIFICATION (?IFS)

Недопустимая спецификация файла.

31. ? ILLEGAL IN IMMEDIATE MODE (?IIM)

Неправильное использование оператора INPUT.

32. ? ILLEGAL I/O DIRECTION (?IID)

Неправильное использование объявленного файла: попытка ввода из файла для записи данных, или, наоборот, попытка вывода в файл для считывания данных.

33. ? ILLEGAL RECORD SIZE (?IRS)

Неправильно указана длина записи в операторе OPEN.

34. ? INCONSISTENT NUMBER OF SUBSCRIPTS (?INS)

Неверное обращение к массиву.

35. ? INPUT STRING ERROR (?ISE)

Нет закрывающих кавычек при вводе значения символьной переменной оператором INPUT. Предупреждение — программа продолжает выполняться дальше.

36. ? INTEGER OVERFLOW (?IOV)

Целая переменная имеет значение или больше, чем +32767, или меньше, чем —32768.

37. ? LINE TOO LONG (?LTL)

Строка программы слишком велика, т. е. количество символов в строке программы превышает 255 символов.

38. ? LINE TOO LONG TO TRANSLATE (?TLT)

Строка программы слишком длинна для транслятора.

39. ? NEGATIVE SQUARE ROOT (?NGS)

Попытка извлечения квадратного корня из отрицательного числа. Предупреждение — программа продолжает выполняться, но значение функции равно нулю.

40. ? NESTED FOR STATEMENT WITH SAME CONTROL VARIABLE (?FSV)

Использование одной и той же переменной цикла во вложенных циклах.

41. ? NEXT WITHOUT FOR (?NWF)

Оператор NEXT не имеет соответствующего оператора FOR.

42. ? NOT A VALID DEVICE (?NVD)

В спецификации файла содержится недопустимое имя устройства.

43. ? NOT ENOUGH ROOM (?NER)

Не хватает памяти для размещения файла.

44. ? NUMBER AND STRING (?NSM)

В одном выражении используются числовые и символьные переменные.

45. ? OUT OF DATA (?OOD)

Выход за пределы списка данных, объявленных в операторах DATA.

46. ? PRINT USING ERROR (?PRU)

Недопустимая спецификация формата в операторе PRINT USING.

47. ? PROGRAMM TOO BIG (?PTB)

Программа не помещается в оперативную память.

48. ? RETURN WITHOUT GOSUB (?RWG)

Оператор RETURN появляется в программе раньше соответствующего оператора GOSUB.

49. ? STRING STORAGE OVERFLOW (?SSO)

Не хватает памяти для символьных переменных.

50. ? STRING TOO LONG (?STL)

Символьная переменная содержит более 255 символов.

51. ? SUBSCRIPT OUT OF BOUNDS (?SOB)

Значение индекса элемента массива (переменной с индексом) отрицательно или больше допустимого.

52. ? SUBSTITUTE ERROR (?SUB)

Ошибка в записи команды SUB.

53. ? SYNTAX ERROR (?SYN)

Синтаксическая ошибка (т. е. ошибка в записи оператора).

54. ? TOO MANY GOSUB (?TMG)

Количество подпрограмм в программе превышает допустимое.

55. ? TOO MANY ITEMS IN COMMON (?TIC)

В операторе COMMON число переменных более 255.

56. ? UNDEFINED FUNCTION (?UFN)

Функция пользователя не определена.

57. ? UNDEFINED LINE NUMBER (?ULN)

Ссылка на несуществующий номер строки.

58. ? USE REPLACE (?RPL)

Попытка записать на ВЗУ файл под именем, уже имеющимся у файла на ВЗУ. Запись игнорируется.

59. ? VIRTUAL ARRAY CHANNEL ALREADY IN USE (?VCU)

Виртуальный массив, описанный в операторе DIM, уже был определен ранее.

Приложение 6

## КОМАНДЫ БЕЙСИКА

| Команда | Действие |
|---|---|
| LIST | Выводит всю программу или ее последние строки, если программа не умещается на экране |
| LIST—N | Выводит строки до N-й включительно |
| LIST N— | Выводит строки, начиная с N-й |
| LIST $N_1$—$N_2$ | Выводит строки с $N_1$-й до $N_2$-й |
| LIST N | Выводит строку N |
| SCR | Очищает оперативную память |
| NEW *имя* | Очищает оперативную память и присваивает программе имя |
| RENAME *имя* | Программе, находящейся в оперативной памяти, присваивает новое имя |
| DEL—N | Стирает строки программы до N-й включительно |
| DEL N— | Стирает строки программы от N-й до конца |
| DEL $N_1$—$N_2$ | Стирает строки с $N_1$-й до $N_2$-й |
| DEL $N_1$, $N_2$, ... | Стирает строки $N_1$, $N_2$, ... |
| RESEQ | Перенумеровывает строки, начиная с 10, с шагом 10 |
| RUN (RUNNH) | Передает управление программе, находящейся в оперативной памяти (программа начинает выполняться) |
| GOTO N | Передает управление N-й строке программы |
| SAVE *имя* | Записывает программу на ВЗУ |
| SAVE LP: | Выводит программу на ПУ |
| UNSAVE *имя* | Стирает программу с ВЗУ |
| REPLACE *имя* | Записывает новую версию программы на ВЗУ, старая версия при этом уничтожается |
| OLD *имя* | Загружает в оперативную память программу с ВЗУ |
| RUN *имя* | Загружает в оперативную память программу с ВЗУ, после чего программа начинает выполняться |
| SUBN$\gamma\alpha_1\gamma\alpha_2\gamma$M | Заменяет в строке N последовательность символов $\alpha_1$ (M-е вхождение) на последовательность $\alpha_2$, $\gamma$ — разделитель |

## СПИСОК ЛИТЕРАТУРЫ


1. Вирт Н. Систематическое программирование. Введение.— М.: Мир, 1977.
2. Штернберг Л. Ф. Разработка и отладка программ.— М.: Радио и связь, 1984.
3. Мейер Б., Бодуэн К. Методы программирования.— М.: Мир.— 1982.— Т. 1, 2.
4. Кнут Д. Искусство программирования для ЭВМ.— М.: Мир.— 1976.— Т. 1, 2, 3.
5. Холл П. Вычислительные структуры. Введение в нечисленное программирование.— М.: Мир, 1978.
6. Гудман С., Хидетниеми С. Введение в разработку и анализ алгоритмов.— М.: Мир, 1981.
7. Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ.— М.: Мир, 1981.
8. Любимский Э. З., Мартынюк В. В., Трифонов Н. П. Программирование.— М.: Наука, 1980.
9. Нивергельт Ю., Фаррар Дж., Рейнгольд Э. Машинный подход к решению математических задач.— М.: Мир, 1977.
10. Уорт Т. Программирование на языке Бейсик.— М.: Машиностроение, 1981.
11. Бауэр Ф., Гооз Г. Информатика.— М.: Мир, 1976.
12. Тихонов А. Н., Костомаров Д. П. Вводные лекции по прикладной математике.— М.: Наука, 1981.
13. Уэзерелл Ч. Этюды для программистов.— М.: Мир, 1982.
14. ГОСТ 19.002-80. ЕСПД Схемы алгоритмов и программ. Правила выполнения.
15. ГОСТ 19.003-80. ЕСПД Схемы алгоритмов и программ. Обозначения условные графические.
16. Дал У., Дейкстра Э., Хоор К. Структурное программирование.— М.: Мир, 1975.
17. Йодан Э. Структурное программирование и конструирование программ.— М.: Мир, 1979.
18. Хьюз Дж., Мичтом Дж. Структурный подход к программированию.— М.: Мир, 1980.
19. Турский В. Методология программирования.— М.: Мир, 1981.
20. Вирт Н. Алгоритмы + структуры данных = программы.— М.: Мир, 1985.
21. Касьянов В. Н., Сабельфельд В. К. Сборник заданий по практикуму на ЭВМ.— М.: Наука, 1986.
22. Бахвалов Н. С., Бетелин А. Б., Бетелин В. Б. и др. Практикум по программированию.— М.: Изд-во МГУ, 1986.
23. Трейстер Р. Персональный компьютер фирмы ИБМ.— М.: Мир, 1986.
24. Калиткин Н. Н. Численные методы.— М.: Наука, 1978.

1 р. 20 к.