С. М. Кашаев, Л. В. Шерстнева

БЕЙСИК
для школьников
Подготовка к ЕГЭ

Популярный язык программирования

Свободно распространяемая среда программирования

Разработка алгоритмов и реализация программ

Задания ЕГЭ по программированию

ИНФОРМАТИКА И ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

С. М. Кашаев
Л. В. Шерстнева

БЕЙСИК
для школьников
Подготовка к ЕГЭ

Санкт-Петербург
«БХВ-Петербург»
2012

УДК 681.3.068+800.92Basic(075.3)
ББК 32.973.26-018.1я721
К31

Кашаев, С. М.

К31 Бейсик для школьников. Подготовка к ЕГЭ / С. М. Кашаев, Л. В. Шерстнева. — СПб.: БХВ-Петербург, 2012. — 272 с.: ил. — (ИиИКТ)

ISBN 978-5-9775-0870-4

Книга включает все темы, выносимые на Единый государственный экзамен по информатике, касающиеся программирования, на примере языка программирования Бейсик: синтаксис языка, основные операторы, алгоритмические структуры, одномерные и двумерные массивы, строки и записи, подпрограммы и функции, математические вычисления. В каждой главе приводятся необходимые теоретические сведения и типовые задания с подробными пояснениями; по темам, которые есть в заданиях ЕГЭ, разобраны задания прошлых лет. Значительная часть книги посвящена обучению разработке алгоритмов и реализации их в виде программ.

Книга предназначена для подготовки к сдаче ЕГЭ, а также для использования в учебном процессе учащимися и преподавателями школ, колледжей и техникумов.

На сайте издательства приведены примеры программ из книги.

Для общеобразовательных учреждений

УДК 681.3.068+800.92Basic(075.3)
ББК 32.973.26-018.1я721

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Елена Васильева*
Редактор	*Юрий Рожко*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 31.05.12.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 21,93.
Тираж 1300 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-0870-4

Оглавление

Введение

Основным практическим результатом школьного курса информатики является формирование навыков программирования на одном языке высокого уровня — Бейсик, Паскаль, Си. Анализ учебного процесса показывает, что это составляет наиболее трудную часть курса информатики. Об этом говорят и результаты Единого государственного экзамена (ЕГЭ) по информатике. Так, статистика результатов за последние годы относит дисциплину "Информатика и информационно-коммуникационные технологии" к категории достаточно трудных для учащихся. Фактически эта трудность заложена в алгоритмизации задач и программировании. От учащихся в экзаменационных билетах требуется анализ, разработка и последующее программирование алгоритмов. Эти вопросы занимают большое место в Едином государственном экзамене и формируют категорию наиболее сложных заданий. Задача нашей книги сводится к последовательному и поэтапному формированию навыков программирования, а также к подготовке читателей к решению заданий ЕГЭ по данной теме.

В настоящее время используются различные языки программирования. При этом в школьной программе традиционными являются Бейсик и Паскаль. В представленной книге сделан выбор в пользу языка Бейсик, который можно считать наиболее популярным в школьной программе (относится к категории базовых языков программирования для школьников).

В настоящее время существует несколько программных продуктов, которые предназначены для написания и выполнения программ на Бейсике. Традиционно на протяжении последних лет наиболее популярной является среда Visual Basic компании Microsoft. Последняя версия данной среды Microsoft Visual Basic 2010 является удобной как для начинающих программистов, так и для профессиональных разработчиков программного обеспечения. Она и будет использоваться в данной книге для рассмотрения всех примеров и заданий на языке Бейсик. Важным моментом является и то, что Microsoft Visual Basic 2010 можно свободно установить с сайта компании Microsoft.

Книга построена таким образом, чтобы учащиеся школ (а также техникумов и колледжей) смогли самостоятельно рассмотреть как необходимую теоретическую информацию, так и на разнообразных примерах получить навыки практического про-

граммирования. Здесь рассмотрено большое количество заданий ЕГЭ по дисциплине "Информатика и информационно-коммуникационные технологии". И если вы собираетесь сдавать ЕГЭ по этой дисциплине, то мы поможем познакомиться с типовыми задачами, которые вас ожидают.

В целом книга состоит из восьми глав, содержание которых охватывает основные разделы программирования на языке Бейсик. Можно сказать, что главы книги включают все темы, выносимые на Единый государственный экзамен, касающиеся практического программирования. Структура каждой главы построена по схожему сценарию. Так, вместе с необходимыми теоретическими сведениями, которые излагаются в понятном для учащихся стиле, приводятся типовые примеры программных разработок. В большинстве глав (1—4, 7, 8) выполнен разбор заданий, которые *встречались в билетах Единого государственного экзамена в прошлые годы*.

Краткое содержание книги

В *главе 1* читатели познакомятся с основными операторами языка Бейсик и технологией выполнения программ в среде Visual Basic 2010. Мы разберем типы данных, организацию ввода и вывода информации.

В любой, даже несложной, программе используются операторы циклов и проверки условий. Циклы позволяют организовать повторяющиеся вычисления. Что касается условий, то в зависимости от выполнения либо невыполнения условия могут выполняться (или не выполняться) определенные фрагменты программы. Рассмотрению циклов и условий посвящена *глава 2* книги.

В *главе 3* подробно разбирается работа с одномерными массивами, которые являются одной из наиболее используемых на практике структур данных. Рассматриваются задачи поиска элементов, вычисления суммы и среднего значения в массиве.

Двумерные массивы встречаются практически в каждом билете Единого государственного экзамена, и работе с такими структурами посвящена *глава 4*. Аналогично содержанию третьей главы мы разберем задачи поиска и суммирования элементов по условию.

В *главе 5* рассматриваются строки, которые используются при работе с текстовой информацией.

Из *главы 6* вы узнаете, каким образом можно разделить программу на отдельные блоки — пользовательские процедуры и функции. Практически все программные разработки строятся в виде совокупности подобных блоков.

Математические вычисления составляют доминирующую составляющую программных разработок. И в *главе 7* мы разберем примеры решений уравнений и неравенств, численное интегрирование и метод Монте-Карло.

Глава 8 практически целиком построена на рассмотрении примеров заданий Единого государственного экзамена. Все эти примеры отличаются достаточной сложностью и связаны с различными алгоритмами обработки данных.

Электронный архив примеров программ, приведенных в книге, выложен на FTP-сервер издательства "БХВ-Петербург" по адресу: **ftp://85.249.45.166/9785977508704.zip**. Ссылка доступна и со страницы книги на сайте издательства **www.bhv.ru**. Описание папок электронного архива приведено в *приложении*.

От авторов книги

Нам бы хотелось выразить благодарность всем читателям, которые познакомились с нашей книгой. Что касается методики изложения информации, то она связана с работой на курсах по подготовке к Единому государственному экзамену в Нижегородском государственном техническом университете. Большую помощь в работе нам оказал заместитель директора Института дистанционного обучения Воронков Юрий Васильевич, и ему мы очень признательны.

Необходимо также упомянуть издательство "БХВ-Петербург" за сотрудничество в плане подготовки к изданию наших книг на протяжении последних лет. Особенно хотим поблагодарить заместителей главного редактора издательства Людмилу Юрьевну Еремеевскую и Рыбакова Евгения Евгеньевича за поддержку, ценные советы и внимание.

Книга может быть использована школьниками в качестве вспомогательного материала по информатике в процессе учебных занятий, а также при подготовке к Единому государственному экзамену. Думаем, что книга может быть полезна и учителям информатики в плане организации занятий и подбора примеров. В заключение подчеркнем, что материал, изложенный в книге, предназначается для самостоятельного изучения технологии программирования на языке Бейсик в системе Visual Basic 2010.

Разумеется, в книге возможны некоторые неточности, за которые заранее приносим читателям свои извинения и о которых (если они будут обнаружены) просим присылать сообщения.

Связаться с авторами книги можно по адресу электронной почты **mail@bhv.ru** или с помощью сайта **www.bhv.ru** издательства "БХВ-Петербург".

ГЛАВА 1

Знакомство с языком Бейсик и средой Visual Basic 2010

Из языков программирования среди учащихся и педагогов школ большой популярностью пользуется Бейсик. При этом в качестве среды программирования в последнее время чаще всего используется система Visual Basic компании Microsoft, которая на протяжении последних пятнадцати лет является одним из наиболее удобных средств для обучения программированию в учебных заведениях. Нашей задачей в данной книге является изучение языка Бейсик на практических задачах, которые выносятся на Единый государственный экзамен по дисциплине "Информатика и информационно-коммуникационные технологии". Все рассмотренные в книге теоретические разделы и примеры программирования алгоритмов приводятся в среде Visual Basic 2010.

Сам язык Бейсик был разработан достаточно давно и на заре компьютерной техники использовался исключительно в качестве средства для изучения основ программирования в школе. Качественное изменение Бейсика произошло при выпуске компанией Microsoft в 90-х годах прошлого века своего программного продукта Visual Basic 3.0. Начиная с этого момента, программирование на Бейсике становится вполне профессиональным, и среда Visual Basic позволила разработать многочисленное количество сложных прикладных проектов, функционирующих на платформе Windows. С одной стороны, простота и удобство, а с другой — разнообразные профессиональные ресурсы для разработки приложений сделали Visual Basic необыкновенно популярным как в учебной среде, так и среди профессионалов программирования.

Основой функциональности системы Visual Basic является возможность создания и выполнения программ на языке Бейсик. В целом Visual Basic представляет собой технологическое средство для разработки программ, каждая из которых реализует тот или иной алгоритм. Понятие алгоритма занимает ключевое место в процессе программирования и его стоит пояснить подробнее.

Алгоритм представляет собой строгую систему правил, определяющую последовательность действий для решения конкретной задачи. Следуя такой системе, в разных ситуациях нужно действовать совершенно одинаково (по единой схеме). Можно сказать, что алгоритмом является такая последовательность арифметических и

логических действий, которая позволяет решить поставленную задачу за конечное число шагов. В качестве решаемой задачи может быть, например, нахождение решения уравнения. *Этапы разработки* вычислительной программы выглядят следующим образом:

- *Постановка задачи*, которая выполняется специалистом в предметной области (математика, физика, строительство и т. д.). На этом этапе определяется общий подход к решению задачи. Среди задач, рассматриваемых в книге, мы встретимся с поиском максимального значения в массиве данных, сортировкой чисел, расчетом средних показателей и т. д.
- *Анализ задачи и моделирование*, которые приводят к построению (или выбору) математической модели. Этот этап очень важен, т. к. в ряде случаев анализ показывает, что поставленную задачу можно решить аналитическими методами.
- *Построение алгоритма* решения задачи, выполняемое на основании математической модели. В большинстве ситуаций существуют несколько способов построения работающего алгоритма. От разработчика всегда требуется найти оптимальную систему правил для решения поставленной задачи.
- *Реализация алгоритма в виде работающей программы* на одном из языков программирования. Этот этап часто называют *кодированием* — записью алгоритма в виде набора синтаксических конструкций выбранного языка программирования.
- *Отладка* и *тестирование* программы. Отладка заключается в устранении программных ошибок. Для поиска ошибок разработчик должен предложить набор тестов, представляющих собой контрольные примеры с характерными параметрами, для которых решение задачи известно. Тестирование позволяет ответить на вопрос — является ли разработанная программа решением данной задачи.

Описанная последовательность шагов достаточно понятна, но ее осуществление зависит от сложности поставленной задачи (для простых задач реализация перечисленных шагов достаточно очевидна, а для сложных ситуаций часто требуется длительное время и соответственно большие усилия). Фактически все программные разработки, которые вам известны и с которыми вы познакомитесь в этой книге, также вписываются в данную схему. Мы будем выполнять перечисленные этапы разработки программ, начиная с простых задач. Затем после приобретения определенного опыта в последующих главах разберем разделы, которые являются ключевыми, как для школьного курса информатики, так и для ЕГЭ по дисциплине "Информатика и информационно-коммуникационные технологии".

Запуск системы Visual Basic 2010

Система Visual Basic 2010 является приложением Windows, и после ее установки на компьютере для того, чтобы начать работу, следует обратиться к меню **Пуск**, выбрать **Программы** и в нем Microsoft Visual Basic 2010. Если же вы не располагаете данным программным продуктом, то скачать его в ознакомительных целях можно непосредственно с сайта компании Microsoft.

После запуска приложения Visual Basic 2010 на экране отображается стартовое окно данной программы (рис. 1.1). В левой части этого окна присутствуют два раздела, которые мы будем активно использовать:

- **New Project** — для создания нового проекта (каждый проект будет содержать одну из рассматриваемых далее программ);
- **Open Project** — для открытия существующего проекта (для того, чтобы вернуться к предыдущей разработке).

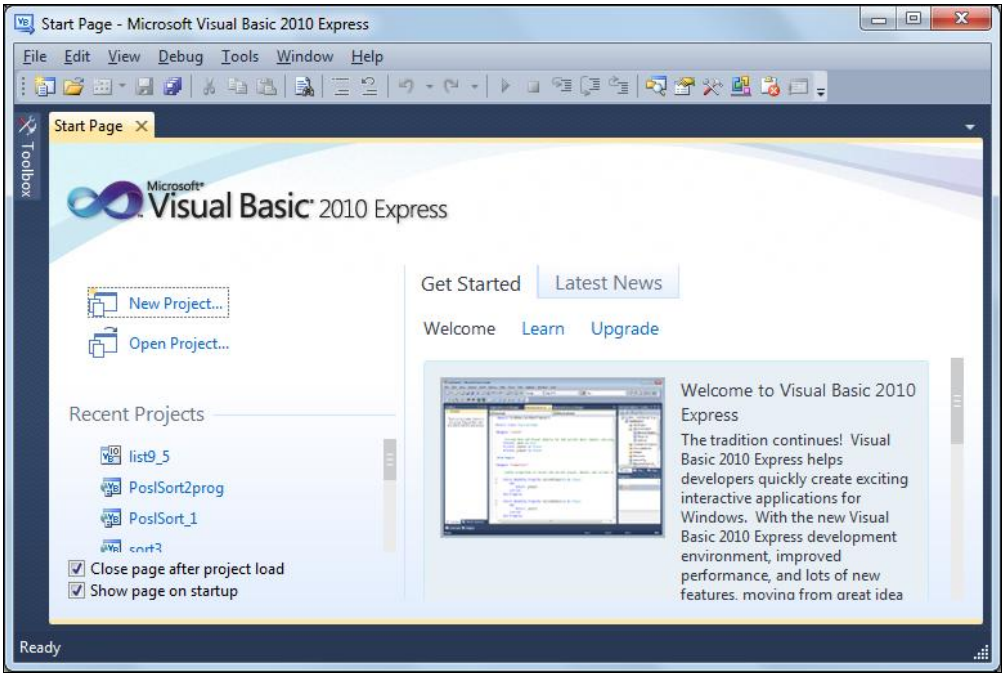

Рис. 1.1. Стартовое окно приложения Visual Basic 2010

Ниже этих разделов располагается перечень проектов, с которыми данный пользователь недавно работал, и если вы захотите вернуться к одной из своих предыдущих программ, то проще всего найти ее в этом списке. В данном случае мы только начинаем работу с Visual Basic 2010 и, поэтому, следует выбрать вариант создания нового проекта — щелкнуть по кнопке **New Project**.

На рис. 1.2 показано следующее окно, которое открывается перед нами — здесь пользователем осуществляется выбор одного из возможных типов создаваемого проекта. Дело в том, что Visual Basic 2010 предлагает создание различных вариантов программных разработок. Эти варианты подробно рассмотрены в ряде изданий [1, 2]. Однако выполнение заданий Единого государственного экзамена предусматривает использование только одного типа приложения — Console Application (консольное приложение). В связи с этим в данной книге мы и ограничимся рассмотрением разработок именно этого типа.

Таким образом, в окне на рис. 1.2 в качестве дальнейших практических действий выберем вариант **Console Application** (его же следует использовать при рассмотрении всех приводимых далее примеров).

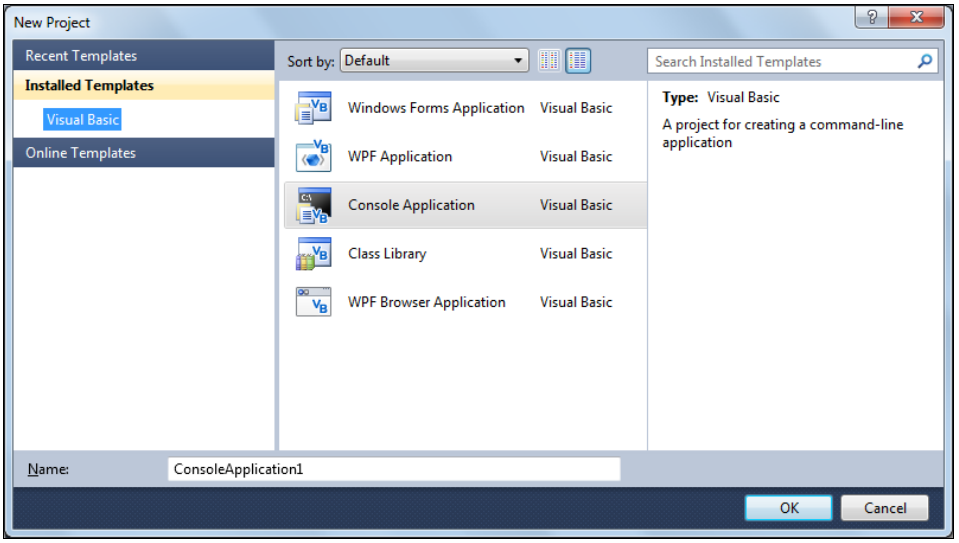

Рис. 1.2. Окно для выбора типа создаваемого проекта

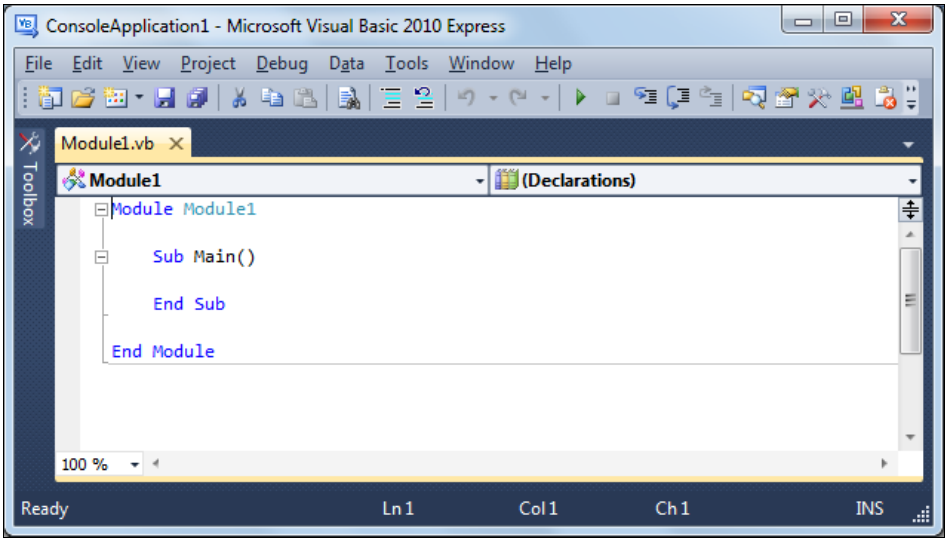

Рис. 1.3. Окно для написания кода программы

В результате перед нами открывается следующее окно (рис. 1.3), которое представляет собой редактор для написания программного кода. В этом окне мы и будем разрабатывать тексты программ на языке Бейсик. На рис. 1.3 показаны ключевые слова среды Visual Basic 2010:

- `Module` — начало программного модуля;
- `Sub` — начало подпрограммы;
- `End` — завершение той или иной программной конструкции.

Фактически содержание окна, показанного на рис. 1.3, представляет собой заготовку для написания нами программного кода — эта часть программного кода проекта уже создана системой автоматически. От нас требуется в данную заготовку вставлять свои строки программного кода.

Таким образом, в данный момент мы подошли к этапу написания программы и далее в следующем разделе рассмотрим первую (разумеется, очень простую) программу, которую затем выполним в среде Visual Basic 2010.

Программа вывода сообщения на экран

Наша задача заключается в создании программы, которая выводит на экран сообщение: "Пример первой программы". Подобные программы традиционно рассматриваются в качестве первых при изучении той или иной системы программирования. Фактически в этом случае мы получаем представление о технологическом процессе разработки (начиная от создания текста на языке Бейсик до выполнения разработанной программы). Текст программы, который сейчас нам потребуется набрать в окне редактора, представлен в рис. 1.4. Здесь от нас в уже сформированную заготовку требуется ввести только одну строку:

```
Console.Write("Пример первой программы")
```

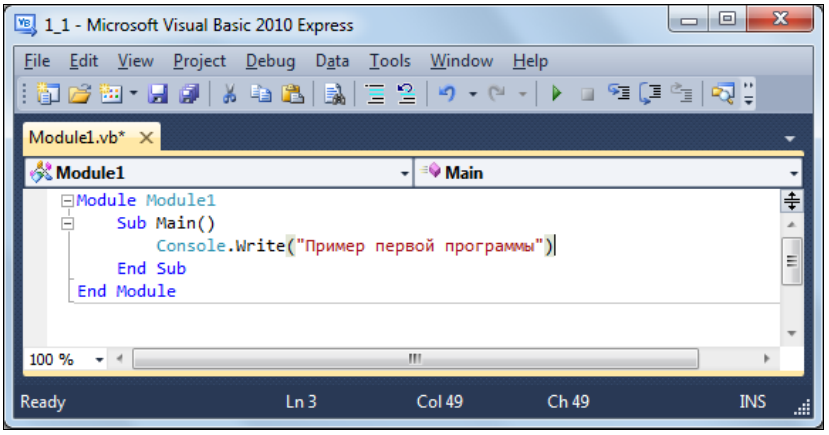

Рис. 1.4. Окно редактора кода с текстом программы вывода сообщения на экран

Перед тем как эту строку прокомментировать, сохраним нашу программу на диске, а затем запустим ее на выполнение.

Для сохранения проекта следует в меню **File** выбрать пункт **Save All**. В результате на экране мы увидим новое окно (рис. 1.5), в котором от нас требуется выбрать название проекта и указать место его расположения на диске компьютера (или в сети).

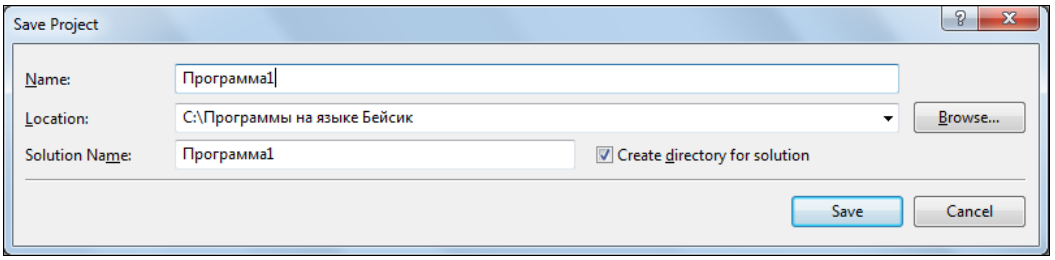

Рис. 1.5. Окно сохранения программного проекта

Теперь следующая наша задача заключается в выполнении данного проекта на компьютере. Для этого можно воспользоваться пунктом **Start Debugging** из меню **Debug**. Другой вариант запуска проекта на выполнение состоит в использовании одноименной кнопки  на панели инструментов. В результате на экране всего на одно мгновение появляется новое окно (рис. 1.6), в котором отображается сообщение "`Пример первой программы`". После этого мы опять увидим на экране окно редактора программного кода с текстом нашей программы (см. рис. 1.4). Таким образом, в реализованном варианте система быстро выполнила программу (вывела сообщение на экран) и вернулась к исходному проекту в окне редактора.

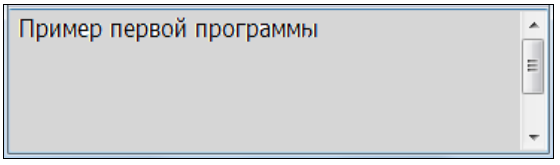

Рис. 1.6. Вывод информации программой, представленной в листинге 1.1

Для того чтобы задержать на экране окно вывода с текстом сообщения, воспользуемся ожиданием ввода информации с клавиатуры в конце программы. В этом случае программа будет ожидать ввода с клавиатуры, и когда это произойдет, то она завершит свою работу. В связи с этим изменим текст нашей программы (см. рис. 1.4) на вариант, представленный в листинге 1.1. После запуска программы в таком варианте на экране будет сформировано выводимое сообщение, которое пропадет только после нажатия клавиши <Enter> на клавиатуре.

Листинг 1.1. Программа вывода сообщения на экран

```
Module Module1
  Sub Main()
    Console.Write("Пример первой программы")
    Console.Read()
  End Sub
End Module
```

Разберем содержание листинга 1.1. Так, первая строка говорит о начале программного модуля (`Module`). Далее располагается процедура `Main`, которую ограничивают

строки `Sub Main()` и `End Sub`. Фактически эти строки представляют собой заготовку (каркас) программы. Эта заготовка создается системой автоматически. Наша задача заключается в написании содержательной части программы, которая в данном случае связана с использованием объекта `Console`. При этом мы использовали метод `Write` этого объекта для вывода информации. Текст выводимого сообщения передается в качестве параметра метода. Упрощенно можно сказать, что здесь мы воспользовались функцией вывода информации на экран. Далее в следующей строке программы используется метод `Read` того же объекта `Console` для ожидания ввода данных с клавиатуры. При этом нас интересует только факт ввода, а само содержание в данном случае интереса не представляет.

Теперь скорректируем текст программы, который сейчас будет выглядеть так, как показано в листинге 1.2. Здесь мы организовали вывод того же сообщения на экран, но только реализовали это с помощью двух программных строк.

Листинг 1.2. Вывод сообщения на экран (вариант 2)

```
Module Module1
  Sub Main()
     Console.Write("Пример ")
     Console.Write("первой программы")
     Console.Read()
  End Sub
End Module
```

Если же мы собираемся организовать вывод нашего сообщения на экране в две строки, то следует воспользоваться вариантом, представленным в листинге 1.3. Результат работы программы показан на рис. 1.7. В тексте программы мы воспользовались методом `WriteLine`, который после вывода информации осуществляет перевод строки.

Листинг 1.3. Вывод сообщения на экран (вариант 3)

```
Module Module1
  Sub Main()
     Console.WriteLine("Пример")
     Console.Write("первой программы")
     Console.Read()
  End Sub
End Module
```

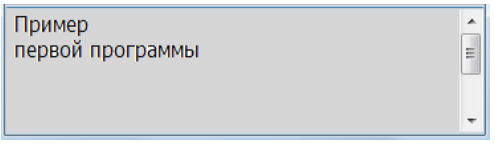

Рис. 1.7. Вывод информации программой, представленной в листинге 1.3

Вычисления в программе

Как правило, большинство программ связано с организацией вычислительных действий. В качестве простого примера на данную тему рассмотрим реализацию программы, которая должна обеспечить возведение в квадрат значения целого числа, вводимого с клавиатуры. После этого результат вычисления необходимо вывести на экран. Подобная цель достаточно понятна, а набор правил для ее реализации выглядит так:

- ввод целого числа с клавиатуры;
- возведение этого числа в квадрат;
- вывод полученного результата на экран.

После формулировки задачи и определенности с алгоритмом действий перейдем к написанию программного кода. Для этого в текстовом редакторе среды Visual Basic 2010 следует набрать текст программы, представленной в листинге 1.4, которая реализует поставленную задачу.

Листинг 1.4. Вычисление квадрата числа, вводимого с клавиатуры

```
Module Module1
  Sub Main()
      Dim N As Integer
      N = Console.ReadLine()
      N = N * N
      Console.WriteLine("Результат возведения в квадрат")
      Console.Write(N)
      Console.Read()
    End Sub
End Module
```

На рис. 1.8 представлено окно вывода, демонстрирующее результат работы программы после ввода целого числа с клавиатуры.

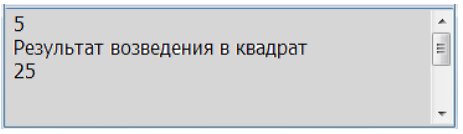

Рис. 1.8. Вывод информации в программе, представленной в листинге 1.4

В листинге 1.4 для возведения в квадрат мы воспользовались перемножением исходного значения на себя. Однако можно также воспользоваться непосредственно операцией возведения в степень:

```
N = N ^2
```

Новое ключевое слово языка Бейсик, которое нам встретилось в листинге 1.4, это — `Dim`. Оно означает объявление переменной (или переменных) в программе.

Переменная представляет собой зарезервированное место в оперативной памяти компьютера, предназначенное для хранения данных в программе.

> ***ПРИМЕЧАНИЕ***
>
> Дело в том, что данные, с которыми мы работаем, необходимо где-то хранить. Для этого предназначается основная (оперативная) память компьютера. Такая память состоит из набора однотипных *ячеек*, при этом каждая ячейка имеет свой номер, который называется *адресом*.

Чтобы не работать с адресами ячеек напрямую (не запоминать длинные числовые адреса ячеек) предусмотрена возможность присвоения ячейкам символического имени *(идентификатора)*. В нашем случае мы создали переменную `N`, и для нее указали тип `Integer`, который определяет хранение целочисленного данного. Для переменных указанного типа в памяти отводится 32 бита (32 двоичных разряда или 4 байта). При этом числа, хранящиеся в переменных типа `Integer`, являются знаковыми, и их диапазон составляет от -2^{31} до $2^{31}-1$.

> ***ПРИМЕЧАНИЕ***
>
> Поскольку любые данные в памяти компьютера хранятся в двоичной системе счисления, то кроме имени переменной следует указать и *тип*, определяющий *диапазон значений*, принимаемых переменной, и *способ ее обработки* вычислительной системой.

Без дополнительной информации о *типе* данных, хранящихся в некоторой ячейке памяти, компьютеру было бы невозможно решить, что именно представляют собой эти данные.

Имена (идентификаторы) переменных следует выбирать самостоятельно, и в рассматриваемой программе мы выбрали в качестве имени единственной переменной идентификатор `N`. Любые имена переменных в языке Бейсик строятся по следующим правилам:

- имена могут включать буквы, цифры и знак подчеркивания;
- имя состоит из одного слова, а если требуется пробел в имени, то он заменяется на символ подчеркивания (например, `St_1` будет правильным вариантом для имени, а `St 1` приведет к ошибке на этапе компиляции текста программы);
- имя всегда начинается с буквы либо со знака подчеркивания;
- между двумя именами должен располагаться, как минимум, один пробел;
- имена переменных не могут совпадать с зарезервированными в языке *ключевыми (служебными) словами* (например, ни одну переменную в программе нельзя назвать `Sub`).

Далее в тексте данной программы нам встретился *оператор присваивания* (обозначается знаком равенства). Это наиболее часто используемый оператор, который работает следующим образом: сначала вычисляется выражение, стоящее *справа* от оператора присваивания, а затем результат записывается в переменную, стоящую *слева* от данного знака. Например, после выполнения оператора

```
K=K+2
```

текущее значение переменной `K` увеличится на `2`.

Важно отметить, что справа от оператора присваивания может располагаться любое число либо выражение, а слева обязательно должно находиться имя переменной (мы должны указать системе — куда следует записать результат вычисления или просто какое-либо данное).

Продолжим рассмотрение содержания листинга 1.4. После описания переменной целого типа располагается строка, где с клавиатуры вводится значение переменной `N`. Для этого мы воспользовались методом `ReadLine`. Обратим внимание на то, что в данном случае необходимо ввести с клавиатуры строку, содержащую целое число.

После ввода значения переменной `N` осуществляется возведение его в квадрат. Для этого мы воспользовались знаком, обозначающим умножение (`*`). В данном случае значение переменной `N` умножается само на себя. Фактически это и является возведением исходного значения в квадрат. Результат операции умножения заносится обратно в переменную `N` (в ячейку памяти, которую мы отвели для переменной).

Важно подчеркнуть, что введенное с клавиатуры значение должно представлять собой целое число (тип `Integer`), а в случае ввода набора букв вместо целого числа при выполнении программы произойдет ошибка. Это связано с тем, что операция умножения для строк и символов отсутствует.

После выполнения умножения в программе организуется вывод информации на экран уже знакомым нам способом.

Типы данных

Для решения любой задачи с использованием программирования осуществляются действия над определенными данными. При этом данные могут иметь разные типы. *Тип данных* определяет множество значений, которые могут принимать объекты программы (константы и переменные). Также тип данных определяет и совокупность действий, которые можно выполнять над данными определенного типа. Например, с числами можно выполнять большой набор действий (сложение, вычитание, умножение, деление и т. д.), а строки можно только складывать.

Тип данных очень важен при выделении памяти под переменные. Это связано с тем, что каждому типу данных соответствует определенное число байтов памяти компьютера.

Рассмотрим, что представляют собой стандартные типы данных в системе Visual Basic 2010. При организации вычислений программисты наиболее часто используют *целочисленные* типы данных, перечень которых представлен в табл. 1.1. Здесь указаны минимальные и максимальные значения для каждого типа данных, а также количество отводимых для них байтов в памяти компьютера. Действия над целыми числами указанных типов определены лишь тогда, когда исходные данные (операнды) и результат лежат в заданном интервале. В противном случае возникает *переполнение* [3].

Таблица 1.1. *Целочисленные типы данных в Visual Basic 2010*

Тип	Диапазон значений	Размер в байтах
SByte	От -128 до 127	1
UShort	От 0 до 65535	2
Short	От -32768 до 32767	2
Integer	От -2 147 483 648 до 2 147 483 647	4
UInteger	От 0 до 4 294 967 295	4
Long	От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807	8
ULong	От 0 до 18 446 744 073 709 551 615	8

Приведем пример описания нескольких переменных целочисленного типа:

```
Dim A1 As Integer
Dim A2 As Short
Dim A3 As Long
```

Как известно [3], числа в вычислительной системе представлены в двоичной системе счисления и на каждое число в оперативной памяти отводится определенное количество разрядов (битов). В этом плане прокомментируем используемые в Visual Basic 2010 целые числовые типы данных. Так, наименьшее число разрядов (8) имеет тип `SByte`. В этом случае минимальное число равно -128, а максимальное 127. Такой тип данных позволяет работать только с короткими знаковыми числами.

Для вычислений наиболее часто используется тип `Integer` (под данное отводится 32 двоичных разряда), в котором минимальное число равно -2^{31}, а самое большое положительное равняется $2^{31}-1$. Если требуется отразить бóльшие целые числа, то следует воспользоваться типом `Long`.

Не все числа, с которыми приходится работать, являются целыми. В большинстве вычислений используются дробные числа. С точки зрения математики таких чисел бесконечно много. Что же касается вычислительной системы, то их также много, но общее количество в этом случае конечно.

Дробные числа, кроме целой части, включают еще и дробную составляющую, отделяемую от целой разделителем.

Для чисел с дробной частью предназначены типы `Single` и `Double`, которые хранят числа с плавающей запятой. При этом в записи числа выделяются две компоненты: *мантисса* и *порядок*. В этом случае число представляется в виде мантиссы (как правило, представляет собой число в пределах от 1 до 10), умноженной на 10 в определенной степени (степень соответствует порядку). Приведем примеры нескольких чисел в таком формате:

- $4700 = 4{,}7 \cdot 10^3$, что в указанном формате выглядит 4,7000000E+03.
- $-25000 = -2{,}5 \cdot 10^4$ или в указанном формате -2,5000000E+04.
- $-0{,}0005 = -5 \cdot 10^{-4}$ или -5,0000000E-04.

Подчеркнем, что в системе Visual Basic 2010 при записи чисел основание 10 заменяется на букву E, после которой размещается значение порядка.

Переменные, объявленные как Decimal, содержат числа с фиксированной запятой. В отличие от чисел с плавающей запятой, числа данного типа не имеют множителя "10 в степени". Это позволяет избежать ошибок округления, которые могут возникнуть при обработке чисел с плавающей запятой. В связи с этим рекомендуется применять тип Decimal, когда вы производите сложные расчеты.

В табл. 1.2 приведены варианты дробных числовых типов в системе Visual Basic 2010.

Таблица 1.2. *Дробные числовые типы данных в Visual Basic 2010*

Тип	Диапазон значений	Размер в байтах и комментарий
Single	Отрицательные числа от -3,4E38 до -1,4E-45; Положительные числа от -1,4E-45 до 3,4E38.	4 Число с плавающей запятой
Decimal	От -79 228 162 514 264 337 593 543 950 335 до 79 228 162 514 264 337 593 543 950 335	16 Число с фиксированной запятой
Double	Отрицательные числа от -1,79E308 до -4,9E-324; Положительные числа от 4,9E-324 до 1,79E308.	8 Число с плавающей запятой двойной точности

Приведем пример использования числовых переменных дробного типа в программе. В листинге 1.5 приведен пример вычисления квадратного корня числа, имеющего тип Decimal. На рис. 1.9 показано окно с результатом вычислений — мы видим, что результат вычисления квадратного корня содержит большое количество цифр после разделителя целой и дробной части.

Листинг 1.5. Вычисление квадратного корня числа, вводимого с клавиатуры

```
Module Module1
   Sub Main()
      Dim X, Y As Decimal
      X = Console.ReadLine()
      Y = Math.Sqrt(X)
      Console.WriteLine("Результат вычисления квадратного корня")
      Console.Write(Y)
      Console.Read()
   End Sub
End Module
```

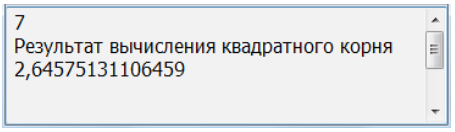

Рис. 1.9. Вывод информации в программе, приведенной в листинге 1.5

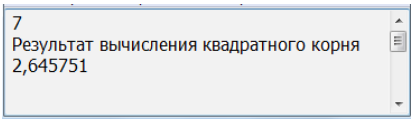

Рис. 1.10. Вывод информации в программе, приведенной в листинге 1.6

Теперь немного изменим программу, а именно укажем для переменных `X` и `Y` тип данных `Single`. В листинге 1.6 приведен ее текст. Соответствующий результат вычисления корня в программе показан на рис. 1.10. Видно, что корень из числа 7 будет содержать меньшее количество разрядов после запятой, чем в результате работы программы, приведенной в листинге 1.5.

Листинг 1.6. Вычисление квадратного корня числа (вариант 2)

```
Module Module1
    Sub Main()
        Dim X, Y As Single
        X = Console.ReadLine()
        Y = Math.Sqrt(X)
        Console.WriteLine("Результат")
        Console.Write(Y)
        Console.Read()
    End Sub
End Module
```

Можно также непосредственно задать в программе необходимое число с указанием мантиссы и порядка (листинг 1.7). Результат работы такой программы приведен на рис. 1.11.

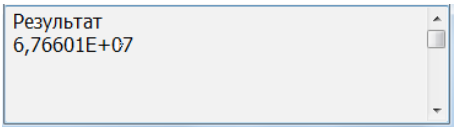

Рис. 1.11. Вывод информации в программе, приведенной в листинге 1.7

Листинг 1.7. Вычисление квадратного корня числа, указанного с помощью мантиссы и порядка

```
Module Module1
  Sub Main()
     Dim X, Y As Single
     X = 4.57789E+15
     Y = Math.Sqrt(X)
     Console.WriteLine("Результат")
```

```
    Console.Write(Y)
    Console.Read()
  End Sub
End Module
```

Для представления одиночных символов (букв, цифр и ряда других символов) предназначен *символьный* тип данных. Для каждой переменной данного типа в памяти отводится 2 байта (кодировка Unicode), а для описания используется ключевое слово `Char`. В листинге 1.8 приведен пример вывода двух символов на экран. При этом один из символов вводится с клавиатуры, а другой задается непосредственно в программе (при установлении значения символа используются двойные кавычки).

Листинг 1.8. Пример использования символьных переменных

```
Module Module1
  Sub Main()
    Dim X, Y As Char
    X = Console.ReadLine()
    Y = "Ю"
    Console.Write("Символ X-> ")
    Console.WriteLine(X)
    Console.Write("Символ Y-> ")
    Console.Write(Y)
    Console.Read()
  End Sub
End Module
```

Для более содержательной текстовой информации используется тип `String`. В этом случае под переменную отводится строка, которая может содержать до 2 млрд символов. В программе при инициализации строк также используются обрамляющие двойные кавычки. В листинге 1.9 приведен пример использования двух переменных строкового типа. При этом одна из строк вводится с клавиатуры, а другая задается непосредственно в программе. Результат работы такой программы приведен на рис. 1.12.

Листинг 1.9. Пример использования строковых переменных

```
Module Module1
  Sub Main()
    Dim X, Y As String
    Console.WriteLine("Введите свое имя")
    X = Console.ReadLine()
    Y = "Добрый день,"
    Console.Write(Y)
```

```
      Console.Write(X)
      Console.Write("!")
      Console.Read()
  End Sub
End Module
```

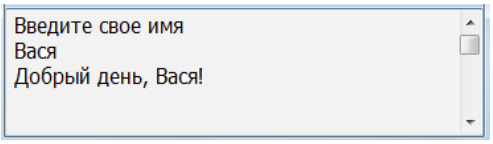

Рис. 1.12. Вывод информации в программе, приведенной в листинге 1.9

Кроме упомянутых типов данных, в Бейсике используется *логический* (другое название — *булевый тип*). Ключевое слово для указания данного типа — `Boolean`. Данные этого типа могут принимать только два возможных значения: `True` (истина) и `False` (ложь). Важно заметить, что при переводе числовых данных в логические значение 0 становится `False`, а остальные значения — `True`. Когда логические данные переводятся в числовые, то `False` становится 0, а `True` — 1. Приведем пример использования переменной данного типа (листинг 1.10) в программе. Здесь выполняется сравнение введенного с клавиатуры числа с цифрой 3. Если введенное число оказывается больше, то выводится значение `True`. В противном случае выводится `False`.

Листинг 1.10. Пример использования логических переменных

```
Module Module1
  Sub Main()
    Dim B As Boolean
    Dim X As Integer
    Console.WriteLine("Введите целое число")
    X=Console.ReadLine()
    B = X > 3
    Console.WriteLine("Высказывание о том, что данное число больше трех")
    Console.Write(B)
    Console.Read()
  End Sub
End Module
```

Переменные типа `Date` хранят значения даты и времени. Значение даты должно заключаться между символами # и быть в формате "месяц/день/год". В листинге 1.11 приведен пример использования переменной типа `Date` в программе. Здесь мы осуществляем инициализацию переменной `D` конкретным значением даты. После этого выполняется преобразование даты к строковому типу (для этого используется стандартная функция `Format`) и вывод значения даты на экран.

Листинг 1.11. Пример использования переменной типа `Date`

```
Module Module1
  Sub Main()
     Dim D As Date
     D = #12/31/2011#
     Console.WriteLine(Format(D, "dd.MM.yyyy"))
     Console.Read()
  End Sub
End Module
```

Объявления переменных

В системе Visual Basic 2011 можно использовать как явное, так и неявное объявление переменных. Явное объявление означает указание имени и типа переменной перед ее использованием. Это осуществляется как с помощью уже знакомого оператора `Dim`, так с помощью ряда других операторов (`Public`, `Static`, `Private`).

Переменная, объявленная при помощи оператора `Dim`, доступна из любого места программы в пределах области видимости, содержащей оператор `Dim`. Например, если переменная объявлена внутри модуля вне любой процедуры, то такая переменная доступна из любого места этого модуля. Если же переменная объявлена внутри процедуры, то она доступна только в пределах этой процедуры. Такая переменная называется *локальной*.

Чтобы определить доступность переменной более детально, применяются операторы `Private` и `Public`.

Использование оператора `Public` означает, что переменная имеет общий доступ, т. е. доступ без каких-либо ограничений. Переменная вида `Public` не может быть объявлена внутри процедуры.

Если переменная объявлена как `Static`, то она остается существовать в памяти и сохраняет свое последнее значение после завершения работы процедуры, в которой была объявлена. Переменная типа `Static` не может быть объявлена вне процедуры.

С помощью одного оператора можно объявлять несколько переменных, разделяя их запятыми. Примеры таких ситуаций мы уже упоминали ранее.

Часть "`As тип данных`" при объявлении переменной может отсутствовать. В этом случае Visual Basic назначает переменной тип значения, которое присваивается при объявлении. Если же тип данных не указан, а переменная не инициализируется никаким значением, то система назначит ей тип данных `Object` (в этом случае в переменной может храниться данное любого типа).

По умолчанию Visual Basic устанавливает режим явного объявления переменных. Для того чтобы это изменить, можно выполнить одно из следующих действий:

- указать в начале программного кода опцию `Option Explicit Off`;
- выделить в окне **Solution Explorer** (для этого следует в меню **View** выбрать пункт **Other Windows**) соответствующий проект и выбрать в его контекстном

меню пункт **Properties** (Свойства). На вкладке **Compile** списка **Option explicit** выбрать требуемое значение для компилятора.

Операции и выражения

Выражение определяет порядок выполнения действий над данными и состоит из операндов, круглых скобок и знаков операций (также в выражении могут присутствовать функции). Например, выражение может быть построено так:

```
X1+X2*(Y1+Y2)
```

Здесь мы использовали имена переменных (или констант), знаки сложения, умножения и круглые скобки для указания приоритетности вычислений. Разумеется, при вычислении выражения все объекты, которые составляют выражение, должны быть уже определены.

В Бейсике можно выделить несколько типов выражений:

- *арифметические*, которые предназначены для арифметических действий над числами;
- *логические* — для сравнения данных;
- *символьные* — для работы с текстом.

Можно выделить две категории операций:

- *унарные* (действие над одним операндом), которых немного;
- *бинарные* (действие над двумя операндами), которые составляют большинство.

В программных вычислениях наиболее широко представлены арифметические и логические операции, которые приведены в табл. 1.3. В этой таблице большинство операций являются бинарными, за исключением двух последних унарных операций.

Таблица 1.3. *Арифметические и логические операции в Visual Basic 2010*

Операция	Действие
+	Сложение
-	Вычитание
*	Умножение
/	Деление
\	Целочисленное деление
^	Возведение в степень
Mod	Остаток от деления
And	Логическое и арифметическое И
Or	Логическое и арифметическое ИЛИ

Таблица 1.3 *(окончание)*

Операция	Действие
`Xor`	Исключающее ИЛИ
`-`	Перемена знака (унарная операция)
`Not`	Логическое отрицание (унарная операция)

Кроме представленных арифметических и логических операций (см. табл. 1.3), в системе Visual Basic имеется множество стандартных процедур, функций и методов. Они позволяют выполнять тригонометрические вычисления, извлекать квадратный корень из числа и т. д. С некоторыми из них мы уже встречались, а о других еще поговорим далее в этой главе.

Некоторые из указанных в табл. 1.3 операций являются *побитовыми*, т. е. действие выполняется побитно (поразрядно) над каждым битом (разрядом) операнда (операндов).

Рассмотрим несколько наиболее интересных операций.

Арифметическая операция "И" (`And`) осуществляет логическое побитовое умножение операндов в соответствии со следующими правилами:

- `1 And 1 = 1`;
- `1 And 0 = 0`;
- `0 And 1 = 0`;
- `0 And 0 = 0`.

Здесь определены действия над каждой парой соответствующих битов операндов. Например, мы собираемся вычислить выражение с числами, указанными в десятичной системе счисления:

```
5 And 11.
```

В этом случае на уровне компьютерных вычислений это выглядит так (для определенности будем рассматривать восьмиразрядные числа):

```
0000 0101 And 0000 1011 = 0000 0001.
```

Здесь исходные числа (5 и 11) записаны в десятичной системе счисления, но в компьютере они хранятся в двоичной системе [3]. И над каждой парой соответствующих битов чисел производится операция логического умножения `And`. В результате вычисления получим следующий ответ:

```
5 And 11 = 1.
```

Листинг 1.12 содержит пример программы, демонстрирующей использование арифметической операции "И". В качестве результата выполнения на экране мы увидим цифру 2.

ПРИМЕЧАНИЕ

Проверим полученный результат вручную. Если перевести 23 в двоичную систему счисления, то результат будет таким — 0001 0111. Соответственно для числа 10 перевод в двоичную систему — 0000 1010. Из сопоставления битов видно, что результат арифметической операции "И" равен 0000 0010. Таким образом, сформированный программой ответ 2 в десятичной системе правильный.

Листинг 1.12. Пример вычисления с использованием арифметической операции "И"

```
Module Module1
    Sub Main()
        Dim A1, A2, A3 As Integer
        A1 = 23
        A2 = 10
        A3 = A1 And A2
        Console.Write(A3)
        Console.Read()
    End Sub
End Module
```

Арифметическая операция "ИЛИ" (`Or`) производит логическое побитовое сложение операндов в соответствии со следующими правилами:

- `1 Or 1 = 1`;
- `1 Or 0 = 1`;
- `0 Or 1 = 1`;
- `0 Or 0 = 0`.

Например, если мы собираемся вычислить выражение: `17 Or 3`, то на уровне компьютерных вычислений это выглядит так:

`0001 0001 Or 0000 0011 = 0001 0011`.

В результате получим:

`17 Or 3 = 19`.

Операция "Исключающее ИЛИ" (`Xor`) выполняет логическую операцию в соответствии со следующими правилами:

- `1 Xor 1 = 0`;
- `1 Xor 0 = 1`;
- `0 Xor 1 = 1`;
- `0 Xor 0 = 0`.

Например, если мы собираемся вычислить выражение: `15 Xor 3` над десятичными числами, то на уровне компьютерных вычислений это выглядит так:

`0000 1111 Xor 0000 0011 = 0000 1100`.

В результате вычисления получим `15 Xor 3 = 12`.

Операция арифметического отрицания (`Not`) выполняет побитовую инверсию (над одним операндом). Двоичные нули заменяются единицами, а единицы нулями.

Операции *отношения* выполняют сравнение двух операндов и определяют: истинно (`True`) выражение или ложно (`False`). Результат выражения с использованием таких операций имеет логический тип (о нем мы уже говорили). Определены следующие операции отношения:

- `<` — меньше;
- `>` — больше;
- `=` — равно;
- `>=` — больше или равно;
- `<=` — меньше или равно;
- `<>` — не равно.

Результат каждого логического выражения с использованием одной из перечисленных операций отношения может принимать одно из двух значений: `False` и `True`. Операции отношения определены над числовыми переменными, над символьными переменными и строками. Приведем несколько примеров использования операций отношения:

- `6 >= 3`, результат которой равен `True`;
- `2.25 > 7.9`, с результатом `False`;
- `"Pete" = "Pets"`, что дает результат `False`.

Примеры вычислений в программах

В этом разделе мы приведем ряд примеров программ, которые демонстрируют использование уже рассмотренных ранее типов данных при различных вычислительных действиях. В качестве первого примера продемонстрируем работу с целыми числами. В листинге 1.13 приведен пример программы, демонстрирующей действия, которые могут выполняться над целыми числами.

Листинг 1.13. Пример действий над целыми числами

```
Module Module1
  Sub Main()
     Dim N, M, W As Integer
     N = 29
     M = 6
     W = N * M
     Console.Write("29 умножить на 6 равно  ")
     Console.WriteLine(W)
     W = N \ M
     Console.Write("Результат целочисленного деления 29 на 6 равен ")
     Console.WriteLine(W)
```

```
      W = N Mod M
      Console.Write("Остаток от деления 29 на 6 равен  ")
      Console.WriteLine(W)
      Console.Read()
  End Sub
End Module
```

Приведенные здесь манипуляции над числами весьма несложные. Следует только обратить внимание на операцию `Mod` (вычисление остатка от деления). Так, после деления одного целого числа на другое получается целая часть результата и остаток, который и представляет собой результат данной операции. Например, остаток от деления 25 на 6 равен 1 (целая часть результата деления равна 4).

Рассмотрим теперь пример использования в вычислениях дробных чисел (листинг 1.14).

Листинг 1.14. Пример умножения дробных чисел

```
Module Module1
   Sub Main()
       Dim Z, W, X As Single
       X = 2.567
       Z = 6.78
       W = X * Z
       Console.Write("Результат умножения равен  ")
       Console.WriteLine(W)
       Console.Read()
   End Sub
End Module
```

В результате (17,40426) умножения данных чисел мы видим пять цифр после запятой. При выводе дробных чисел часто желательно ограничить число разрядов после разделителя целой и дробной части. Для этого следует воспользоваться функцией `Format`. В примере, приведенном в листинге 1.15, мы определили только две цифры после разделителя целой и дробной части:

```
Format(W, ".00").
```

Здесь второй параметр функции `Format` является спецификатором формата. После символа "точка" (определяет разделитель дробной и целой части) располагаются два символа "0". Этот символ говорит об обязательном включении цифры в данном разряде в результат форматирования.

После запуска данной программы мы увидим на экране число 17,40.

Листинг 1.15. Использование функции `Format` при выводе дробных чисел

```
Module Module1
    Sub Main()
        Dim Z, W, X As Single
```

```
        X = 2.567
        Z = 6.78
        W = X * Z
        Console.Write("Результат умножения равен  ")
        Console.WriteLine(Format(W, ".00"))
        Console.Read()
    End Sub
End Module
```

В качестве выводимых данных могут быть выражения, включающие стандартные функции. В листинге 1.16 приведен пример вычисления значения функции синус от аргумента (значение градусов), введенного с клавиатуры. Для преобразования значения аргумента из градусов в радианы мы воспользовались числом "Пи" (π), которое заложено в систему Visual Basic 2010 — `Math.PI`.

Листинг 1.16. Вычисление значения функции синус от аргумента в градусах

```
Module Module1
    Sub Main()
       Dim X, Y As Single
       Console.WriteLine("Введите значение аргумента в градусах")
       X = Console.ReadLine()
       Y = Math.Sin(Math.PI * X / 180)
       Console.Write("Результат вычисления синуса равен ")
       Console.Write(Format(Y, "0.000"))
       Console.Read()
    End Sub
End Module
```

Рассмотрим теперь программу, в которой осуществляется обмен значений двух переменных. Если просто записать в переменную (в ячейку памяти) новое данное, то предыдущая информация, содержащаяся в переменной, пропадет. Поэтому для обмена значений переменных необходимо предварительно определить еще и третью (вспомогательную) переменную. Она понадобится для промежуточного запоминания значения одной из двух исходных переменных. В листинге 1.17 приведен пример программы, которая реализует этот обмен.

Листинг 1.17. Обмен значений двух переменных

```
Module Module1
    Sub Main()
        Dim N, M, W As Integer
        Console.WriteLine("Введите значение N")
        N = Console.ReadLine()
        Console.WriteLine("Введите значение M ")
        M = Console.ReadLine()
```

```
        W = N
        N = M
        M = W
        Console.Write ("N = ")
        Console.WriteLine (N)
        Console.Write ("M = ")
        Console.WriteLine (M)
        Console.Read()
    End Sub
End Module
```

Хотя пока сделаны только первые шаги в плане алгоритмизации задач, но, тем не менее, у нас уже сформирована необходимая база для создания несложных разработок.

Константы

Иногда в программе необходимо использовать данные, которые в ходе ее выполнения не изменяются. Для работы с подобной информацией в Бейсике предусмотрена такая категория, как *константы*. Часто они определяются (имя и значение) в начале программы и далее просто используются. Пример, приведенный в листинге 1.18, дает представление о том, как в плане синтаксиса реализуется определение констант в тексте программы. Так, мы ввели константу `Koeff`, представляющую некоторый коэффициент, и далее использовали ее при вычислении значения переменной `Z`.

Листинг 1.18. Пример использования константы

```
Module Module1
    Sub Main()
        Const Koeff = 0.55
        Dim X, Z As Single
        Console.Write("Введите X ")
        X = Console.ReadLine()
        Z = Koeff * X
        Console.Write("Результат равен  ")
        Console.WriteLine(Z)
        Console.Read()
    End Sub
End Module
```

Кроме числовых констант существуют и другие их виды:

- *логические* константы служат для проверки истинности или ложности некоторых условий в программе и могут принимать только одно из двух значений: `True` или `False`;

- *символьные* константы могут принимать значение любого печатаемого символа и записываются как символ, заключенный в двойные кавычки.

Кроме того, можно использовать и *строковые константы* — это любые последовательности символов, заключенные в двойные кавычки. Как правило, строковые константы служат для записи приглашений к вводу данных, выдаваемых программой, вывода диагностических сообщений и т. д. Например, строковые константы можно записать так:

```
"Введите значение X: "
"Ответ="
```

Именованные константы перечисляются в разделе описаний программы оператором `Const` в следующем виде:

```
Const Имя [ As тип данных ] = Значение
```

В квадратных скобках определяется часть, которая может отсутствовать при описании. Так, мы можем указать только имя константы и присваиваемое ей значение. Тип данных может при описании константы не указываться.

Работа с символами и строками

Кроме чисел, другим распространенным видом данных являются *символы* и наборы символов. В данном разделе мы об этом и поговорим. В Бейсике для переменных, предназначенных для размещения одиночных символов, существует тип `Char`.

За каждым символом закреплен определенный код из шестнадцати двоичных битов (кодировка Unicode). В листинге 1.19 представлен пример, который демонстрирует использование двух функций: `Asc` и `Chr`, предназначенных для работы с символами.

Листинг 1.19. Пример работы с символами

```
Module Module1
    Sub Main()
        Dim Z, W As Char
        Dim X As Integer
        W = "A"
        X = Asc(W)
        Console.Write("X равно  ")
        Console.WriteLine(X)
        Z = Chr(X)
        Console.Write("Z равно  ")
        Console.WriteLine(Z)
        Console.Read()
    End Sub
End Module
```

Здесь мы использовали ключевое слово `Char` для описания переменных `Z` и `W`, предназначенных для размещения в них символов. Далее, используя уже знакомые символы двойных кавычек ("), мы поместили в переменную `W` код символа `"A"`.

Для получения кода определенного символа используется стандартная функция `Asc`, которая возвращает числовое значение кода символа, задаваемого в качестве параметра. Другая функция `Chr` выполняет обратное преобразование (формирует символ по его коду).

В листинге 1.20 приведена модификация предыдущей программы. При этом вместо переменной `W` мы использовали именованную константу:

```
Const W = "A"
```

С помощью добавления к коду символа `"A"` единицы мы в итоге получаем на экране символ `"B"`.

Листинг 1.20. Пример использования константы при работе с символами

```
Module Module1
    Sub Main()
        Const W = "A"
        Dim Z As Char
        Dim X As Integer
        X = Asc(W)+1
        Console.Write("X равно  ")
        Console.WriteLine(X)
        Z = Chr(X)
        Console.Write("Z равно  ")
        Console.WriteLine(Z)
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще один пример (листинг 1.21), связанный с набором символов (строки символов). Здесь после ввода строки символов с клавиатуры мы просто выводим данную строку на экран.

Листинг 1.21. Ввод и последующий вывод строки символов на экран

```
Module Module1
    Sub Main()
        Dim X As String
        X = Console.ReadLine()
        Console.WriteLine("Введенная строка")
        Console.Write(X)
        Console.Read()
    End Sub
End Module
```

Еще один пример приведен в листинге 1.22. Здесь при вводе строки данных мы выделяем из нее первый и последний символы. Эти символы в итоге отображаются на экране. При этом мы воспользовались стандартной функцией `Mid`, которая позволяет выделить из исходной строки подстроку (часть строки). У данной функции три параметра:

- первый параметр представляет собой исходную строку;
- второй параметр является номером символа, с которого производится выделение подстроки;
- третий параметр определяет количество символов, которые выделяются из исходной строки.

Листинг 1.22. Выделение двух символов из строки

```
Module Module1
    Sub Main()
        Dim X As String
        Dim N As Integer
        X = Console.ReadLine()
        N = Len(X)
        Console.WriteLine("Результат")
        Console.WriteLine (Mid(X, 1, 1))
        Console.WriteLine (Mid(X, N, 1))
        Console.Read()
    End Sub
End Module
```

Примеры

ПРИМЕР 1.1

Необходимо вычислить сумму длин сторон прямоугольника (периметр прямоугольника). С клавиатуры вводятся значения двух соседних сторон рассматриваемого прямоугольника. Результат вычисления периметра следует вывести на экран. Текст необходимой программной разработки для решения данной задачи приведен в листинге 1.23. Здесь используются два оператора ввода, в диалоге с которыми мы должны ввести два числа, обозначающие соседние стороны прямоугольника (для них в программе отводятся переменные `A` и `B`). После этого вычисляется удвоенная сумма сторон, которая и выводится на экран.

Листинг 1.23. Вычисление периметра прямоугольника

```
Module Module1
    Sub Main()
        Dim A, B, S As Single
```

```
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
        B = Console.ReadLine()
        S = 2 * (A + B)
        Console.Write("Периметр равен  ")
        Console.Write(S)
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.2

Необходимо вычислить косинус и тангенс угла, значение которого вводится с клавиатуры. Будем считать, что вводимое с клавиатуры значение угла измеряется в градусах. Однако стандартные тригонометрические функции системы Visual Basic требуют указания аргумента в радианах. В листинге 1.24 представлена необходимая программа, где мы воспользовались двумя стандартными функциями:

- `Math.Cos(Y)` — для вычисления косинуса;
- `Math.Tan(Y)` — для вычисления тангенса.

При вычислениях нам понадобилось число π, для чего мы воспользовались данной константой, заложенной в систему Visual Basic (`Math.PI`). В тексте программы осуществляется пересчет угла из градусов в радианы, после чего выполняются вычисления тригонометрических функций.

Листинг 1.24. Вычисление тригонометрических функций

```
Module Module1
    Sub Main()
        Dim Y, C, S As Single
        Console.Write("Введите Y ")
        Y = Console.ReadLine()
        Y = Y * Math.PI / 180
        C = Math.Cos(Y)
        Console.Write("Косинус равен ")
        Console.WriteLine(C)
        S = Math.Tan(Y)
        Console.Write("Тангенс равен ")
        Console.Write(S)
        Console.Read()
    End Sub
End Module
```

В листинге 1.25 приведена модификация предыдущей программы, где мы воспользовались уже знакомой функцией `Format` для форматирования результатов (обязательная цифра в разряде единиц целой части и две цифры в дробной части).

Листинг 1.25. Вычисление тригонометрических функций с форматированием

```
Module Module1
    Sub Main()
        Dim Y, C, S As Single
        Console.Write("Введите Y ")
        Y = Console.ReadLine()
        Y = Y * Math.PI / 180
        C = Math.Cos(Y)
        Console.Write("Косинус равен ")
        Console.WriteLine(Format(C, "0.00"))
        S = Math.Tan(Y)
        Console.Write("Тангенс равен ")
        Console.Write(Format(S, "0.00"))
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.3

Требуется написать программу, которая будет вычислять стоимость покупки в магазине канцтоваров. Для определенности будем считать, что в покупаемый набор входят тетради и ручки (и тетради и ручки одного вида). С клавиатуры вводятся:

- стоимость одной тетради;
- стоимость одной ручки;
- количество тетрадей и количество ручек.

После ввода необходимой информации в программе (листинг 1.26) выполняются арифметические действия, а результат отображается на экране. При выводе округление осуществляется до 10 копеек.

Листинг 1.26. Вычисление стоимости покупки канцтоваров

```
Module Module1
    Sub Main()
        Dim Ct, Ck, Sum As Single
        Dim Nt, Nk As Integer
        Console.Write("Введите цену тетради ")
        Ct = Console.ReadLine()
        Console.Write("Введите цену карандаша ")
        Ck = Console.ReadLine()
        Console.Write("Введите количество тетрадей ")
        Nt = Console.ReadLine()
        Console.Write("Введите количество карандашей ")
        Nk = Console.ReadLine()
        Sum = Nt * Ct + Nk * Ck
```

```
        Console.Write("Сумма покупки равна ")
        Console.WriteLine(Format(Sum, "0.0"))
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.4

Необходимо выполнить вычисление площади прямоугольного треугольника. При этом два катета треугольника вводятся с клавиатуры.

Здесь мы должны вспомнить формулу для вычисления площади прямоугольного треугольника:

$$S = \frac{a \cdot b}{2}, \tag{1.1}$$

где a и b обозначают длины катетов. В листинге 1.27 представлена необходимая программа, в которой после ввода двух параметров вычисляется площадь треугольника.

Листинг 1.27. Вычисление площади прямоугольного треугольника

```
Module Module1
    Sub Main()
        Dim A, B, S As Single
        Console.Write("Введите значение 1-го катета ")
        A = Console.ReadLine()
        Console.Write("Введите значение 2-го катета ")
        B = Console.ReadLine()
        S = A * B / 2
        Console.Write("Площадь равна ")
        Console.WriteLine(Format(S, "0.000"))
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.5

Требуется написать программу, которая должна рассчитать время поездки из пункта *A* в пункт *C*. При этом необходимо сначала доехать по шоссе из пункта *A* в *B*, а затем по грунтовой дороге из *B* в пункт *C*. Длины путей (в километрах) из *A* в *B* и из *B* в *C* вводятся с клавиатуры. Также с клавиатуры вводятся скорости по шоссе и по грунтовой дороге (единица измерения для скоростей — километры в час (км/час)). Время в пути необходимо вычислить с точностью до часа (следует округлить вычисленное значение).

Программа должна выполнить арифметические действия и отобразить результат (время в пути из пункта *A* в *C*) на экране. В листинге 1.28 представлена необходи-

мая программа. При выводе результата мы использовали форматирование с включением только одной обязательной цифры в разряде единиц — `Format(T,"0")`.

Для округления можно воспользоваться стандартной функцией `T=Math.Round(T)`.

Листинг 1.28. Вычисление времени в пути

```
Module Module1
    Sub Main()
        Dim Dab, Dbc, Vab, Vbc, T As Single
        Console.Write("Ввод пути из A в B  ")
        Dab = Console.ReadLine()
        Console.Write("Ввод пути из B в C  ")
        Dbc = Console.ReadLine()
        Console.Write("Ввод скорости по шоссе   ")
        Vab = Console.ReadLine()
        Console.Write("Ввод скорости по грунтовой дороге  ")
        Vbc = Console.ReadLine()
        T = Dab / Vab + Dbc / Vbc
        Console.WriteLine(Format(T, "0"))
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.6

Необходимо написать программу, которая должна выполнить преобразование времени, выраженного в секундах, в соответствующее значение, представленное в минутах и секундах. После выполнения необходимых преобразований результат следует отобразить на экране. В листинге 1.29 представлена программа, решающая поставленную задачу. Для получения количества минут мы воспользовались целочисленным делением исходного значения на 60. В данном случае остаток от целочисленного деления является числом секунд, добавляемым к вычисленным минутам.

Вывод информации программой показан на рис. 1.13.

Листинг 1.29. Преобразование значения времени в минуты и секунды

```
Module Module1
    Sub Main()
        Dim Vrema, Minuti, Secundi As Integer
        Console.Write("Ввод значения времени в секундах   ")
        Vrema = Console.ReadLine()
        Minuti = Vrema \ 60
        Secundi = Vrema Mod 60
        Console.Write(Minuti)
        Console.Write(" мин ")
```

```
        Console.Write(Secundi)
        Console.Write(" сек ")
        Console.Read()
    End Sub
End Module
```

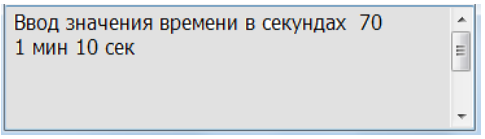

Рис. 1.13. Вывод информации в программе, приведенной в листинге 1.29

ПРИМЕР 1.7

Требуется написать программу, которая должна выполнить преобразование времени, выраженного в секундах, в соответствующее значение, только представленное в часах, минутах и секундах. Программа (листинг 1.30) получается лишь немного сложнее предыдущего примера. Здесь лишь надо учесть, что в часе 3600 секунд.

Вывод информации программой показан на рис. 1.14.

Листинг 1.30. Преобразование значения времени в часы, минуты и секунды

```
Module Module1
    Sub Main()
        Dim Vrema, Chas, Minuti, Secundi As Integer
        Console.Write("Ввод значения времени в секундах   ")
        Vrema = Console.ReadLine()
        Chas = Vrema \ 3600
        Minuti = (Vrema Mod 3600) \ 60
        Secundi = Vrema — Chas * 3600 — Minuti * 60
        Console.Write(Chas)
        Console.Write(" час ")
        Console.Write(Minuti)
        Console.Write(" мин ")
        Console.Write(Secundi)
        Console.Write(" сек ")
        Console.Read()
    End Sub
End Module
```

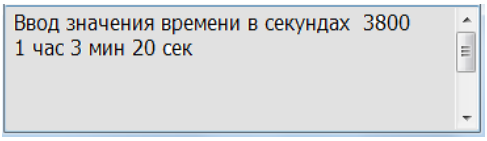

Рис. 1.14. Вывод информации в программе, приведенной в листинге 1.30

ПРИМЕР 1.8

С клавиатуры вводится двузначное число. Необходимо сформировать другое двузначное число, в котором цифры исходного числа будут переставлены. Например, если исходное число 57, то число с переставленными цифрами равно 75. В листинге 1.31 приведена соответствующая программа. Здесь мы воспользовались уже знакомыми операциями `\` и `Mod` для выделения первой и второй цифры в числе. После этого цифра, присутствующая в разряде единиц, умножается на 10 (становится в новом числе цифрой в разряде десятков) и складывается с другой.

Вывод информации программой показан на рис. 1.15.

Листинг 1.31. Перестановка цифр в двузначном числе

```
Module Module1
    Sub Main()
        Dim A As Integer
        Console.Write("Введите двузначное целое число ")
        A = Console.ReadLine()
        A = (A \ 10) + (A Mod 10) * 10
        Console.Write("Число с перестановкой цифр равно ")
        Console.WriteLine(A)
        Console.Read()
    End Sub
End Module
```

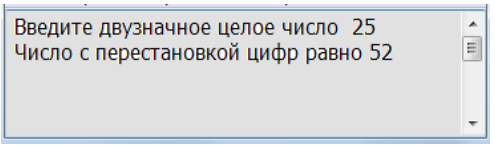

Рис. 1.15. Вывод информации в программе, приведенной в листинге 1.31

ПРИМЕР 1.9

С клавиатуры вводится трехзначное число. Необходимо подсчитать сумму цифр этого числа. Например, если исходное число 257, то число с просуммированными цифрами равно 14. В листинге 1.32 приведена программная разработка, решающая данную задачу. Переменные `B`, `C` и `D` мы отвели, соответственно, для первой, третьей и второй цифры исходного числа.

Вывод информации программой показан на рис. 1.16.

Листинг 1.32. Суммирование цифр в трехзначном числе

```
Module Module1
    Sub Main()
        Dim A, B, C, D, S As Integer
```

```
        Console.Write("Введите трехзначное целое число ")
        A = Console.ReadLine()
        B = A Mod 10
        C = A \ 100
        D = (A Mod 100) \ 10
        S = B + C + D
        Console.Write("Сумма цифр равна ")
        Console.WriteLine(S)
        Console.Read()
    End Sub
End Module
```

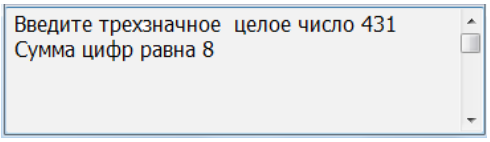

Рис. 1.16. Вывод информации в программе, приведенной в листинге 1.32

ПРИМЕР 1.10

С клавиатуры вводится четырехзначное число. Необходимо подсчитать сумму цифр этого числа. Например, если исходное число 1257, то число с просуммированными цифрами равно 15. В листинге 1.33 приведена программа, которая является фактически чуть более усложненным вариантом предыдущей разработки. Назначение переменных выглядит так:

- `B` — для цифры в разряде единиц;
- `C` — для цифры в разряде тысяч;
- `D` — для цифры в разряде десятков;
- `E` — для цифры в разряде сотен.

Вывод информации программой показан на рис. 1.17.

Листинг 1.33. Суммирование цифр в четырехзначном числе

```
Module Module1
    Sub Main()
        Dim A, B, C, D, E, S As Integer
        Console.Write("Введите четырехзначное целое число ")
        A = Console.ReadLine()
        B = A Mod 10
        C = A \ 1000
        D = (A Mod 100) \ 10
        E = (A Mod 1000) \ 100
        S = B + C + D + E
```

```
        Console.Write("Сумма цифр равна ")
        Console.WriteLine(S)
        Console.Read()
    End Sub
End Module
```

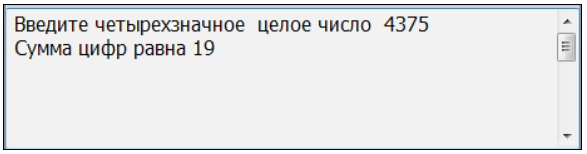

Рис. 1.17. Вывод информации в программе, приведенной в листинге 1.33

ПРИМЕР 1.11

С клавиатуры вводится вещественное число. Необходимо определить первую цифру дробной части этого числа. Например, если исходное число 1257.237, то такая цифра равна 2. В листинге 1.34 приведена программа, которая реализует поставленную задачу. Мы воспользовались умножением исходного числа на 10 для перевода первой цифры из дробной части в разряд единиц целой части. После этого с помощью стандартной функции `Int` преобразуем полученный результат к целому числу. Далее осталось сформировать только остаток от целочисленного деления полученного результата на 10.

Вывод информации программой показан на рис. 1.18.

Листинг 1.34. Выделение первой цифры дробной части числа

```
Module Module1
    Sub Main()
        Dim A As Integer
        Dim B As Single
        Console.Write("Введите дробное число ")
        B = Console.ReadLine()
        A = Int(B * 10) Mod 10
        Console.Write("Первая цифра дробной части равна ")
        Console.WriteLine(A)
        Console.Read()
    End Sub
End Module
```

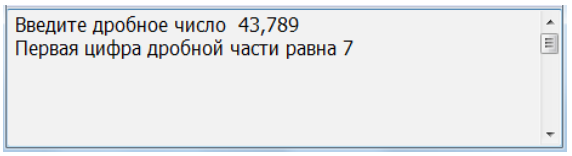

Рис. 1.18. Вывод информации в программе, приведенной в листинге 1.34

ПРИМЕР 1.12

С клавиатуры вводится вещественное число. Необходимо найти сумму первых двух (старших) цифр дробной части числа. Например, если исходное число 1257.237, то сумма двух первых цифр дробной части равна 5. В листинге 1.35 приведена необходимая программа. Аналогично шагам предыдущего примера мы сначала переносим первые две цифры из дробной части в целую. После этого отбрасываем оставшуюся дробную часть и формируем двузначное число. На завершающем этапе следует выделение цифр целого числа.

Вывод информации программой показан на рис. 1.19.

Листинг 1.35. Выделение двух первых цифр дробной части числа

```
Module Module1
    Sub Main()
        Dim A, C, D, Sum As Integer
        Dim B As Single
        Console.Write("Введите дробное число ")
        B = Console.ReadLine()
        A = Int(B * 100) Mod 100
        C = A Mod 10
        D = A \ 10
        Sum = C + D
        Console.Write("Сумма равна ")
        Console.WriteLine(Sum)
        Console.Read()
    End Sub
End Module
```

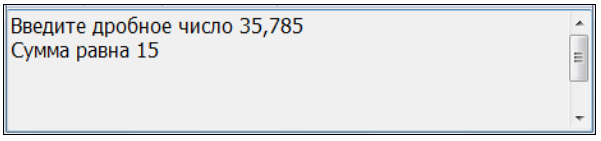

Рис. 1.19. Вывод информации в программе, приведенной в листинге 1.35

ПРИМЕР 1.13

С клавиатуры вводится четырехзначное число. Необходимо в этом числе переставить первую и вторую цифру (поменять цифру в разряде единиц и цифру в разряде десятков). В листинге 1.36 приведена необходимая программа, в которой основные используемые приемы нам уже знакомы. Так, для выделения двух младших разрядов в числе `A` мы воспользовались двумя операторами:

- `B = A Mod 10`;
- `C =(A Mod 100) \ 10`.

После этого в исходном числе необходимо оставить цифры только в разрядах сотен и тысяч (получить исходное число без цифр в разрядах десятков и единиц). Таким образом, в результате мы получили составляющие искомого числа. На завершающем этапе эти составляющие осталось объединить в единое целое:

`E= D + C + B*10`.

Вывод информации программой показан на рис. 1.20.

Листинг 1.36. Перестановка цифр в числе

```
Module Module1
    Sub Main()
        Dim A, B, C, D, E As Integer
        Console.Write("Введите четырехзначное число  ")
        A = Console.ReadLine()
        B = A Mod 10
        C = (A Mod 100) \ 10
        D = (A \ 100) * 100
        E = D + C + B * 10
        Console.Write("Число с переставленными цифрами равно  ")
        Console.WriteLine(E)
        Console.Read()
    End Sub
End Module
```

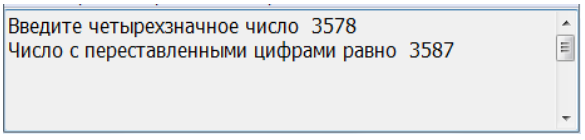

Рис. 1.20. Вывод информации в программе, приведенной в листинге 1.36

ПРИМЕР 1.14

С клавиатуры вводятся координаты точки `(X, Y)`, расположенной на плоскости. Необходимо программно определить, попадает ли точка в область, определяемую следующими соотношениями:

`X>5` и `Y>7`.

Если точка попадает, то на экран необходимо выдать соответствующее информирующее сообщение. Соответственно, если точка не попала в указанную область на экране, то это также должно быть отражено соответствующим сообщением.

В разработке (листинг 1.37) мы воспользовались переменной логического (булевого) типа.

Листинг 1.37. Проверка принадлежности точки определенной области

```
Module Module1
    Sub Main()
        Dim X, Y As Single
        Dim W As Boolean
        Console.Write("Введите координату X ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y ")
        Y = Console.ReadLine()
        W = (X > 5) And (Y > 7)
        Console.Write("Точка принадлежит области ")
        Console.WriteLine(W)
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.15

С клавиатуры вводятся координаты точки (X, Y), расположенной на плоскости. Необходимо программно определить: попадает ли точка в круг с центром в начале координат и радиусом 1?

Здесь мы должны вычислить расстояние от начала координат до точки на плоскости. При этом нам понадобится возведение чисел в квадрат. Для этого мы воспользуемся стандартной функцией. Разработка, реализующая поставленную задачу, приведена в листинге 1.38.

Листинг 1.38. Проверка принадлежности точки заданному кругу

```
Module Module1
    Sub Main()
        Dim X, Y, Z As Single
        Dim W As Boolean
        Console.Write("Введите координату X")
        X = Console.ReadLine()
        Console.Write("Введите координату Y")
        Y = Console.ReadLine()
        Z = Math.Pow(X, 2) + Math.Pow(Y, 2)
        W = (Z <= 1)
        Console.Write("Точка принадлежит области ")
        Console.WriteLine(W)
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.16

С клавиатуры вводятся координаты точки (X, Y), расположенной на плоскости. Необходимо программно определить: попадает ли точка в круг радиуса 5 с центром в точке X0 и Y0. Предполагается, что координаты центра круга также вводятся с клавиатуры. Разработка, реализующая поставленную задачу, приведена в листинге 1.39.

Листинг 1.39. Проверка принадлежности точки кругу

```
Module Module1
    Sub Main()
        Dim X, Y, X0, Y0, Z As Single
        Dim W As Boolean
        Console.Write("Введите координату X точки")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        Console.Write("Введите координату X0 центра круга")
        X0 = Console.ReadLine()
        Console.Write("Введите координату Y0 центра круга ")
        Y0 = Console.ReadLine()
        Z = Math.Sqrt(Math.Pow(X — X0, 2) + Math.Pow(Y — Y0, 2))
        W = (Z <= 5)
        Console.Write("Точка принадлежит области ")
        Console.WriteLine(W)
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.17

На плоскости расположен круг радиуса R с центром в точке X0 и Y0. Указанные параметры вводятся с клавиатуры. Необходимо программно определить: попадает ли центр координат в данный круг. Программа, решающая поставленную задачу, приведена в листинге 1.40.

Листинг 1.40. Проверка принадлежности центра координат кругу

```
Module Module1
    Sub Main()
        Dim X0, Y0, R, Z As Single
        Dim W As Boolean
        Console.Write("Введите координату X0 центра круга ")
        X0 = Console.ReadLine()
        Console.Write("Введите координату Y0 центра круга  ")
        Y0 = Console.ReadLine()
        Console.Write("Введите радиус круга ")
```

```
        R = Console.ReadLine()
        Z = Math.Sqrt(Math.Pow(X0, 2) + Math.Pow(Y0, 2))
        W = (Z <= R)
        Console.Write("Центр координат принадлежит кругу  ")
        Console.WriteLine(W)
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.18

В плоскости расположены две точки. Координаты точек вводятся с клавиатуры. Необходимо программно определить расстояние между точками. Для решения задачи здесь необходимо воспользоваться теоремой Пифагора. Программа, решающая поставленную задачу, приведена в листинге 1.41.

Листинг 1.41. Вычисление расстояния между точками

```
Module Module1
    Sub Main()
        Dim X1, Y1, X2, Y2, Z As Single
        Console.Write("Введите координату X первой точки ")
        X1 = Console.ReadLine()
        Console.Write("Введите координату Y первой точки  ")
        Y1 = Console.ReadLine()
        Console.Write("Введите координату X второй точки ")
        X2 = Console.ReadLine()
        Console.Write("Введите координату Y второй точки  ")
        Y2 = Console.ReadLine()
        Z = Math.Sqrt(Math.Pow(X1 - X2, 2) + Math.Pow(Y1 - Y2, 2))
        Console.Write("Расстояние равно  ")
        Console.WriteLine(Z)
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.19

В плоскости построен треугольник. Координаты вершин треугольника вводятся с клавиатуры. Необходимо программно определить периметр треугольника. Текст программной разработки приведен в листинге 1.42.

Листинг 1.42. Вычисление периметра треугольника

```
Module Module1
    Sub Main()
        Dim X1, Y1, X2, Y2, X3, Y3, Z As Single
```

```
        Console.Write("Введите координату X первой вершины  ")
        X1 = Console.ReadLine()
        Console.Write("Введите координату Y первой вершины   ")
        Y1 = Console.ReadLine()
        Console.Write("Введите координату X второй  вершины  ")
        X2 = Console.ReadLine()
        Console.Write("Введите координату Y второй  вершины   ")
        Y2 = Console.ReadLine()
        Console.Write("Введите координату X третьей   вершины  ")
        X3 = Console.ReadLine()
        Console.Write("Введите координату Y третьей   вершины   ")
        Y3 = Console.ReadLine()
        Z = Math.Sqrt((X1 — X2) ^ 2 + (Y1 — Y2) ^ 2)
        Z = Z + Math.Sqrt((X2 — X3) ^ 2 + (Y2 — Y3) ^ 2)
        Z = Z + Math.Sqrt((X1 — X3) ^ 2 + (Y1 — Y3) ^ 2)
        Console.Write("Периметр равен  ")
        Console.WriteLine(Z)
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.20

С клавиатуры вводится трехзначное число. Необходимо определить, есть ли в этом числе одинаковые цифры. Например, если исходное число 525, то программа должна выдать ответ "`True`". В листинге 1.43 приведена программа, решающая поставленную задачу. Назначение переменных таково:

- `B` — для цифры в разряде единиц;
- `C` — для цифры в разряде десятков;
- `D` — для цифры в разряде сотен.

Листинг 1.43. Анализ наличия одинаковых цифр в трехзначном числе

```
Module Module1
    Sub Main()
        Dim A, B, C, D As Integer
        Dim W As Boolean
        Console.Write("Введите трехзначное число ")
        A = Console.ReadLine()
        B = A Mod 10
        C = (A Mod 100) \ 10
        D = A \ 100
        W = (B = C) Or (B = D) Or (C = D)
        Console.Write("Одинаковые цифры в числе ")
        Console.WriteLine(W)
```

```
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 1.21

С клавиатуры вводятся три различных числа. Необходимо сделать вывод о том, какое из чисел является максимальным. В листинге 1.44 приведена программа, решающая поставленную задачу.

Листинг 1.44. Поиск максимального числа

```
Module Module1
    Sub Main()
        Dim A, B, C As Integer
        Dim W1, W2, W3 As Boolean
        Console.Write("Введите первое число ")
        A = Console.ReadLine()
        Console.Write("Введите второе число ")
        B = Console.ReadLine()
        Console.Write("Введите третье число ")
        C = Console.ReadLine()
        W1 = (A > B) And (A > C)
        W2 = (B > A) And (B > C)
        W3 = (C > A) And (C > B)
        Console.Write("A максимальное  ")
        Console.WriteLine(W1)
        Console.Write("B максимальное  ")
        Console.WriteLine(W2)
        Console.Write("C максимальное  ")
        Console.WriteLine(W3)
        Console.Read()
    End Sub
End Module
```

Если не включать условие единственности максимального числа, то фрагмент листинга 1.38 с установлением значений `W1`, `W2` и `W3` изменится на следующий:

```
W1= (A>=B) And (A>=C)
W2= (B>=A) And (B>=C)
W3= (C>=A) And (C>=B)
```

ПРИМЕР 1.22

С клавиатуры вводятся три числа. Необходимо сделать вывод о том, все ли они равны между собой или нет. В листинге 1.45 приведена программа, решающая поставленную задачу.

Листинг 1.45. Проверка чисел на равенство

```
Module Module1
    Sub Main()
        Dim A, B, C As Integer
        Dim W As Boolean
        Console.Write("Введите первое число ")
        A = Console.ReadLine()
        Console.Write("Введите второе число ")
        B = Console.ReadLine()
        Console.Write("Введите третье число ")
        C = Console.ReadLine()
        W = (A = B) And (A = C)
        Console.Write("Все числа одинаковые   ")
        Console.WriteLine(W)
        Console.Read()
    End Sub
End Module
```

Типовые задачи и задания из ЕГЭ

В этом разделе приведены готовые листинги программ, которые вам необходимо проанализировать. Создания каких-либо программных разработок здесь не требуется. В задачах желательно найти правильные ответы без использования компьютера. В последнем разделе главы приведены правильные ответы.

ЗАДАЧА 1.1

В листинге 1.46 приведена программа с использованием действий над целыми числами. Необходимо определить значения переменных `A` и `B`, которые будут выведены на экран.

Листинг 1.46. Программа к задаче 1.1

```
Module Module1
    Sub Main()
        Dim A, B As Integer
        A = 3 + 8 * 4
        B = (A Mod 10) + 14
        A = (B Mod 10) + 2
        Console.Write("A = ")
        Console.WriteLine(A)
        Console.Write("B = ")
        Console.WriteLine(B)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.2

В листинге 1.47 приведена программа с использованием действий над целыми числами. Необходимо определить значения переменных `A` и `B`.

Листинг 1.47. Программа к задаче 1.2

```
Module Module1
    Sub Main()
        Dim A, B As Integer
        A = 1819
        B = (A \ 100) * 10 + 9
        A = (10 * B — A) Mod 100
        Console.Write("A = ")
        Console.WriteLine(A)
        Console.Write("B = ")
        Console.WriteLine(B)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.3

В листинге 1.48 приведена программа с использованием действий над целыми числами. Необходимо без выполнения ее на компьютере определить значение переменной `A`.

Листинг 1.48. Программа к задаче 1.3

```
Module Module1
    Sub Main()
        Dim A, B As Integer
        A = 1
        B = 9
        A = -A + B * 2
        A = 4 * A Mod 10
        Console.Write("A = ")
        Console.WriteLine(A)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.4

В листинге 1.49 приведен фрагмент программы. Необходимо определить, что появится на экране после ее выполнения.

Листинг 1.49. Программа к задаче 1.4

```
Module Module1
    Sub Main()
        Dim A As Integer
        A = 1
        A = A * 200
        Console.WriteLine(" A")
        Console.Write(A)
        Console.Write(A)
        Console.WriteLine(A)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.5

В листинге 1.50 приведен фрагмент программы. Необходимо без выполнения ее на компьютере определить, что появится на экране после ее выполнения.

Листинг 1.50. Программа к задаче 1.5

```
Module Module1
    Sub Main()
        Dim A As Integer
        A = 100
        A = A + 1
        A = A + 4
        Console.WriteLine(A)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.6

В листинге 1.51 приведен фрагмент программы. Необходимо определить, что появится на экране после ее выполнения.

Листинг 1.51. Программа к задаче 1.6

```
Module Module1
    Sub Main()
        Dim A As Integer
        A = 100
        A = A – A
        A = A + 5
```

```
        Console.WriteLine(A)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.7

В листинге 1.52 приведена программа. Необходимо определить, что появится на экране после ее выполнения.

Листинг 1.52. Программа к задаче 1.7

```
Module Module1
    Sub Main()
        Dim N As Integer
        N = 2
        N = N * N * N
        N = N ^ 2 * N ^ 2
        Console.WriteLine(N)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.8

В листинге 1.53 приведен фрагмент программы с использованием вещественных чисел. Необходимо определить, что появится на экране после ее выполнения.

Листинг 1.53. Программа к задаче 1.8

```
Module Module1
    Sub Main()
        Dim Z, W, X As Single
        W = 2 * 2.54
        Z = 1.15
        X = W — 2 * Z
        Console.WriteLine(Format(X, "0.0"))
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.9

В листинге 1.54 приведен пример программы, которая касается вычислительных действий над целыми числами. Необходимо определить результат, который мы увидим на экране после выполнения программы.

Листинг 1.54. Программа к задаче 1.9

```
Module Module1
    Sub Main()
        Dim N, M, W As Integer
        N = 7
        M = 12
        N = N + 1
        W = N
        N = M
        M = W
        Console.WriteLine(N)
        Console.WriteLine(M)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.10

В листинге 1.55 приведена программа, которая реализует использование знакомой функции, предназначенной для работы с символами. Необходимо определить, что будет выведено на экран.

Листинг 1.55. Программа к задаче 1.10

```
Module Module1
    Sub Main()
        Dim X As Integer
        Dim W As Char
        W = "5"
        X = Asc(W) - Asc("0")
        W = "2"
        X = 10 * X + Asc(W) - Asc("0")
        Console.WriteLine(X)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.11

В листинге 1.56 приведена программа, в которой используется уже знакомая функция для работы с символами. Необходимо определить, что будет выведено на экран.

Листинг 1.56. Программа к задаче 1.11

```
Module Module1
    Sub Main()
        Dim W As Char
```

```
        Dim X As Integer
        W = "4"
        X = Asc(W) — Asc("9")
        W = "5"
        X = 10 * X + Asc(W) — Asc("7")
        Console.WriteLine(X)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.12

В листинге 1.57 приведена программа, которая использует две уже знакомые функции для работы с символами. Необходимо определить, что будет выведено на экран в конце программы.

Листинг 1.57. Программа к задаче 1.12

```
Module Module1
    Sub Main()
        Dim W As Char
        Dim X As Integer
        W = "5"
        X = Asc(W) — 1
        W = Chr(X + 1)
        X = 10 * (Asc(W) — Asc("4"))
        Console.WriteLine(W)
        Console.WriteLine(X)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.13

В листинге 1.58 приведена программа с использованием действий над целыми числами. Необходимо определить значение переменной C, которое будет выведено на экран в конце программы.

Листинг 1.58. Программа к задаче 1.13

```
Module Module1
    Sub Main()
        Dim A, B, C As Integer
        A = 7
        A = A — 4
        B = -A
        C = -A + 2 * B
```

```
        Console.WriteLine(C)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.14

В листинге 1.59 приведена программа с использованием действий над целыми числами. Необходимо определить значения переменных `X` и `Y`, которые в конце программы будут выведены на экран.

Листинг 1.59. Программа к задаче 1.14

```
Module Module1
    Sub Main()
        Dim X, Y As Integer
        X = 8 + 2 * 5
        Y = (X Mod 10) + 14
        X = (Y \ 10) + 3
        Console.WriteLine(X)
        Console.WriteLine(Y)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 1.15

В листинге 1.60 приведена программа с использованием действий над целыми числами. Необходимо определить значения переменных `X` и `Y`, которые в конце программы будут выведены на экран.

Листинг 1.60. Программа к задаче 1.15

```
Module Module1
    Sub Main()
        Dim X, Y As Integer
        X = 4 + 8 * 3
        Y = (X Mod 10) + 15
        X = (Y \ 10) + 3
        Console.WriteLine(X)
        Console.WriteLine(Y)
        Console.Read()
    End Sub
End Module
```

Ответы к задачам и заданиям из ЕГЭ

Задача 1.1

Первый вычислительный оператор в программе листинга 1.46 приводит к результату: `A=35`. Остаток от деления 35 на 10 равен 5. Далее это число складывается с 14 и результат (19) записывается в переменную `B`. Таким образом, ответ на один вопрос — значение переменной `B` равно 19. Остаток от деления 19 на 10 равен 9. Далее это число складывается с числом 2 и результат (11) записывается в переменную `A`. В результате значение переменной `A` равняется 11.

Ответ: 11 и 19.

Задача 1.2

Результат целочисленного деления 1819 на 100 равен 18. Это значение умножается на 10, а результат складывается с числом 9 (в итоге получается 189). Таким образом, значение `B` после выполнения второго оператора равно 189. В следующем операторе выражение в скобках дает результат 71 (189 умножается на 10, а из результата вычитается 1819). Остаток от деления 71 на 100 равен 71. Таким образом, значение переменной `A` равняется 71, а переменной `B` — 189.

Задача 1.3

Первый вычислительный оператор в программе листинга 1.48 приводит к занесению в переменную `A` числа 17 (из 18 вычитается 1). Далее в следующем операторе порядок операций говорит о том, что сначала будет выполнена операция умножения, а затем вычисление остатка от деления 10. В результате в процессе вычисления мы получим число 8.

Задача 1.4

В программе листинга 1.49 использовано несколько операторов вывода информации на экран. При этом первый оператор выводит пробел и букву `A`, после чего выполняется перевод строки. В следующих строках три раза подряд выводится значение переменной `A`. В результате мы увидим:

```
 A
200200200
```

Задача 1.5

В связи с приведенным комментарием при постановке задачи значение переменной `A` сначала увеличится на 1, а затем еще на 4. В результате на экране мы увидим:

```
105
```

Задача 1.6

Учитывая содержание листинга 1.51, значение переменной `A` сначала уменьшается на 100, а затем увеличивается на 5. В результате на экране мы увидим число 5.

Задача 1.7

Здесь сначала 2 возводится в куб (результат равен 8). После этого выполняется вычисление квадрата числа 8, который умножается сам на себя. В результате получим 4096.

Задача 1.8

Здесь выполняются действия с дробными числами. Вывод результата представлен с ограничением числа разрядов дробной части.

Вычисления вручную дают:

```
5.08 - 2.3 = 2.78
```

Округление до одного разряда дробной части приводит к числу 2.8.

Задача 1.9

Фактически в тексте программы листинга 1.54 осуществляется обмен значений. При этом предварительно исходное значение переменной `N` увеличивается на единицу. В результате мы увидим на экране, что `N` равняется 12, а `M` примет значение 8.

Задача 1.10

Первое вычисление `X` представляет вычитание кодов двух символов. Учитывая, что коды цифр расположены плотно, вычисление приводит к ответу: `X=5`. Второе действие формирует ответ, равный 52.

Задача 1.11

Здесь первая вычислительная операция дает ответ: –5. Второе действие формирует ответ: –52.

Задача 1.12

Вычисление `X` во второй строке исполнительной части программы приводит к коду числа 4 (из кода числа 5 вычитается единица). В следующей строке получаем символ 5. Это значение будет первым выведено на экран. Второй результат — это число 10. Таким образом, на экране мы увидим:

```
5
10
```

Задача 1.13

Первый вычислительный оператор (после присвоения переменной `A` значения `7`) в программе листинга 1.58 приводит к результату: `A=3`. Далее вычисляется значение переменной `B`, что приводит к значению `-3`. После этого вычисляется значение переменной `C`, что дает значение `-9`. Это результат сложения `-3` и `-6`.

Ответ: –9.

Задача 1.14

В первой строке с вычислением рассчитывается значение переменной `X`, которое равно 18. Далее в следующей строке число 14 складывается с остатком от деления 18 на 10. В результате этого действия переменная `Y` принимает значение 22. В последней вычислительной строке производится целочисленное деление 22 на 10, что дает 2. К этому значению добавляется 3 и результат заносится в переменную `X`. Таким образом `X=5`.

Ответ: `X=5 Y=22`.

Задача 1.15

В первой строке с вычислением рассчитывается значение переменной `X`, которое равно 28. Далее в следующей строке число 15 складывается с остатком от деления 28 на 10. В результате этого действия переменная `Y` принимает значение 23. В последней вычислительной строке выполняется целочисленное деление 23 на 10, что дает 2. К этому значению добавляется 3 и результат заносится в переменную `X`.

Ответ: `X=5 Y=23`.

ГЛАВА 2

Условия, выбор и циклы

Программы, которые мы рассмотрели в *главе 1*, были достаточно простыми и заключались лишь в последовательном выполнении инструкций. Исходные данные на эту последовательность не влияли, и шаги алгоритма были одни и те же независимо от действий пользователя. В реальных ситуациях необходимо выполнять те или иные действия в зависимости от определенных входных значений (часто не буквально входных значений, а результатов текущих вычислений). Например, из курса математики вы знаете, что соотношения для нахождения решения квадратного уравнения различаются в зависимости от параметров (коэффициентов) этого уравнения. В языке Бейсик, как и в любом другом языке программирования, имеются средства для реализации подобных ситуаций — это несколько вариантов оператора условия.

В большом числе случаев при построении алгоритмов требуется выполнять периодически повторяющиеся действия. Для этого в языках программирования существуют операторы циклов. *Цикл* или циклический вычислительный процесс характеризуется повторением одних и тех же действий над различными данными. Числом повторений цикла управляет специальная переменная, называемая *счетчиком цикла*. На счетчик накладывается условие, определяющее, до каких пор следует выполнять цикл. Повторяемый блок вычислений называют *телом цикла*. В цикле должно быть обеспечено изменение значения счетчика, чтобы он мог завершиться через конечное число шагов. Однократное выполнение тела цикла называют его *шагом*.

В организации цикла часто подразумевается, что число шагов цикла (повторений) известно заранее. Однако встречаются и ситуации, когда количество прохождений цикла заранее неизвестно. Оно определяется лишь постепенно после некоторого количества повторений. В этом случае применяется другая разновидность цикла — это *цикл с условием*. В языке Бейсик предусмотрено два варианта цикла с условием:

- условие проверяется перед телом цикла;
- условие проверяется после тела цикла.

Далее в этой главе мы подробно рассмотрим как необходимые теоретические сведения по данной теме, так и практические примеры, которые помогут вам сформировать необходимые навыки в плане самостоятельного программирования.

Оператор условия

Это один из ключевых операторов в любом языке программирования. *Оператор условия* позволяет организовать дальнейшее выполнение алгоритма по одному из двух направлений. Эта возможность с учетом уже рассмотренных в *главе 1* базовых сведений по языку Бейсик позволяет реализовывать очень сложные алгоритмы.

Познакомимся с оператором условия на простом практическом примере программной разработки. Так, задача заключается во вводе целого числа с клавиатуры и его последующем анализе. Сам анализ достаточно простой: если число меньше 5, то его квадрат выводится на экран. В противном случае (если данное условие не выполняется) на экран ничего не выводится. В листинге 2.1 приведена программа, которая решает данную задачу.

Листинг 2.1. Пример использования однострочного оператора условия

```
Module Module1
    Sub Main()
        Dim Z As Integer
        Console.Write("Введите целое число ")
        Z = Console.ReadLine()
        If Z < 5 Then     Console.Write(Z^2)
        Console.Read()
    End Sub
End Module
```

Здесь мы использовали вариант однострочного оператора условия `If`. В этом случае условие и необходимые операторы, выполняемые при истинности данного условия, располагаются в одной строке (после ключевого слова `Then`). В данном случае при истинности условия `Z < 5` выполняется строка:

```
Console.Write(Z^2)
```

Здесь вычисляется квадрат введенного числа, а затем этот квадрат выводится на экран.

В одной строке после ключевого слова `Then` может быть записан не один, а несколько операторов. В этом случае операторы следует отделять друг от друга двоеточием. В качестве примера в листинге 2.2 приведена модификация предыдущей программы, где вместо одного оператора в строке мы использовали два.

Листинг 2.2. Использование нескольких операторов в однострочном операторе условия

```
Module Module1
    Sub Main()
        Dim Z, Z2 As Integer
        Console.Write("Введите целое число ")
```

```
        Z = Console.ReadLine()
        If Z < 5 Then Z2 = Z ^ 2 : Console.Write(Z2)
        Console.Read()
    End Sub
End Module
```

Если требуется выполнение блока операторов при выполнении условия, то формат оператора `If` выглядит следующим образом:

```
If условие Then
   операторы
End If
```

где под условием понимается любое выражение, результат которого имеет тип `Boolean`. Ключевая комбинация слов `End If` обозначает окончание многострочной конструкции условия.

В качестве примера в листинге 2.3 приведен вариант предыдущей программы, где использовано выполнение нескольких строк при истинности условия.

Листинг 2.3. Пример использования многострочного оператора условия

```
Module Module1
    Sub Main()
        Dim Z, Z2 As Integer
        Console.Write("Введите целое число ")
        Z = Console.ReadLine()
        If Z < 5 Then
            Z2 = Z ^ 2
            Console.Write("Квадрат введенного числа равен ")
            Console.Write(Z2)
        End If
        Console.Read()
    End Sub
End Module
```

Приведенные примеры демонстрируют неполную форму оператора условия. Это связано с тем, что не указан оператор (операторы), который должен выполняться, если условие ложно. Однако существует и *полная форма* оператора условия:

```
 If  условие Then
   операторы
 Else
   операторы
 End If
```

В этом случае при выполнении условия осуществляется выполнение операторов между ключевыми словами `Then` и `Else`. Если же условие не выполняется, то выполняются операторы, расположенные после `Else` вплоть до `End If`. Таким образом, в этом случае всегда функционирует одна из двух групп операторов.

Для иллюстрации полной формы оператора `If` в листинге 2.4 приведен пример, базирующийся на предыдущей программе. Здесь в случае, если введенное число меньше пяти, его квадрат выводится на экран, в противном же случае на экран выводится квадратный корень из введенного значения.

Листинг 2.4. Пример использования полной формы оператора условия

```
Module Module1
    Sub Main()
        Dim Z As Integer
        Console.Write("Введите целое число ")
        Z = Console.ReadLine()
        If Z < 5 Then
            Console.Write(Z^2)
        Else
            Console.Write(Math.Sqrt(Z))
        End If
        Console.Read()
    End Sub
End Module
```

В полной форме оператора `If` также можно использовать несколько операторов подряд в конструкции условия. Модернизация предыдущей программы с целью демонстрации этого приведена в листинге 2.5.

Листинг 2.5. Пример использования полной формы оператора условия

```
Module Module1
    Sub Main()
        Dim Z As Integer
        Dim Z2 As Single
        Console.Write("Введите целое число ")
        Z = Console.ReadLine()
        If Z < 5 Then
            Z = Z ^2
            Console.Write(Z)
        Else
            Z2 = Math.Sqrt(Z)
            Console.Write(Z2)
        End If
        Console.Read()
    End Sub
End Module
```

Приведем еще один пример, в котором осуществляется ввод с клавиатуры трех чисел. Далее в случае, если они все равны, на экран выводится соответствующее

сообщение. В случаях, если равна только одна пара чисел или если все введенные числа различны, также формируются соответствующие сообщения. Листинг 2.6 содержит текст данной программы. Здесь мы использовали *вложенный* оператор `If`. Так, после ключевого слова `Else` одного оператора `If` располагается еще один оператор `If`.

Листинг 2.6. Сопоставление трех введенных с клавиатуры чисел

```
Module Module1
    Sub Main()
        Dim Z, X, W As Integer
        Console.Write("Введите X ")
        X = Console.ReadLine()
        Console.Write("Введите Y ")
        W = Console.ReadLine()
        Console.Write("Введите Z ")
        Z = Console.ReadLine()
        If X = Z And X = W Then
            Console.Write("Все числа равны")
        Else
            If X = Z Or X = W Or W = Z Then
                Console.Write("Только два числа равны")
            Else
                Console.Write("Все числа разные")
            End If
        End If
        Console.Read()
    End Sub
End Module
```

В подобных ситуациях можно использовать форму *составного условного оператора*. Этот вариант удобно применять, когда имеется более двух вариантов расчета. Составной оператор может включать произвольное число условий и ветвей расчета. Его общий вид выглядит следующим образом:

```
If условие 1 Then
     операторы 1
ElseIf условие 2 Then
     операторы 2
   ...
ElseIf условие N Then
     операторы N
Else
     операторы 0
End If
```

При использовании такого оператора последовательно проверяются логические выражения (условия от первого до `N`-го). Если одно из этих выражений истинно, то

выполняется соответствующая группа операторов, после чего управление передается на оператор, располагающийся после `End If` (следующий за данным условным оператором). Если все условия ложны, то выполняется `оператор 0` (при условии, что он задан). При этом число ветвей `N` неограниченно, а последней ветви (`Else`) может и не быть.

В качестве примера на данную тему рассмотрим такую задачу: необходимо по трем введенным с клавиатуры числам принять решение:

- если среди данных чисел имеется одно, которое превосходит два других, то его необходимо вывести на экран;
- если же нет одного превосходящего два других, то об этом необходимо вывести сообщение на экран.

Листинг 2.7 содержит текст необходимой программы.

Листинг 2.7. Анализ трех введенных с клавиатуры чисел

```
Module Module1
    Sub Main()
        Dim Z, X, W  As Integer
        Console.Write("Введите X ")
        X = Console.ReadLine()
        Console.Write("Введите W  ")
        W = Console.ReadLine()
        Console.Write("Введите Z ")
        Z = Console.ReadLine()
        If X > Z And X > W Then
            Console.Write("X больше двух других чисел")
        ElseIf Z > X And Z > W Then
            Console.Write("Z больше двух других чисел")
        ElseIf W > X And W > Z Then
            Console.Write("W больше двух других чисел")
        Else
            Console.Write("Максимальное число не одно")
        End If
        Console.Read()
    End Sub
End Module
```

Оператор выбора

Для случаев, когда требуется осуществить выбор одного значения из конечного набора вариантов, оператор `If` удобнее заменить оператором *выбора (переключателем)* `Select Case`. Эта конструкция состоит из анализируемого выражения и набора операторов `Case` на каждое возможное значение выражения. Работает данная

конструкция следующим образом. Сначала проводится вычисление выражения указанного после `Select Case`. Затем полученное значение сравнивается со значениями, указанными в конструкциях `Case`. Если в одной из конструкций значения совпадают, то выполняются соответствующие операторы. Синтаксис конструкции оператора *выбора* следующий:

```
Select Case выражение
  Case значение 1
     операторы 1
  Case значение 2
     операторы 2
  . . .
  Case Else
     операторы
End Select
```

Если ни одно значение не совпало с анализируемым выражением, то выполняются операторы, расположенные в конструкции `Case Else`.

В листинге 2.8 приведен простой пример использования оператора выбора, с помощью которого организуется вывод текстового названия месяца в зависимости от числового обозначения месяца, вводимого с клавиатуры.

Листинг 2.8. Формирование названия месяца по его порядковому номеру

```
Module Module1
    Sub Main()
        Dim Z As Integer
        Console.Write("Введите номер месяца ")
        Z = Console.ReadLine()
        Select Case Z
            Case 1
                Console.Write("Январь")
            Case 2
                Console.Write("Февраль")
            Case 3
                Console.Write("Март")
            Case 4
                Console.Write("Апрель")
            Case 5
                Console.Write("Май")
            Case 6
                Console.Write("Июнь")
            Case 7
                Console.Write("Июль")
            Case 8
                Console.Write("Август")
            Case 9
                Console.Write("Сентябрь")
```

```
            Case 10
                Console.Write("Октябрь")
            Case 11
                Console.Write("Ноябрь")
            Case 12
                Console.Write("Декабрь")
            Case Else
                Console.Write("Номер месяца указан неправильно")
        End Select
        Console.Read()
    End Sub
End Module
```

Можно в конструкции `Case` указывать не одно значение, а несколько. В этом случае их необходимо перечислять через запятую.

Также организация списков `Case` может быть построена в виде диапазонов. Это удобнее, чем перечислять через запятую возможные значения, если таких достаточно много. Диапазоны указываются в виде:

<Минимальное значение> `To` *<Максимальное значение>*

В диапазон входят все значения от минимального до максимального включительно.

В качестве примера приведем программу формирования названия времени года в зависимости от числового номера месяца (листинг 2.9).

Листинг 2.9. Формирование времени года по номеру месяца

```
Module Module1
    Sub Main()
        Dim Z As Integer
        Console.Write("Введите номер месяца ")
        Z = Console.ReadLine()
        Select Case Z
            Case 1, 2, 12
                Console.Write("Зима")
            Case 3 To 5
                Console.Write("Весна")
            Case 6 To 8
                Console.Write("Лето")
            Case 9 To 11
                Console.Write("Осень")
            Case Else
                Console.Write("Номер месяца указан неправильно")
        End Select
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще один пример на тему оператора выбора. Необходимо по введенному с клавиатуры номеру тарифа и количеству оплачиваемых месяцев выдать значение суммы для оплаты. При этом будем считать, что тарифа три, и соответствие номеров тарифов и оплаты за месяц выглядит следующим образом:

- для тарифа 1 сумма оплаты за месяц 12000;
- для тарифа 2 сумма оплаты за месяц 10000;
- для тарифа 3 сумма оплаты за месяц 9000.

Листинг 2.10 содержит текст данной программы.

Листинг 2.10. Анализ тарифа и вычисление суммы для оплаты

```
Module Module1
    Sub Main()
        Dim M, N As Integer
        Dim Sum, Tar As Integer
        Console.Write("Ввести число оплачиваемых месяцев ")
        M = Console.ReadLine()
        Console.Write("Ввести номер тарифа ")
        N = Console.ReadLine()
        Select Case N
            Case 1
                Tar = 12000
            Case 2
                Tar = 10000
            Case 3
                Tar = 9000
            Case Else
                Tar = 0
                Console.Write("Номер тарифа указан неправильно")
        End Select
        If Tar > 0 Then
            Sum = Tar * M
            Console.Write("К оплате ")
            Console.Write(Sum)
        End If
        Console.Read()
    End Sub
End Module
```

Оператор цикла *For*

В программе часто требуется повторять определенные действия. Такое повторение называется *циклом*. А сама последовательность выполняемых команд представляет собой *тело цикла*. Наиболее простая и в то же время часто встречающаяся ситуация

связана со случаем, когда число повторений цикла известно заранее. На языке Бейсик для реализации такого алгоритма действий используется оператор `For`.

Подсчет количества выполняемых действий осуществляется при помощи специальной переменной — *счетчика*. Наиболее простая (и часто используемая) форма цикла `For` выглядит так:

```
For счетчик=A To B
   Операторы
Next
```

В этом случае выполняется последовательное увеличение (на единицу) значения счетчика. При этом `A` — начальное значение счетчика, а `B` — его конечное значение. После очередного выполнения тела цикла (ключевое слово `Next` определяет границу цикла) значение счетчика увеличивается на единицу. Далее это новое значение сравнивается с предельным (конечным) значением. Если счетчик не вышел за конечное значение, то цикл повторяется опять (выполняются операторы между конструкциями `For` и `Next`). Таких проходов цикла может быть сколь угодно много. Но, как только счетчик превысит конечное значение, выполнение цикла прекращается и будет выполнен оператор, следующий за `Next`.

Более общая форма оператора `For` включает возможность установки *шага цикла*:

```
For счетчик=A To B Step I
   Операторы
Next
```

В этом случае после ключевого слова `Step` указывается значение приращения счетчика цикла.

Рассмотрим сначала пример, связанный с последовательным увеличением счетчика на единицу. Наша задача заключается в том, чтобы обеспечить ввод с клавиатуры десяти оценок по определенной дисциплине. После этого программа должна подсчитать средний балл по введенным оценкам и вывести его на экран. Текст разработки приведен в листинге 2.11. Здесь переменная `J` является счетчиком цикла.

Результат работы такой программы приведен на рис. 2.1.

Листинг 2.11. Расчет среднего балла по дисциплине

```
Module Module1
    Sub Main()
        Dim A, J As Integer
        Dim Srednee As Single
        Srednee = 0.0
        For J = 1 To 10
            Console.Write("Введите оценку ")
            A = Console.ReadLine()
            Srednee = Srednee + A
        Next
        Srednee = Srednee / 10
```

```
        Console.Write("Средняя оценка ")
        Console.Write(Srednee)
        Console.Read()
    End Sub
End Module
```

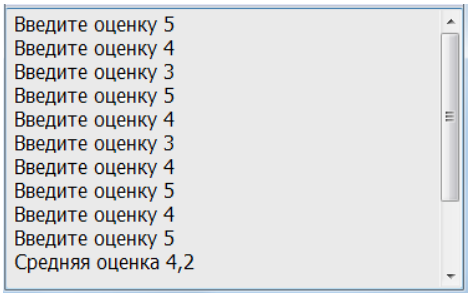

Рис. 2.1. Вывод информации в программе, приведенной в листинге 2.11

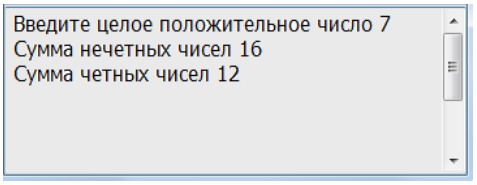

Рис. 2.2. Вывод информации в программе, приведенной в листинге 2.12

Рассмотрим теперь пример программы, которая вычисляет отдельно сумму четных чисел и сумму нечетных чисел в исходной последовательности целых чисел от 1 до N. Значение N вводится с клавиатуры, а вычисленные суммы чисел выводятся на экран. В листинге 2.12 приведен текст данной программы, а результат ее работы показан на рис. 2.2.

Листинг 2.12. Расчет сумм четных и нечетных чисел

```
Module Module1
    Sub Main()
        Dim N, J As Integer
        Dim Summa1, Summa2 As Integer
        Console.Write("Введите целое положительное число ")
        N = Console.ReadLine()
        Summa1 = 0
        Summa2 = 0
        For J = 1 To N
            If (J Mod 2) = 1 Then
                Summa1 = Summa1 + J
            Else
                Summa2 = Summa2 + J
            End If
        Next
        Console.Write("Сумма нечетных чисел ")
        Console.WriteLine(Summa1)
        Console.Write("Сумма четных чисел ")
        Console.WriteLine(Summa2)
```

```
        Console.Read()
    End Sub
End Module
```

Рассмотрим теперь построение следующего алгоритма: требуется по введенному с клавиатуры целому положительному числу выдать заключение — является ли оно простым.

> ***Примечание***
>
> *Простыми* считаются целые положительные числа, которые делятся без остатка только на себя и на единицу.

Идея алгоритма заключается в анализе ситуаций, когда остаток от деления исходного числа N на числа в интервале от 1 до N равен 0. Если таких случаев будет больше двух, то число не простое. В противном случае число является простым. Текст необходимой программы для решения данной задачи представлен в листинге 2.13. Результат работы программы показан на рис. 2.3.

Листинг 2.13. Проверка, является ли введенное число простым

```
Module Module1
    Sub Main()
        Dim N, J As Integer
        Dim Summa As Integer
        Console.Write("Введите целое положительное число ")
        N = Console.ReadLine()
        Summa = 0
        For J = 1 To N
            If (N Mod J) = 0 Then
                Summa = Summa + 1
            End If
        Next
        If Summa > 2 Then
            Console.Write("Число не простое")
        Else
            Console.Write("Число простое")
        End If
        Console.Read()
    End Sub
End Module
```

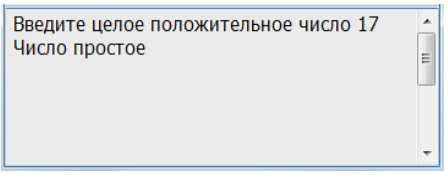

Рис. 2.3. Вывод информации в программе, приведенной в листинге 2.13

Однако рассмотренный алгоритм не является оптимальным с точки зрения вычислительных действий. Очевидно, что проверку "является ли число простым" можно организовать более эффективным способом. А именно: если мы найдем в интервале от 2 до N – 1 хотя бы одно число, на которое N делится без остатка, то из этого уже следует, что N не является простым.

Для реализации этой идеи в виде программной разработки нам потребуется новый оператор. Это — `Exit For`, прерывающий выполнение цикла и передающий управление оператору, который должен выполняться после окончания цикла. В листинге 2.14 показана программа, которая делает рассмотренную проверку чисел более эффективной. Мы ввели переменную `Flag`, которая играет роль индикатора. В начале программы этой переменной присваивается значение ноль. Если в процессе анализа находится значение, на которое исходное число делится без остатка, то происходит установка переменной `Flag` в единицу и цикл завершается. В заключительной части программы в зависимости от значения `Flag` выполняется вывод соответствующего сообщения.

Листинг 2.14. Более эффективный алгоритм анализа простых чисел

```
Module Module1
    Sub Main()
        Dim N, J As Integer
        Dim Flag As Integer
        Console.Write("Введите целое положительное число ")
        N = Console.ReadLine()
        Flag = 0
        For J = 2 To N — 1
            If (N Mod J) = 0 Then
                Flag = 1
                Exit For
            End If
        Next
        If Flag = 1 Then
            Console.Write("Число не простое")
        Else
            Console.Write("Число простое ")
        End If
        Console.Read()
    End Sub
End Module
```

Все рассмотренные примеры касались использования счетчика цикла на увеличение. Однако, как мы уже говорили, в языке Бейсик имеется вариант оператора `For` с произвольным шагом. Например, можно организовать уменьшение счетчика при организации цикла. Рассмотрим пример на эту тему.

Задана числовая последовательность Ne^N, где `N` является целым и изменяется от 1 до 50. Необходимо найти такое максимальное `N`, при котором значение элемента

этой последовательности меньше 10000. В данном случае мы организуем цикл в направлении уменьшения `N` от 50 до 1. При нахождении искомого числа цикл завершается.

В листинге 2.15 приведена программа, которая решает поставленную задачу. В результате ее выполнения мы увидим на экране число 7.

Листинг 2.15. Пример организации счетчика на уменьшение

```
Module Module1
    Sub Main()
        Dim N, Nmin As Integer
        Dim F As Single
        For N = 50 To 1 Step -1
            F = Math.Exp(N) * N
            If F < 10000 Then
                Nmin = N
                Exit For
            End If
        Next
        Console.Write(Nmin)
        Console.Read()
    End Sub
End Module
```

Цикл с предусловием

Рассмотренный цикл `For` выполняет необходимую функциональность, когда число повторений тела цикла известно к моменту его начала. Однако часто приходится решать задачи, когда число повторений цикла заранее неизвестно. В ряде ситуаций это значение определяется по мере выполнения вычислительных действий, и тогда применяют другую разновидность цикла — *цикл с условием*. В языке Бейсик предусмотрено два цикла с условием:

- условие цикла проверяется перед циклом *(цикл с предусловием)*;
- условие цикла проверяется после цикла *(цикл с постусловием)*.

Рассмотрим сначала организацию цикла с предусловием. На языке Бейсик данная схема реализуется с использованием следующей синтаксической конструкции:

```
While логическое условие
   тело цикла
End While
```

Здесь, пока истинно логическое условие, выполняется тело цикла.

Рассмотрим практический пример на эту тему. Так, используя цикл с предусловием, решим задачу, которая связана с вычислением факториала.

Примечание

Факториалом целого положительного числа N называется произведение всех целых чисел от 1 до N включительно.

Условие задания таково: необходимо найти наименьшее целое положительное число, факториал которого не меньше 10^{15}. Программное решение данной задачи приведено в листинге 2.16.

Результат работы программы показан на рис. 2.4.

Листинг 2.16. Пример использования цикла с предусловием

```
Module Module1
    Sub Main()
        Const A = 1.0E+15
        Dim N As Integer
        Dim F As Single
        N = 1
        F = 1
        While F < A
            N = N + 1
            F = F * N
        End While
        Console.WriteLine(N)
        Console.Write(F)
        Console.Read()
    End Sub
End Module
```

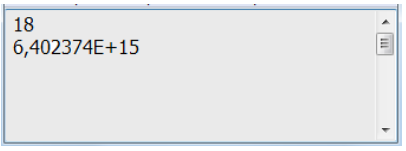

Рис. 2.4. Вывод информации в программе, приведенной в листинге 2.16

Цикл с постусловием

В этом случае условие проверяется после цикла (в конце цикла). Цикл повторяется до тех пор, пока проверка указанного условия будет давать результат `True`, т. е. пока условие выполняется. Важно отметить, что если условие сразу окажется ложным, то цикл все равно выполняется один раз.

Для цикла с постусловием сначала выполняется тело цикла, а затем управление передается на проверку условия. В зависимости от истинности или ложности условия тело цикла выполняется повторно или же происходит переход к оператору, следующему за телом цикла.

Основное различие двух рассмотренных вариантов циклов с условием: цикл с постусловием гарантированно *выполняется хотя бы раз*, а цикл с предусловием может быть не выполнен ни разу, если условие сразу же окажется ложным.

На языке Бейсик данный тип цикла реализуется с помощью следующей синтаксической конструкции:

```
Do
  тело цикла
Loop While условие
```

Рассмотрим пример, в котором пользователю предоставляется возможность ввода с клавиатуры целых чисел и их суммирования. При вводе нуля суммирование заканчивается и на экране отображается результат. Текст необходимой программы приведен в листинге 2.17. Результат работы программы показан на рис. 2.5.

Листинг 2.17. Пример использования цикла с постусловием

```
Module Module1
    Sub Main()
        Dim A, Summa As Single
        Summa = 0
        Do
            Console.WriteLine("Введите число")
            A = Console.ReadLine
            Summa = Summa + A
        Loop While A <> 0
        Console.Write ("Сумма равна ")
        Console.Write(Summa)
        Console.Read()
    End Sub
End Module
```

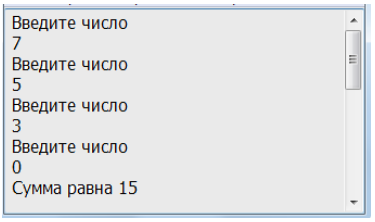

Рис. 2.5. Вывод информации в программе, приведенной в листинге 2.17

Метки

В большинстве случаев алгоритмы требуют организации программных переходов. Такие переходы можно реализовать и с использованием уже рассмотренных ранее ресурсов (условий и циклов). Однако в Бейсике имеется *оператор безусловного*

перехода, который весьма полезен на начальном этапе изучения программирования. Синтаксис оператора выглядит так:

```
Goto метка
```

В этом операторе ключевую роль играет *метка* — произвольный идентификатор, позволяющий именовать определенный оператор в программе. После метки необходимо поставить двоеточие.

В листинге 2.18 приведен текст программы, которая позволяет вычислить произведение чисел, вводимых с клавиатуры. При вводе нуля в качестве очередного числа программа завершает работу, а результат выводится на экран.

Результат работы программы показан на рис. 2.6.

Листинг 2.18. Пример использования меток в программе

```
Module Module1
    Sub Main()
        Dim A, Multi As Single
        Multi = 1
M1:     Console.WriteLine("Введите число")
        A = Console.ReadLine()
        If A = 0 Then GoTo M2
        Multi = Multi * A
        GoTo M1
M2:     Console.Write("Результат ")
        Console.WriteLine(Multi)
        Console.Read()
    End Sub
End Module
```

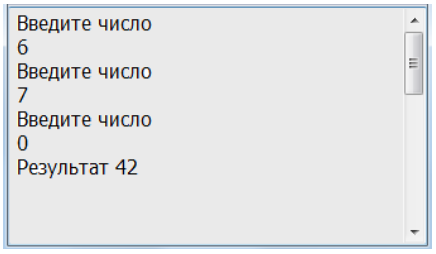

Рис. 2.6. Вывод информации в программе, приведенной в листинге 2.18

Работа с символьными строками

Мы уже упоминали о том, что для работы с цепочками символов в Visual Basic имеется специальный тип данных — `String`. В этом разделе мы рассмотрим несколько практических примеров на тему работы со строковыми данными.

В Бейсике возможны следующие действия над строками:

- ввод/вывод;
- сложение;
- присваивание;
- сравнение.

В листинге 2.19 приведен простой пример использования сложения, ввода/вывода и присваивания при работе со строковыми переменными. Результат работы программы показан на рис. 2.7.

Листинг 2.19. Пример работы со строками

```
Module Module1
    Sub Main()
        Dim A, B, C, D As String
        Console.WriteLine("Введите имя")
        B = Console.ReadLine()
        A = "Добрый день,"
        C = "!"
        D = A + B + C
        Console.WriteLine(D)
        Console.Read()
    End Sub
End Module
```

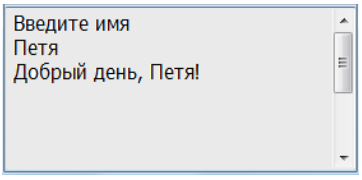

Рис. 2.7. Вывод информации в программе, приведенной в листинге 2.19

Довольно распространенное действие связано со сравнением строк. Здесь никакой новизны по сравнению с ранее рассмотренными примерами нет. Например, для сравнения двух переменных строкового типа достаточно использовать следующий фрагмент программного кода:

```
If Str1 = Str2 Then
```

Рассмотрим пример на данную тему. Так, с клавиатуры вводятся логин и пароль — набор символов, представленных в виде двух отдельных строк. Необходимо проверить, соответствуют ли они истинным значениям. Программа, решающая данную задачу, приведена в листинге 2.20. В данном тексте мы разместили две переменные: `Aist` и `Bist`, предназначенные для истинных значений пароля и логина.

Листинг 2.20. Проверка логина и пароля

```
Module Module1
    Sub Main()
        Dim A, B, Aist, Bist As String
        Aist = "ZZZ"
        Bist = "123"
        Console.WriteLine("Введите логин")
        A = Console.ReadLine()
        Console.WriteLine("Введите пароль")
        B = Console.ReadLine()
        If A <> Aist Or B <> Bist Then
            Console.WriteLine("Пароль или логин введены с ошибкой")
        Else
            Console.WriteLine("Авторизация выполнена!")
        End If
        Console.Read()
    End Sub
End Module
```

Рассмотрим пример использования функции работы со строками `Mid`, которая предназначена для выделения фрагмента из исходной строки. Данная функция имеет три параметра:

- первый определяет исходную строку;
- второй определяет номер символа, начиная с которого выделяется необходимый фрагмент;
- третий параметр определяет количество выделяемых символов.

В листинге 2.21 приведен пример, в котором из строки, содержащей фамилию и имя, выделяется только имя. Для этого мы указали, что необходимо выделить 4 символа, начиная с 8-й позиции.

Листинг 2.21. Выделение части строки и вывод ее на экран

```
Module Module1
    Sub Main()
        Dim A, B As String
        A = "Иванов Петр"
        B = Mid(A, 8, 4)
        Console.WriteLine(B)
        Console.Read()
    End Sub
End Module
```

Типовые примеры и задания из ЕГЭ

Далее мы разберем ряд практических примеров, которые можно отнести к использованию уже рассмотренных конструкций. Некоторые из приведенных примеров встречались в билетах Единого государственного экзамена в прошлые годы.

Подсчет суммы цифр в числе

Требуется разработать алгоритм подсчета суммы цифр в целом положительном числе, представленном в десятичной системе счисления. Например, в числе 3512_{10} сумма цифр равна 11_{10}, что определяется с помощью простого суммирования. Нам необходимо построить алгоритм, который будет автоматически выполнять указанное действие.

> ***Примечание***
>
> В *главе 1* мы рассмотрели подобные задачи, но в них каждый раз ограничивалась разрядность числа. Здесь же мы не будем использовать это ограничение.

Вместе с разработкой алгоритма требуется написать программу, которая должна выполнять данную процедуру с целым числом, вводимым с клавиатуры.

Идея решения задачи заключается в использовании операции целочисленного деления и вычисления остатка от целочисленного деления. Так, если мы возьмем остаток деления исходного числа на 10, то получим самую младшую цифру исходного числа. Далее следует исходное число разделить на 10 и опять вычислить остаток от деления полученного результата на 10. В результате мы получим цифру, расположенную в разряде десятков исходного числа. Этот процесс следует продолжать до тех пор, пока результатом деления на 10 не окажется ноль. Теперь алгоритм можно реализовать в виде программы (листинг 2.22). Здесь мы воспользовались знакомым циклом с предусловием. Результат работы программы показан на рис. 2.8.

Листинг 2.22. Программа подсчета суммы цифр в десятичном числе

```
Module Module1
    Sub Main()
        Dim N, A, Summa As Integer
        Console.WriteLine("Введите целое число")
        N = Console.ReadLine()
        Summa = 0
        While N > 0
            A = N Mod 10
            Summa = Summa + A
            N = N \ 10
        End While
        Console.WriteLine(Summa)
        Console.Read()
    End Sub
End Module
```

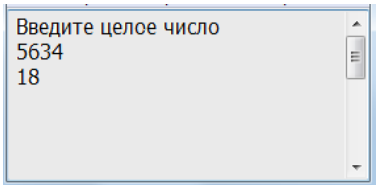

Рис. 2.8. Вывод информации в программе, приведенной в листинге 2.22

Анализ четности пары чисел

Требуется написать программу, которая определяет, имеется ли среди введенных с клавиатуры целых чисел `A` и `B` хотя бы одно четное. Вариант реализации требуемого алгоритма представлен в листинге 2.23. Здесь мы сначала вычисляем остатки от деления двух исходных чисел на 2. Если оба этих остатка равны единице, то, следовательно, среди введенных чисел четных нет. В противном случае имеется хотя бы одно четное число.

Листинг 2.23. Проверка наличия четных чисел

```
Module Module1
    Sub Main()
        Dim A, B As Integer
        Console.WriteLine("Введите первое целое число")
        A = Console.ReadLine()
        Console.WriteLine("Введите второе целое число")
        B = Console.ReadLine()
        A = A Mod 2
        B = B Mod 2
        A = A + B
        If A = 2 Then
            Console.WriteLine("Четных чисел нет")
        Else
            Console.WriteLine("Четные числа есть")
        End If
        Console.Read()
    End Sub
End Module
```

Реализуем еще один вариант решения, который связан с разработкой программы, использующей операцию `And`. В этом случае (листинг 2.24) вместо арифметического сложения остатков от деления используется операция — `And`. А именно: вычисляются условие равенства единице остатков от деления исходных чисел на 2. Если оба остатка от деления равны единице, то на экран будет выведено соответствующее сообщение об отсутствии четных чисел.

Листинг 2.24. Вариант алгоритма анализа чисел с использованием `And`

```
Module Module1
    Sub Main()
        Dim A, B As Integer
        Console.WriteLine("Введите первое целое число")
        A = Console.ReadLine()
        Console.WriteLine("Введите второе целое число")
        B = Console.ReadLine()
        A = A Mod 2
        B = B Mod 2
        If A = 1 And B = 1 Then
            Console.WriteLine("Четных чисел нет")
        Else
            Console.WriteLine("Четные числа есть")
        End If
        Console.Read()
    End Sub
End Module
```

Построение треугольников из отрезков

Требуется написать программу, которая определяет, можно ли построить треугольник из отрезков с длинами `X`, `Y`, `Z`. Программа должна выводить на экран соответствующее текстовое сообщение.

Идея алгоритма заключается в том, что в треугольнике сумма каждой пары сторон должна быть больше третьей стороны, и программно мы должны сравнить сумму каждой пары имеющихся отрезков с третьим отрезком. Если в каком-то случае (из трех) длина одного отрезка будет превышать сумму длин двух других, то, следовательно, и треугольник построить нельзя.

Существуют несколько вариантов написания программы в соответствии с данными условиями. Один из примеров правильной программы представлен в листинге 2.25. Здесь мы ввели переменную `Flag`, в которую, в случае возможности построения треугольника, заносится значение 1. При этом в начале программы значение этой переменной устанавливается равным нулю.

Результаты работы программы для различных исходных данных показаны на рис. 2.9 и 2.10.

Листинг 2.25. Проверка построения треугольника из отрезков

```
Module Module1
    Sub Main()
        Dim X, Y, Z As Single
        Dim Flag As Integer
        Console.WriteLine("Введите длину первого отрезка ")
```

```
        X = Console.ReadLine()
        Console.WriteLine("Введите длину второго отрезка ")
        Y = Console.ReadLine()
        Console.WriteLine("Введите длину третьего отрезка ")
        Z = Console.ReadLine()
        Flag = 0
        If (X + Y) > Z Then
            If (X + Z) > Y Then
                If (Y + Z) > X Then
                    Flag = 1
                End If
            End If
        End If
        If Flag = 1 Then
            Console.WriteLine("Треугольник построить можно")
        Else
            Console.WriteLine("Треугольник построить нельзя")
        End If
        Console.Read()
    End Sub
End Module
```

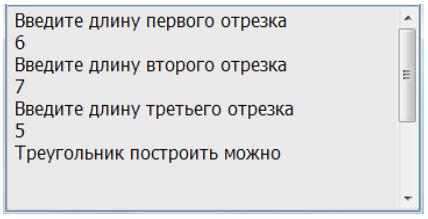

Рис. 2.9. Вывод информации в программе, приведенной в листинге 2.25, при условии ввода данных, с которыми треугольник построить можно

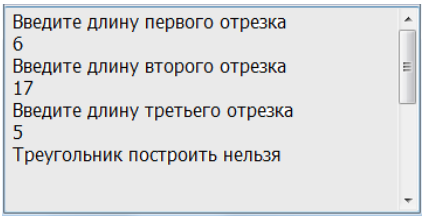

Рис. 2.10. Вывод информации в программе, приведенной в листинге 2.25, при условии ввода данных, с которыми треугольник построить нельзя

Можно привести еще один вариант алгоритма, в котором используется оператор `And`. Листинг 2.26 содержит видоизмененный текст программы. В этом случае программный код получается заметно проще, и фактически все сводится к проверке сложного условия.

Листинг 2.26. Проверка построения треугольника из отрезков (с использованием оператора *And*)

```
Module Module1
    Sub Main()
        Dim X, Y, Z As Single
        Console.WriteLine("Введите длину первого отрезка ")
        X = Console.ReadLine()
```

```
        Console.WriteLine("Введите длину второго отрезка ")
        Y = Console.ReadLine()
        Console.WriteLine("Введите длину третьего отрезка ")
        Z = Console.ReadLine()
        If ((X + Y) > Z) And ((X + Z) > Y) And ((Y + Z) > X) Then
            Console.WriteLine("Треугольник построить можно")
        Else
            Console.WriteLine("Треугольник построить нельзя")
        End If
        Console.Read()
    End Sub
End Module
```

Перевод числа в шестнадцатеричную систему

Необходимо разработать программу, которая после ввода с клавиатуры целого десятичного числа в интервале от 0 до 15 позволит вывести на экран его эквивалент в шестнадцатеричной системе счисления. Один из примеров программы, решающей данную задачу, представлен в листинге 2.27. Здесь учитывается, что в шестнадцатеричной системе счисления знаки, обозначающие цифры, плотно расположены в двух группах:

- от 0 до 9;
- от A до F.

При этом коды символов в каждой из этих групп тоже расположены "плотно". Если число меньше 10, то его символ формируется из кода нуля и добавления самого числа. Если же введенное число больше 9, то в качестве "базы" следует взять латинскую букву "A". И тогда сумма кода буквы "A" и разности между числом и 10 представляет код символа шестнадцатеричной цифры.

Листинг 2.27. Перевод числа в шестнадцатеричную систему

```
Module Module1
    Sub Main()
        Dim Ch As Char
        Dim N As Integer
        Console.WriteLine("Введите  число")
        N = Console.ReadLine()
        If (N >= 0) And (N < 10) Then
            Ch = Chr(Asc("0") + N)
            Console.WriteLine(Ch)
        ElseIf (N >= 10) And (N < 16) Then
            Ch = Chr(Asc("A") + N — 10)
            Console.WriteLine(Ch)
        End If
```

```
        Console.Read()
    End Sub
End Module
```

Подсчет по условию

Требуется написать программу, которая бы позволяла вводить оценки учащихся с клавиатуры и подсчитывала бы отдельно количество сдавших дисциплину (сдавшими считаются те ученики, которые получили оценки не менее тройки) и не сдавших дисциплину (тех, кто получил двойки).

При этом программа должна обеспечить последовательный ввод оценок учащихся. Признаком завершения данных считается ввод очередного числа вне целочисленного интервала от двух до пяти. Один из вариантов программы, решающей поставленную задачу, представлен в листинге 2.28.

Результаты работы программы показаны на рис. 2.11.

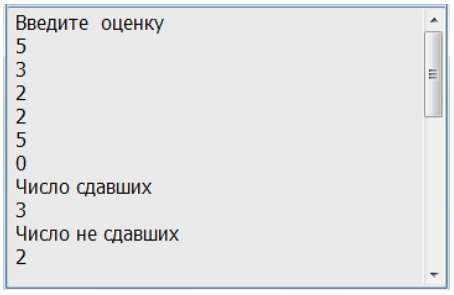

Рис. 2.11. Вывод информации в программе, приведенной в листинге 2.28

Листинг 2.28. Подсчет числа сдавших и числа не сдавших экзамен

```
Module Module1
    Sub Main()
        Dim N, M, Otsenka As Integer
        N = 0
        M = 0
        Console.WriteLine("Введите  оценку")
        Otsenka = Console.ReadLine()
        While Otsenka >= 2 And Otsenka <= 5
            If Otsenka = 2 Then
                M = M + 1
            Else
                N = N + 1
            End If
            Otsenka = Console.ReadLine()
        End While
```

```
        Console.WriteLine("Число сдавших ")
        Console.WriteLine(N)
        Console.WriteLine("Число не сдавших ")
        Console.WriteLine(M)
        Console.Read()
    End Sub
End Module
```

Возможность построения прямоугольного треугольника

Необходимо проверить, может ли быть построен прямоугольный треугольник по длинам сторон A, B, C при условии, что A является гипотенузой. Мы воспользуемся теоремой Пифагора, при этом необходимо задать разумную точность сопоставления сторон треугольника. Программа, решающая поставленную задачу, представлена в листинге 2.29.

Результаты работы программы показаны на рис. 2.12.

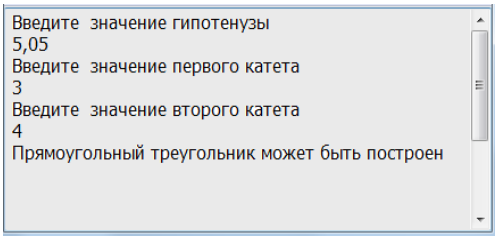

Рис. 2.12. Вывод информации в программе, приведенной в листинге 2.29

Листинг 2.29. Проверка возможности построения треугольника

```
Module Module1
  Sub Main()
   Dim A, B, C, D As Single
   Console.WriteLine("Введите  значение гипотенузы")
   A = Console.ReadLine()
   Console.WriteLine("Введите  значение первого катета")
   B = Console.ReadLine()
   Console.WriteLine("Введите  значение второго катета")
   C = Console.ReadLine()
   D = Math.Sqrt(C ^ 2 + B ^ 2)
   D = D — A
   If D < 0 Then D = -D
   If D < 0.1 Then
    Console.WriteLine("Прямоугольный треугольник может быть построен")
```

```
   Else
    Console.WriteLine("Прямоугольный треугольник не может быть построен")
   End If
    Console.Read()
End Sub
End Module
```

Представление слова с учетом падежа

С клавиатуры вводится денежная сумма в рублях в числовом формате. Программа должна напечатать введенную сумму с правильной формой падежа слова "рубль". Например, "23 рубля" или "51 рубль".

Понятно, что окончание, используемое для слова "рубль", зависит от последних цифр суммы. Самую последнюю цифру можно получить, если взять остаток от деления на 10. Однако если в сумме последние две цифры расположены в интервале от 11 до 14 включительно, то окончание меняется. Используя эту информацию, составим следующую программу (листинг 2.30).

Результаты работы программы показаны на рис. 2.13.

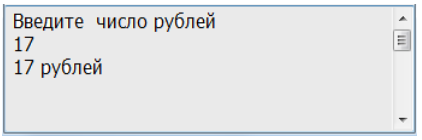

Рис. 2.13. Вывод информации в программе, приведенной в листинге 2.30

Листинг 2.30. Написание слова "рубль" с учетом падежа

```
Module Module1
  Sub Main()
    Dim A, B, S As Integer
     Console.WriteLine("Введите  число рублей")
     S = Console.ReadLine()
     A = S Mod 10
     B = S Mod 100
     Console.Write(S)
     Console.Write(" ")
     If (B=11) Or (B=12) Or (B = 13) Or (B = 14) Or _
            (A > 4) Or (A = 0) Then
            Console.Write("рублей")
        ElseIf A = 1 Then
            Console.Write("рубль")
        Else
            Console.Write("рубля")
        End If
```

```
        Console.Read()
    End Sub
End Module
```

Формирование таблицы стоимости товаров

Известна стоимость единицы товара. Необходимо составить таблицу стоимости 1, 2, 3, ..., `K` единиц товара, при этом значение `K` вводится с клавиатуры. Так как число единиц товара целое, то для него выберем тип `Integer`. Для цены, соответственно, определим дробный тип `Decimal`. Необходимая программа приведена в листинге 2.31.

Результаты работы программы показаны на рис. 2.14.

Листинг 2.31. Формирование таблицы стоимости товаров

```
Module Module1
    Sub Main()
        Dim Tsena As Decimal
        Dim I, K As Integer
        Console.WriteLine("Стоимость единицы товара")
        Tsena = Console.ReadLine()
        Console.WriteLine("Количество единиц товара")
        K = Console.ReadLine()
        For I = 1 To K
            Console.Write(I)
            Console.Write("       ")
            Console.WriteLine(Tsena * I)
        Next
        Console.Read()
    End Sub
End Module
```

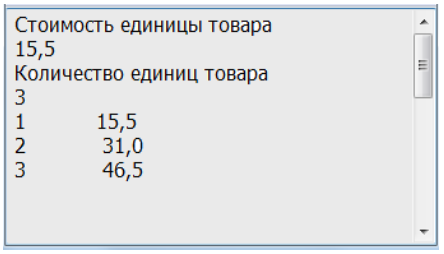

Рис. 2.14. Вывод информации в программе, приведенной в листинге 2.31

Поиск чисел

Требуется найти (программно) все двузначные числа, в записи которых имеется цифра 3 и одновременно само число делится на 3. Здесь мы воспользуемся операторами цикла и условия (листинг 2.32).

Результаты работы программы показаны на рис. 2.15.

Листинг 2.32. Поиск чисел по условию

```
Module Module1
  Sub Main()
    Dim I As Integer
    For I = 10 To 99
       If I Mod 3 = 0 And (I Mod 10 = 3 Or I \ 10 = 3) Then
           Console.WriteLine(I)
       End If
    Next
    Console.Read()
 End Sub
End Module
```

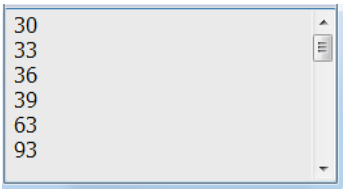

Рис. 2.15. Вывод информации в программе, приведенной в листинге 2.32

Еще один похожий пример. Требуется найти все целые числа в интервале от 100 до 999, сумма цифр которых больше произведения этих цифр. Необходимая для этого программная разработка приведена в листинге 2.33.

Листинг 2.33. Поиск чисел по условию

```
Module Module1
    Sub Main()
        Dim A, B, C, X, Z, I As Integer
        For I = 100 To 999
            A = I Mod 10
            B = I / 100
            C = (I Mod 100) / 10
            X = A * B * C
            Z = A + B + C
            If Z > X Then
                Console.Write(I)
```

```
                Console.Write(" ")
            End If
        Next
        Console.Read()
    End Sub
End Module
```

Анализ чисел

С клавиатуры вводится четырехзначное число. Необходимо программно с использованием оператора цикла определить, имеются ли в нем четные цифры. Разработка представлена в листинге 2.34. Здесь мы последовательно выделяем цифры из исходного числа. Для этого сначала используется остаток от деления исходного числа на 10. В результате мы получим цифру в разряде единиц. Проанализировать ее на четность можно с помощью вычисления остатка от ее деления на 2. После этого мы делим исходное число на 10 и аналогичным образом проверяем на четность цифру в разряде десятков. Это действие выполняется еще два раза для того, чтобы проанализировать цифры в разряде сотен и тысяч.

Листинг 2.34. Проверка четырехзначного числа на наличие четных цифр

```
Module Module1
    Sub Main()
        Dim A, B, Flag, I As Integer
        Console.Write("Введите четырехзначное число")
        A = Console.ReadLine()
        Flag = 0
        For I = 1 To 4
            B = A Mod 10
            If B Mod 2 = 0 Then
                Flag = 1
                Exit For
            End If
            A = A \ 10
        Next
        If Flag = 1 Then
            Console.Write(" В числе есть четные цифры")
        Else
            Console.Write(" В числе нет четных цифр")
        End If
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще один пример на тему анализа чисел. Как и в предыдущей разработке, с клавиатуры вводится четырехзначное число. Необходимо программно

с использованием оператора цикла определить, имеются ли в нем соседние одинаковые цифры (одинаковые цифры в соседних разрядах). Разработка представлена в листинге 2.35.

Листинг 2.35. Анализ числа на наличие соседних одинаковых цифр

```
Module Module1
    Sub Main()
        Dim A, B, C, Flag, I As Integer
        Console.Write("Введите четырехзначное число")
        A = Console.ReadLine()
        Flag = 0
        For I = 1 To 3
            B = A Mod 10
            C = (A Mod 100) \ 10
            If B = C Then
                Flag = 1
                Exit For
            End If
            A = A \ 10
        Next
        If Flag = 1 Then
            Console.Write(" В числе есть соседние одинаковые цифры")
        Else
            Console.Write(" В числе нет соседних одинаковых цифр")
        End If
        Console.Read()
    End Sub
End Module
```

В следующем примере с клавиатуры вводится шестизначное число. Необходимо определить, является ли оно "счастливым". Напомним, что "счастливыми" считаются числа, у которых сумма первых трех цифр равна сумме следующих трех цифр. Разработка представлена в листинге 2.36.

Листинг 2.36. Проверка на счастливое число

```
Module Module1
    Sub Main()
        Dim A, B, C, I As Integer
        Console.Write("Введите шестизначное число")
        A = Console.ReadLine()
        B = 0
        For I = 1 To 3
            B = B + (A Mod 10)
            A = A \ 10
        Next
```

```
        C = 0
        For I = 1 To 3
            C = C + (A Mod 10)
            A = A \ 10
        Next
        If B = C Then
            Console.Write(" Число счастливое")
        Else
            Console.Write(" Число не является счастливым")
        End If
        Console.Read()
    End Sub
End Module
```

В следующем примере с клавиатуры вводится целое число (для определенности содержащее не более 7 знаков). Необходимо определить максимальную цифру в этом числе. Разработка представлена в листинге 2.37.

Листинг 2.37. Поиск максимальной цифры в числе

```
Module Module1
    Sub Main()
        Dim A, B, I As Integer
        Console.Write("Введите  число")
        A = Console.ReadLine()
        B = 0
        I = A
        While I > 0
            If I Mod 10 > B Then B = I Mod 10
            I = I \ 10
        End While
        Console.WriteLine(" Максимальная цифра в числе")
        Console.Write(B)
        Console.Read()
    End Sub
End Module
```

С клавиатуры вводится целое число (для определенности не более 7 знаков и не менее двух). Необходимо определить две максимальные (различные) цифры в этом числе. Например, если введено число 567374, то две максимальные цифры — это 7 и 6. Разработка представлена в листинге 2.38.

Листинг 2.38. Поиск двух максимальных цифр в числе

```
Module Module1
    Sub Main()
        Dim A, B, C, I As Integer
```

```
        Console.Write("Введите  число")
        A = Console.ReadLine()
        B = 0
        I = A
        While I > 0
            If (I Mod 10) > B Then B = I Mod 10
            I = I \ 10
        End While
        C = 0
        I = A
        While I > 0
            If (I Mod 10) > C And (I Mod 10) <> B Then
                C = I Mod 10
            End If
            I = I \ 10
        End While
        Console.WriteLine(" Максимальные цифры в числе")
        Console.Write(B)
        Console.Write(C)
        Console.Read()
    End Sub
End Module
```

С клавиатуры вводится целое число (не более 7 знаков и не менее двух). Необходимо определить минимальную цифру в этом числе и подсчитать, сколько раз она встречается. Например, если введено число 527272, то в нем минимальная цифра 2 встречается три раза. Разработка представлена в листинге 2.39. Для поиска цифры мы отвели переменную `B`, а количество повторений этой цифры будет фиксироваться в переменной `C`.

Листинг 2.39. Поиск минимальной цифры

```
Module Module1
    Sub Main()
        Dim A, B, C, I As Integer
        Console.Write("Введите  число")
        A = Console.ReadLine()
        B = 9
        I = A
        While I > 0
            If (I Mod 10) < B Then
                B = I Mod 10
            End If
            I = I \ 10
        End While
        C = 0
        I = A
```

```
        While I > 0
            If (I Mod 10) = B Then
                C = C + 1
            End If
            I = I \ 10
        End While
        Console.WriteLine(" Минимальная цифра в числе встречается ")
        Console.Write(C)
        Console.WriteLine(" раз ")
        Console.Read()
    End Sub
End Module
```

Графики зависимостей

Необходимо построить график функции `Y(N) = sin(0.2*N)`, где `N` представляет собой последовательность целых чисел в интервале от 0 до 20. Разработка представлена в листинге 2.40. Принято, что одно деление по оси ординат (одно знакоместо на экране) соответствует 0.1, а взаимное расположение экрана и осей координат представлено на рис. 2.16. Ось ординат проходит параллельно левой границы экрана на расстоянии 20 символов. Для символа графика мы выбрали звездочку.

Рис. 2.16. Взаимное расположение экрана и осей координат

Листинг 2.40. График функции

```
Module Module1
    Sub Main()
        Dim N, M, I As Integer
        Dim Y As Single
        For N = 0 To 20
            Y = Math.Sin(0.2 * N)
            M = 20 + Math.Round(Y * 10)
```

```
            For I = 0 To M — 1
                Console.Write(" ")
            Next
            Console.WriteLine("*")
        Next
        Console.Read()
    End Sub
End Module
```

В следующем задании нам необходимо построить функции `y(N) = sin(0.2*N)` в виде гистограммы, где `N` представляет собой последовательность целых чисел в интервале от 0 до 20. Разработка приведена в листинге 2.41. Здесь вместо точки на графике мы выводим столбец звездочек, и чем больше значение функции, тем больше столбец, ему соответствующий.

Листинг 2.41. График функции в виде гистограммы

```
Module Module1
    Sub Main()
        Dim N, M, I As Integer
        Dim Y As Single
        For N = 0 To 20
            Y = Math.Sin(0.2 * N)
            M = 20 + Math.Round(Y * 10)
            For I = 0 To M — 1
                Console.Write("*")
            Next
            Console.WriteLine("*")
        Next
        Console.Read()
    End Sub
End Module
```

Изменение чисел по условию

Необходимо возвести в квадрат трехзначное число, которое вводится с клавиатуры, при условии содержания в нем цифры 7. Результат вывести на экран. Если условие не выполняется, то следует вывести число в исходном виде. Разработка представлена в листинге 2.42.

Листинг 2.42. Возведение числа в квадрат при условии наличия в нем цифры 7

```
Module Module1
    Sub Main()
        Dim N, A, B, C As Integer
        Console.Write("Введите число")
```

```
        N = Console.ReadLine()
        A = N \ 100
        B = (N \ 10) Mod 10
        C = N Mod 10
        If A = 7 Or B = 7 Or C = 7 Then
            N = N * N
        End If
        Console.WriteLine(N)
        Console.Read()
    End Sub
End Module
```

В следующем задании необходимо возвести в квадрат трехзначное число, которое вводится с клавиатуры, при условии, что все цифры в нем различные. В противном случае следует вывести число в исходном виде. Разработка представлена в листинге 2.43.

Листинг 2.43. Возведение числа в квадрат при условии в нем различных цифр

```
Module Module1
    Sub Main()
        Dim N, A, B, C As Integer
        Console.Write("Введите трехзначное число")
        N = Console.ReadLine()
        A = N \ 100
        B = (N Mod 100) \ 10
        C = N Mod 10
        If A <> B And B <> C And A <> C Then
            N = N * N
        End If
        Console.WriteLine(N)
        Console.Read()
    End Sub
End Module
```

Типовые задачи и задания из ЕГЭ

В этом разделе приведены готовые листинги программ, которые вам требуется проанализировать. Создания каких-либо программных разработок здесь не требуется. Желательно находить правильные ответы без использования компьютера. В последнем разделе главы приведены правильные ответы.

ЗАДАЧА 2.1

В листинге 2.44 приведена программа, в начале которой с клавиатуры вводятся значения переменных `A` и `B`. Что будет выведено на экран, если в начале программы введено: `A=7` и `B=5`?

Листинг 2.44. Программа к задаче 2.1

```
Module Module1
    Sub Main()
        Dim A,B,C As Integer
        Console.Write("Введите число A ")
        A = Console.ReadLine()
        Console.Write("Введите число B ")
        B = Console.ReadLine()
        C= A \ 3
        C= C + B Mod 10
        A= C Mod B
        If C < 3 Then
          Console.WriteLine(C*C)
        Else
          Console.WriteLine(C+A)
        End If
        Console.Read()
     End Sub
End Module
```

ЗАДАЧА 2.2

В листинге 2.45 приведена программа, в начале которой с клавиатуры вводится значение переменной A. Что будет выведено на экран, если введено: A=5?

Листинг 2.45. Программа к задаче 2.2

```
Module Module1
    Sub Main()
        Dim A, B As Integer
        Console.Write("Введите число A ")
        A = Console.ReadLine()
        A=A+2
        B= A Mod 4
        B=B+7
        A= B Mod 5
        If (A \ 3) > 1 Then
          If (B \ 5) < 3 then
             Console.WriteLine(B)
          End If
       End If
       Console.Read()
  End Sub
End Module
```

ЗАДАЧА 2.3

Что будет выведено на экран в результате выполнения программы, приведенной в листинге 2.46?

Листинг 2.46. Программа к задаче 2.3

```
Module Module1
    Sub Main()
        Dim A,B As Integer
        A=17
        B=21
        If ((A Mod 7) > 8) Or ((B Mod 5) < 3) Then
               Console.WriteLine(A)
        Else
               Console.WriteLine(B)
        End If
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 2.4

Что будет выведено на экран в результате выполнения программы, приведенной в листинге 2.47?

Листинг 2.47. Программа к задаче 2.4

```
Module Module1
    Sub Main()
        Dim A As Integer
        A=8
        If ((A Mod 3) >= 2) And ((A \ 3) = 2) Then
              Console.WriteLine(A*A)
        Else
              Console.WriteLine(A*A*A)
        End If
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 2.5

Что будет выведено на экран в результате выполнения программы, приведенной в листинге 2.48?

Листинг 2.48. Программа к задаче 2.5

```
Module Module1
    Sub Main()
        Dim A As Integer
        A=5
        If ((A Mod 3) > 2) And ((A \ 3) = 2) then
             Console.WriteLine(A)
        Else
             Console.WriteLine(A*A)
       End If
       Console.Read()
  End Sub
End Module
```

ЗАДАЧА 2.6

Определите значение целочисленной переменной S после выполнения фрагмента программы, представленного в листинге 2.49.

Листинг 2.49. Фрагмент программы к задаче 2.6

```
S=0
N=6
For I=2 to N
  S=S+2*I
Next
```

ЗАДАЧА 2.7

Дан фрагмент программы, который представлен в листинге 2.50. Чему будет равно значение переменных S и T после выполнения фрагмента программы?

Листинг 2.50. Фрагмент к задаче 2.7

```
S=0
T=1
For I=1 to 5
   S = S + T
   T = T + 1 + I
Next
```

ЗАДАЧА 2.8

Определите значение переменной S после выполнения фрагмента программы, представленного в листинге 2.51.

Листинг 2.51. Фрагмент к задаче 2.8

```
S=0
For J=1 to 5
   S=S+J
End If
```

ЗАДАЧА 2.9

Определите значение целочисленной переменной `S` после выполнения фрагмента программы, представленного в листинге 2.52.

Листинг 2.52. Фрагмент к задаче 2.9

```
S=4
N=6
For I=2 To N
    S=S+(S Mod I)
Next
```

ЗАДАЧА 2.10

Определите значение целочисленных переменных `A`, `B` и `C` после выполнения фрагмента программы, представленного в листинге 2.53.

Листинг 2.53. Фрагмент программы к задаче 2.10

```
A = 3
B = A * (A — 1)
C = 5 — B
B = A — C
```

ЗАДАЧА 2.11

Что будет выведено на экран в результате выполнения программы, приведенной в листинге 2.54?

Листинг 2.54. Программа к задаче 2.11

```
Module Module1
    Sub Main()
        Dim A,B,C As Integer
        A=5
        B=A
        A = A + 2*B
       If A > 10 Then
           C = 2*A
```

```
        Else
           C = -2*A
        End If
        Console.WriteLine(C)
        Console.Read()
  End Sub
End Module
```

ЗАДАЧА 2.12

Что будет выведено на экран в результате выполнения программы, приведенной в листинге 2.55?

Листинг 2.55. Программа к задаче 2.12

```
Module Module1
    Sub Main()
        Dim A, B, C As Integer
        A = 5
        B = A — 1
        A = A + 2 * B
        If A > 13 Then
            C = 2 * A
        Else
            C = -2 * A
        End If
        Console.WriteLine(C)
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 2.13

Что будет выведено на экран в результате выполнения программы, приведенной в листинге 2.56?

Листинг 2.56. Программа к задаче 2.13

```
Module Module1
    Sub Main()
        Dim A, B, C As Integer
        A = 5
        B = -A
        A = A + 2 * B
        If A > 3 Then
            C = 2 * A
```

```
        Else
            C = -2 * A
        End If
        Console.WriteLine(C)
        Console.Read()
  End Sub
End Module
```

ЗАДАЧА 2.14

Что будет выведено на экран в результате выполнения программы, приведенной в листинге 2.57?

Листинг 2.57. Программа к задаче 2.14

```
Module Module1
    Sub Main()
        Dim A, B, C As Integer
        A = 5
        B = 10
        A = A + B
        If A > 10 Then
            C = 2 * A
        Else
            C = -3 * A
        End If
        Console.WriteLine(C)
        Console.Read()
  End Sub
End Module
```

Ответы к типовым задачам и заданиям из ЕГЭ

Задача 2.1

Целочисленное деление 7 на 3 дает 2 в качестве результата. После этого полученное значение складывается с числом 5 (остаток от деления 5 на 10) и в результате в переменную `C` заносится 7. Переменной `A` присваивается значение 2 (остаток деления 7 на 5). Оператор условия, которое ложно, приводит к выводу на экран суммы `C` и `A`. Это число 9.

Задача 2.2

Вначале значение переменной `A` увеличивается на 2 (получается 7). После этого в переменной `B` формируется значение 3. Следующая строка увеличивает `B` на 7.

Осталось вычислить остаток деления 10 на 5. Таким образом, перед условными операторами A = 0, а B = 10. В результате первое условие ложно и на экране после введенного значения мы не увидим ничего.

Задача 2.3

Остаток от деления A на 7 равняется 3, а остаток от деления B на 5 равен 1. В результате одно условие из двух истинно и на экране мы увидим 17.

Задача 2.4

Обе составляющие условия истинны. В результате на экране мы увидим 64.

Задача 2.5

Одна из составляющих условия дает ложный результат. В результате на экране мы увидим 25.

Задача 2.6

Цикл в представленном здесь программном фрагменте выполняется 5 раз (по I, начиная с 2 до 6 с шагом 1). Приведем результаты вычисления переменной S после каждого шага:

- I=2; S=4;
- I=3; S=10;
- I=4; S=18;
- I=5; S=28;
- I=6; S=40.

Таким образом, ответ на поставленный вопрос: S = 40.

Задача 2.7

Цикл в данном программном фрагменте выполняется 5 раз (по I, начиная с 1 до 5 с шагом 1). Приведем результаты вычисления переменной S после каждого шага по I:

- I=1; S=1; T=3;
- I=2; S=4; T=6;
- I=3; S=10; T=10;
- I=4; S=20; T=15;
- I=5; S=35; T=21.

Таким образом, значение переменной S = 35, а значение переменной T = 21.

Задача 2.8

Здесь цикл выполняется 5 раз (по J, начиная с 1 до 5). Приведем результаты вычисления переменной S после каждого шага:

- J=1; S=1;
- J=2; S=3;
- J=3; S=6;
- J=4; S=10;
- J=5; S=15.

Таким образом, правильный ответ: S = 15.

Задача 2.9

Цикл выполняется 5 раз (по I, начиная с 2 до 6 с шагом 1). Приведем результаты вычисления переменной S после каждого шага:

- I=2; S=4 (остаток от деления 4 на 2 равен 0);
- I=3; S=5 (остаток от деления 4 на 3 равен 1);
- I=4; S=6 (остаток от деления 5 на 4 равен 1);
- I=5; S=7 (остаток от деления 6 на 5 равен 1);
- I=6; S=8 (остаток от деления 7 на 6 равен 1).

Таким образом, правильный ответ: S = 8.

Задача 2.10

Второй оператор в листинге 2.53 приводит к результату: B = 6. Третий оператор приводит к результату: C = -1. Последнее вычисление меняет значение B: B = 4. В результате значение переменной A = 3, B = 4, C = -1.

Задача 2.11

Переменная B принимает значение 5. После этого вычисляется новое значение переменной A, что дает 15. В результате условие выполняется и значение переменной C равняется 30 (2 умножается на 15).

Задача 2.12

Переменная B принимает значение 4. После этого вычисляется новое значение переменной A, что дает 13 (к 5 добавляется 8). В результате условие не выполняется и значение переменной C равняется -26.

Задача 2.13

Переменная B принимает значение -5. После этого вычисляется новое значение переменной A, что дает -5. В результате условие не выполняется и значение переменной C равняется 10.

Задача 2.14

После присвоения значений переменным A и B вычисляется новое значение переменной A, что дает 15. В результате условие выполняется и значение переменной C равняется 30.

ГЛАВА 3

Одномерные массивы

Во многих практических ситуациях приходится работать с большим объемом данных. Для этого удобно использовать массивы, которые являются достаточно простой в плане программирования структурой данных. Использование массива приводит к выделению в памяти набора ячеек под определенным именем. Формально определение *массива* таково: совокупность однотипных данных, хранящихся в последовательных ячейках памяти и имеющих общее имя. Ячейки называются *элементами* массива. Все элементы массива пронумерованы по порядку, а номер называется *индексом* элемента массива.

Важно отметить, что все элементы массива имеют один и тот же тип данных. Для обращения к конкретному элементу массива необходимо указать имя массива и в круглых скобках индекс элемента. Например, это выглядит так — `A(5)`.

Каждый элемент массива можно использовать в вычислениях, так же как и простую переменную.

В Visual Basic существуют массивы *фиксированного размера* и *динамические* массивы. Массив фиксированного размера имеет неизменный размер, заданный при его объявлении. Динамические массивы могут изменять размер в процессе выполнения.

Массивы могут быть *одномерными* и *многомерными*. В этой главе мы рассмотрим технологию работы с одномерными массивами.

При объявлении массива после его имени в круглых скобках указывается верхняя граница массива (максимальный индекс). Нижней границей индекса всегда является 0. Например, в следующем коде задается массив, состоящий из 11 элементов:

```
Dim A(10) As Integer
```

Здесь указывается тип данных, который имеет каждый элемент массива.

Visual Basic во время выполнения программы имеет возможность изменять размеры массивов. Это позволяет обеспечить более эффективное управление памятью. Для создания динамического массива необходимо объявить массив без задания верхней границы индекса. Например, это выглядит так:

```
Dim A() As Integer
```

Когда в программе требуется поработать с данным массивом, то используется оператор `ReDim`. В данном операторе в качестве параметра указывается необходимая в текущий момент размерность массива. Например, для объявленного массива `A` изменение выглядит так:

```
ReDim A(11)
```

Далее в этой главе рассматриваются примеры работы с одномерными массивами. В целом, данная глава построена на разнообразных практических заданиях, рассмотрение которых позволит вам сформировать навыки практической работы с массивами.

Нахождение суммы элементов массива

Исходная ситуация традиционна для работы с массивами: дан массив `A(J)`, где индекс `J` принимает значения от 1 до `N`. Будем считать, что элементы массива являются целыми числами (имеют тип `Integer`).

> ***Примечание***
>
> В приводимых примерах мы будем использовать элементы массива, начиная с индекса 1.

Сумма элементов массива вычисляется по следующей формуле:

$$S = \sum_{J=1}^{N} A[J]. \qquad (3.1)$$

Алгоритм вычисления суммы `S` по формуле (3.1) достаточно простой и программа, его реализующая, не требует какого-либо пояснения. В листинге 3.1 приведена разработка, обеспечивающая решение исходной задачи. Отметим, что для внесения начальной информации мы воспользовались датчиком случайных чисел — стандартной функцией `Rnd`. Данная функция при обращении к ней выдает случайное (дробное) число в интервале от 0 до 1. Таким образом, в программе с помощью `Rnd` мы обеспечиваем случайное заполнение элементов массива `A` дробными числами.

Для того чтобы при каждом запуске программы данное заполнение было различным, мы в начале программы включили стандартную процедуру `Randomize`, которая обеспечивает каждый раз новую последовательность генерируемых чисел.

Далее в программе выполняется суммирование элементов массива и вывод результата с ограничением разрядности. В результате работы мы увидим на экране подсчитанную сумму.

Листинг 3.1. Вычисление суммы значений элементов массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10), S As Single
```

```
        Dim J As Integer
        Randomize()
        For J = 1 To N
            A(J) = Rnd()
        Next
        S = 0
        For J = 1 To N
            S = S + A(J)
        Next
        Console.Write("Сумма элементов равна ")
        Console.Write(Format(S, "0.00"))
        Console.Read()
    End Sub
End Module
```

Суммирование элементов массива с учетом условия

Рассмотрим задачу суммирования не всех элементов массива, а только тех, которые удовлетворяют определенному условию. При этом для элементов массива укажем целочисленный тип и при заполнении воспользуемся следующей конструкцией:

```
A(J) = Int(10 * Rnd())
```

Здесь мы применили нормировку значений выдаваемых случайным датчиком на 10, а также для отбрасывания дробной части числа использовали стандартную функцию `Int`. В результате массив A будет заполнен целыми числами в интервале от 0 до 9 включительно.

В качестве условия будем считать, что суммироваться должны только те элементы, значения которых кратны 3. В листинге 3.2 приведена необходимая программная разработка. Перечислим действия, которые в ней выполняются:

- заполнение элементов массива случайными значениями;
- организация цикла по количеству элементов массива;
- проверка условия кратности значения элемента массива трем, и если это условие выполняется, то осуществляется добавление значения рассматриваемого элемента к общей сумме.

Аналогичным образом можно использовать и другие условия при суммировании элементов массива.

> ***Примечание***
>
> Для иллюстрации в листинге 3.2 мы вывели на экран вместе с результатом и элементы сформированного массива.

Результат работы такой программы приведен на рис. 3.1.

Листинг 3.2. Суммирование элементов массива при условии кратности 3

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10), S As Integer
        Dim J As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(10 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
        S = 0
        For J = 1 To N
            If A(J) Mod 3 = 0 Then S = S + A(J)
        Next
        Console.WriteLine()
        Console.Write("Сумма интересующих элементов равна ")
        Console.Write(S)
        Console.Read()
    End Sub
End Module
```

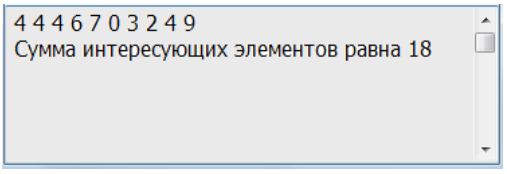

Рис. 3.1. Вывод информации в программе, приведенной в листинге 3.2

Нахождение среднего арифметического

Среднее арифметическое значение элементов массива вычисляется по следующей формуле:

$$S = \frac{1}{N}\sum_{J=1}^{N} A[J]. \qquad (3.2)$$

Алгоритм вычисления соотношения (3.2) похож на ранее рассмотренное вычисление суммы элементов массива, и в листинге 3.3 приведена программа, реализующая решение данной задачи.

Для иллюстрации работы функции `Int` в данном примере мы вывели на экран исходные значения, выдаваемые датчиком `Rnd` (с нормировкой на 10).

Результат работы такой программы приведен на рис. 3.2.

Листинг 3.3. Вычисление среднего арифметического значения элементов массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10), S As Integer
        Dim J As Integer
        Dim T, Srednee As Single
        Randomize()
        For J = 1 To N
            T = 10 * Rnd()
            A(J) = Int(T)
            Console.Write(A(J))
            Console.WriteLine()
        Next
        S = 0
        For J = 1 To N
            S = S + A(J)
        Next
        Srednee = S / N
        Console.Write("Сумма элементов равна ")
        Console.WriteLine(S)
        Console.Write("Среднее значение равно ")
        Console.Write(Srednee)
        Console.Read()
    End Sub
End Module
```

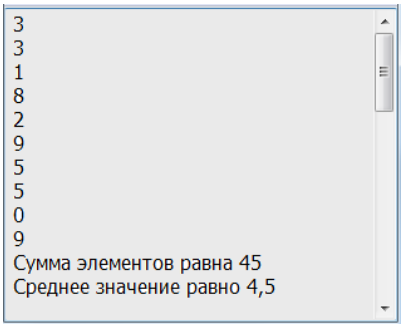

Рис. 3.2. Вывод информации в программе, приведенной в листинге 3.3

Внесем небольшие изменения в предыдущую программу. Так, будем считать, что нам необходимо подсчитать средние значения отдельно:

- по всем элементам массива;
- по 10% исходных данных (10% всех элементов).

Программная разработка, решающая поставленную задачу, приведена в листинге 3.4. Здесь для искомых средних значений мы ввели две переменные: `S1` и `S2`. Для числа элементов, составляющих 10% всей совокупности данных, отвели переменную `N2`.

Результат работы такой программы приведен на рис. 3.3.

Листинг 3.4. Вычисление среднего арифметического значения элементов массива

```
Module Module1
    Sub Main()
        Const N = 1000
        Dim A(1000) As Single
        Dim J, N2 As Integer
        Dim S1, S2 As Single
        Randomize()
        For J = 1 To N
            A(J) = Rnd()
        Next
        S1 = 0
        S2 = 0
        N2 = N \ 10
        For J = 1 To N
            S1 = S1 + A(J)
            If J <= N2 Then S2 = S2 + A(J)
        Next
        S1 = S1 / N
        S2 = S2 / N2
        Console.Write("Среднее значение по всей совокупности равно ")
        Console.WriteLine(S1)
        Console.Write("Среднее значение по 10-ти процентам равно ")
        Console.Write(S2)
        Console.Read()
    End Sub
End Module
```

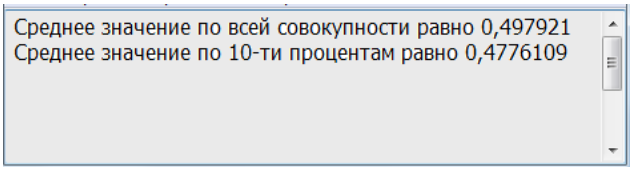

Рис. 3.3. Вывод информации в программе, приведенной в листинге 3.4

Поиск максимального элемента в массиве

В листинге 3.5 приведена программа поиска максимального элемента в числовом массиве. Сам массив предварительно заполняется случайными целыми числами.

Результат работы такой программы приведен на рис. 3.4.

Листинг 3.5. Поиск максимального элемента в массиве

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, Maxsimum As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(100 * Rnd())
            Console.WriteLine(A(J))
        Next
        Maxsimum = A(1)
        For J = 2 To N
            If A(J) > Maxsimum Then Maxsimum = A(J)
        Next
        Console.Write("Максимальный элемент ")
        Console.WriteLine(Maxsimum)
        Console.Read()
    End Sub
End Module
```

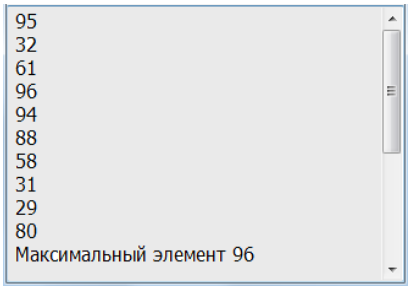

Рис. 3.4. Вывод информации в программе, приведенной в листинге 3.5

Поиск индексов в массиве

Рассмотрим еще один пример. Пусть наша задача заключается в том, чтобы найти индексы максимального и минимального элементов в массиве. В листинге 3.6 приведена программа поиска указанных индексов для массива элементов, предварительно заполненного случайными числами.

Примечание

Если максимальных или минимальных элементов несколько, то данная программа отображает только индексы первых найденных элементов.

Результат работы такой программы приведен на рис. 3.6.

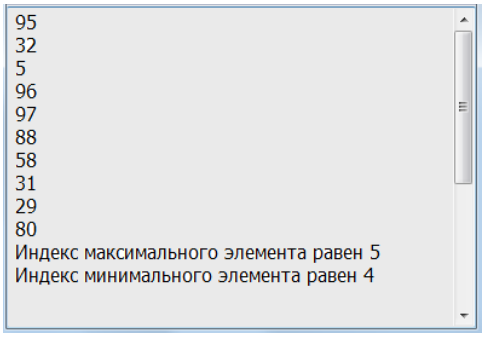

Рис. 3.5. Вывод информации в программе, приведенной в листинге 3.6

Листинг 3.6. Поиск индексов минимального и максимального элементов массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, Jmax, Jmin, Maximum, Minimum As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(100 * Rnd())
            Console.WriteLine(A(J))
        Next
        Maximum = A(1)
        Minimum = A(1)
        Jmax = 1
        Jmin = 1
        For J = 2 To N
            If A(J) > Maximum Then
                Maximum = A(J)
                Jmax = J
            End If
            If A(J) < Minimum Then
                Minimum = A(J)
                Jmin = J
            End If
        Next
        Console.Write("Индекс максимального элемента равен ")
        Console.WriteLine(Jmax)
```

```
        Console.Write("Индекс минимального элемента равен ")
        Console.WriteLine(Jmin)
        Console.Read()
    End Sub
End Module
```

Проверка упорядоченности массива

Иногда возникает необходимость проверки упорядоченности массива — все ли элементы расставлены в порядке возрастания. В листинге 3.7 приведена программа, которая попарно анализирует соседние элементы массива. Здесь мы ввели переменную `Flag`, в которую при нахождении нарушения упорядоченности в массиве (если текущий элемент больше последующего) заносится значение 1 (первоначально значение этой переменной инициализируется нулем). Это и является индикатором при формировании сообщения о неупорядоченности массива чисел. В противном случае если переменная `Flag` остается равной нулю, то можно сказать, что массив упорядочен.

Результат работы такой программы приведен на рис. 3.6.

Листинг 3.7. Анализ упорядоченности элементов массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, Flag As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(100 * Rnd())
            Console.WriteLine(A(J))
        Next
        Flag = 0
        For J = 1 To N — 1
            If A(J) > A(J + 1) Then
                Flag = 1
                Exit For
            End If
        Next
        If Flag = 0 Then
            Console.Write("Массив упорядочен")
        Else
            Console.Write("Массив неупорядочен")
        End If
```

```
        Console.Read()
    End Sub
End Module
```

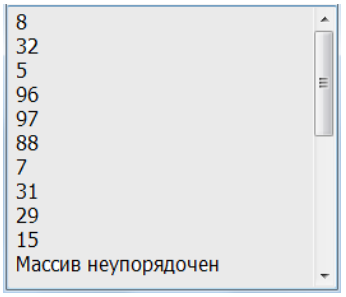

Рис. 3.6. Вывод информации в программе, приведенной в листинге 3.7

Обмен значений массива

Обмен значений элементов массива достаточно часто встречается при работе с массивами. В программе, приведенной в листинге 3.8, осуществляется следующий вариант обмена значений: первый элемент меняется значением с последним, второй с предпоследним и т. д.

Сам алгоритм несложен, а его идея заключается в том, что сначала вычисляется количество обменов. Для этого следует число элементов массива поделить на 2 (выполнить целочисленное деление). После этого в цикле по числу обменов выполняется обмен значений элементов массива.

Результат работы такой программы приведен на рис. 3.7.

Листинг 3.8. Обмен значений элементов массива

```
Module Module1
    Sub Main()
        Const N = 7
        Dim A(7) As Integer
        Dim J, B, L As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(100 * Rnd())
            Console.WriteLine(A(J))
        Next
        L = N \ 2
        For J = 1 To L
            B = A(J)
            A(J) = A(N — J + 1)
            A(N — J + 1) = B
```

```
        Next
        Console.WriteLine()
        For J = 1 To N
            Console.WriteLine(A(J))
        Next
        Console.Read()
    End Sub
End Module
```

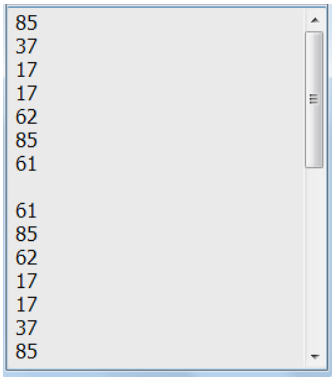

Рис. 3.7. Вывод информации в программе, приведенной в листинге 3.8

Суммирование соседних элементов массива

Необходимо разработать алгоритм поиска номера первого из двух последовательных элементов в целочисленном массиве, состоящем из 10 элементов, сумма которых максимальна (если таких пар несколько, то можно выбрать любую). Решение этой задачи похоже на поиск номера максимального элемента, а отличием является только то, что здесь требуется анализировать сумму двух соседних элементов массива.

> ***Примечание***
>
> Данная задача встретилась в одном из вариантов ЕГЭ в прошлые годы.

В листинге 3.9 приведена необходимая программная разработка. В начале программы в переменную `SumMax` заносится сумма первых двух элементов, а далее последовательно сравниваются с `SumMax` суммы следующих пар элементов. В случае, когда сумма элементов очередной пары оказывается больше `SumMax`, то это приводит к обновлению `SumMax` и фиксированию индекса первого элемента пары в переменной `Jmax`. В конце программы значение `Jmax` выводится на печать.

Результат работы такой программы приведен на рис. 3.8.

Листинг 3.9. Поиск пары соседних элементов с максимальной суммой

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, Jmax, SumMax As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(100 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
        SumMax = A(1) + A(2)
        Jmax = 1
        For J = 2 To N — 1
            If (A(J) + A(J + 1)) > SumMax Then
                Jmax = J
                SumMax = A(J) + A(J + 1)
            End If
        Next
        Console.WriteLine()
        Console.Write(Jmax)
        Console.Read()
    End Sub
End Module
```

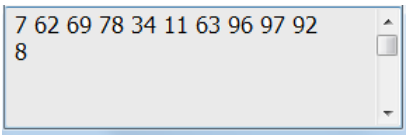

Рис. 3.8. Вывод информации в программе, приведенной в листинге 3.9

Подсчет соседних элементов по условию

Необходимо разработать алгоритм подсчета максимального количества идущих подряд совпадающих элементов в целочисленном массиве, который имеет длину 10. Программа, решающая данную задачу, представлена в листинге 3.10. Учитывая некоторую неочевидность алгоритма, сделаем пояснения, которые предоставляют необходимую информативность. Так, мы для фиксирования максимального количества подряд идущих элементов определили переменную `MaxNums`, а для хранения текущего числа совпадающих элементов ввели переменную `Nums`. В цикле анализируется текущий и предыдущий элементы массива. Если они оказываются равными, то увеличивается значение счетчика `Nums` и анализируется значение следующего элемента массива. Если же предыдущий и текущий элементы не совпадают, то про-

веряется, больше ли Nums чем MaxNums. Если больше, то значение Nums фиксируется в качестве нового значения MaxNums. И в том, и в другом случае после этого сбрасывается в единицу значение Nums и начинается анализ следующего элемента массива.

Примечание

Данная задача встретилась в одном из вариантов ЕГЭ в прошлые годы.

Результат работы такой программы приведен на рис. 3.9.

Листинг 3.10. Подсчет максимального количества подряд идущих элементов массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, Nums, MaxNums As Integer
        Nums = 1
        MaxNums = 1
        Randomize()
        For J = 1 To N
            A(J) = Int(5 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
        For J = 2 To N
            If A(J) = A(J — 1) Then
                Nums = Nums + 1
            Else
                If Nums > MaxNums Then
                    MaxNums = Nums
                End If
                Nums = 1
            End If
        Next
        If Nums > MaxNums Then MaxNums = Nums
        Console.WriteLine()
        Console.Write("Число подряд идущих одинаковых элементов равно ")
        Console.WriteLine(MaxNums)
        Console.Read()
End Sub
End Module
```

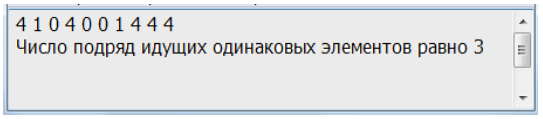

Рис. 3.9. Вывод информации в программе, приведенной в листинге 3.10

Перенос модулей значений в другой массив

Требуется разработать алгоритм получения из заданного целочисленного массива размером 10 элементов другого массива, который будет содержать модули значений элементов первого массива.

> ***Примечание***
>
> Данная задача встретилась в одном из вариантов ЕГЭ в прошлые годы.

Здесь следует вспомнить, что модуль числа a можно записать так:

$|a| = -a$, если $a < 0$;

$|a| = a$, если $a \geq 0$.

Именно это преобразование и необходимо выполнить последовательно над каждым элементом массива. В листинге 3.11 приведена программная разработка для решения данной задачи. Здесь исходным массивом является `A(J)`, а массивом, содержащим модули, — `B(J)`. При заполнении массива с помощью датчика случайных чисел мы обеспечили равновероятное генерирование как положительных, так и отрицательных чисел.

Результат работы такой программы приведен на рис. 3.10.

Листинг 3.11. Перенос модулей значений элементов массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10), B(10) As Integer
        Dim J As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(11 * Rnd()) - 5
            Console.Write(A(J))
            Console.Write(" ")
        Next
        Console.WriteLine()
        For J = 1 To N
            If A(J) < 0 Then
                B(J) = -A(J)
            Else
                B(J) = A(J)
            End If
            Console.Write(B(J))
            Console.Write(" ")
        Next
```

```
        Console.Read()
    End Sub
  End Sub
End Module
```

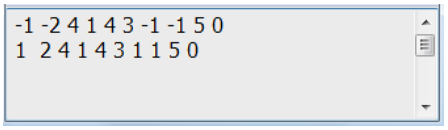

Рис. 3.10. Вывод информации в программе, приведенной в листинге 3.11

Подсчет количества максимальных элементов

Требуется разработать программу подсчета числа элементов в массиве, значения которых равны максимальному элементу. Число элементов в массиве равно 10.

> ***Примечание***
>
> Данная задача встретилась в одном из вариантов ЕГЭ в прошлые годы.

Наиболее простой и очевидный вариант решения заключается в организации двух этапов:

- первоначальный просмотр массива с целью поиска максимального элемента;
- повторный просмотр массива, при котором подсчитывается число элементов, значения которых равны максимальному.

В листинге 3.12 приведено программное решение поставленной задачи данным способом. Результат работы такой программы приведен на рис. 3.11.

Листинг 3.12. Подсчет количества максимальных элементов (вариант 1)

```
Module Module1
   Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, Max, Nums As Integer
        Randomize()
         For J = 1 To N
            A(J) = Int(5 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
        Max= A(1)
        For J=2 To N
```

```
          If A(J)> Max  Then
             Max= A(J)
          End If
       Next
       Nums=0
       For J=1 To N
         If A(J) = Max  then
            Nums= Nums+1
         End If
      Next
      Console.WriteLine()
      Console.Write("Число максимальных элементов равно ")
      Console.Write(Nums)
      Console.Read()
End Sub
End Module
```

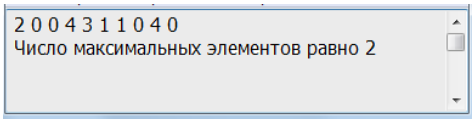

Рис. 3.11. Вывод информации в программе, приведенной в листинге 3.12

Однако такой вариант в инструкции для проверки Единого государственного экзамена рекомендуется оценивать как недостаточно эффективный. Это связано с тем, что в предложенном варианте решения выполняется повторный просмотр элементов массива. Реализуем теперь поставленную задачу по-другому. Построение решения связано с тем, что мы используем переменную `Nums` для подсчета количества максимальных значений массива. При нахождении очередного максимума значение этой переменной сбрасывается в единицу, а при нахождении такого же максимального значения увеличивается на единицу. В листинге 3.13 приведена программная реализация алгоритма.

Листинг 3.13. Подсчет количества максимальных элементов (вариант 2)

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, Max, Nums As Integer
        Randomize()
         For J = 1 To N
            A(J) = Int(5 * Rnd())
        Next
        Max= A(1)
        Nums=1
```

```
        For J=2 to N
         If A(J) > Max  Then
            Max= A(J)
            Nums=1
         Else
           If A(J) = Max  Then   Nums= Nums+1
         End If
       Next
     Console.WriteLine(Nums)
     Console.Read()
End Sub
End Module
```

Изменение значений элементов массива с заданными свойствами

Необходимо изменить определенные значения в массиве. Так, если значение элемента в массиве отрицательное, то меняется знак элемента.

Скажем, если `A(5)=-10`, то после коррекции получим: `A(5)=10`. Кроме того, если значение элемента массива больше 3, то в элемент массива необходимо занести ноль. В листинге 3.14 приведена необходимая программная разработка.

Результат работы такой программы приведен на рис. 3.12.

Листинг 3.14. Изменение значений элементов в массиве

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(11 * Rnd())-5
            Console.Write(A(J))
            Console.Write(" ")
       Next
       Console.WriteLine()
       For J=1 to N
           If A(J) < 0  Then   A(J) = – A(J)
           If A(J) > 3  Then   A(J) =0
           Console.Write(A(J))
           Console.Write(" ")
        Next
       Console.Read()
   End Sub
End Module
```

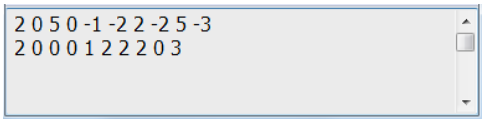

Рис. 3.12. Вывод информации в программе, приведенной в листинге 3.14

Нахождение индексов элементов с заданными свойствами

Необходимо найти и вывести на экран номера элементов массива, значения которых кратны 4. Сам массив заполняется случайными целыми числами в интервале от 1 до 20 включительно. В листинге 3.15 приведена необходимая программная разработка.

Результат работы такой программы приведен на рис. 3.13.

Листинг 3.15. Нахождение индексов элементов массива кратных 4

```
Module Module1
    Sub Main()
        Const N = 20
        Dim A(20) As Integer
        Dim J As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(20 * Rnd()) + 1
            Console.Write(A(J))
            Console.Write(" ")
        Next
        Console.WriteLine()
        For J = 1 To N
            If (A(J) Mod 4) = 0 Then
                Console.Write(J)
                Console.Write(" ")
            End If
        Next
        Console.Read()
    End Sub
End Module
```

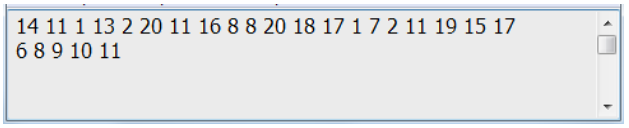

Рис. 3.13. Вывод информации в программе, приведенной в листинге 3.15

Удаление из массива определенного элемента

Допустим, имеется массив из N элементов. Необходимо удалить элемент с определенным индексом (вводимым с клавиатуры). Один из вариантов программы показан в листинге 3.16.

Результат работы такой программы приведен на рис. 3.14.

Листинг 3.16. Удаление определенного элемента из массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, K As Integer
        Console.Write("Введите номер удаляемого элемента массива ")
        K = Console.ReadLine()
        For J = 1 To N
            A(J) = Int(20 * Rnd()) + 1
            Console.Write(A(J))
            Console.Write(" ")
        Next
        For J = K To N — 1
            A(J) = A(J + 1)
        Next
        Console.WriteLine()
        For J = 1 To N — 1
            Console.Write(A(J))
            Console.Write(" ")
        Next
        Console.Read()
    End Sub
End Module
```

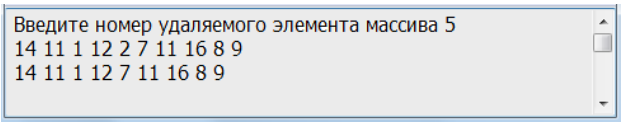

Рис. 3.14. Вывод информации в программе, приведенной в листинге 3.16

Циклическое перемещение элементов массива

Допустим, необходимо осуществить циклическое перемещение элементов по такой схеме: второй элемент перемещается на место первого, третий на место второго и т. д. Исходный первый элемент переходит в последний элемент массива.

Один из вариантов программы показан в листинге 3.17. Результат работы такой программы приведен на рис. 3.15.

Листинг 3.17. Циклическое перемещение элементов массива

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10) As Integer
        Dim J, B As Integer
        For J = 1 To N
            A(J) = Int(20 * Rnd()) + 1
            Console.Write(A(J))
            Console.Write(" ")
        Next
        B = A(1)
        For J = 1 To N — 1
            A(J) = A(J + 1)
        Next
        A(N) = B
        Console.WriteLine()
        For J = 1 To N
            Console.Write(A(J))
            Console.Write(" ")
        Next
        Console.Read()
End Sub
End Module
```

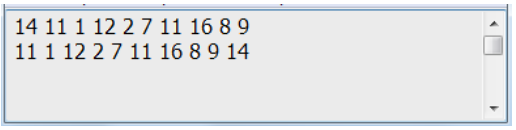

Рис. 3.15. Вывод информации в программе, приведенной в листинге 3.17

Заполнение массива случайными числами

Требуется организовать заполнение массива случайными числами. При этом в массиве не должно оказаться одинаковых элементов. Один из вариантов программы показан в листинге 3.18.

Листинг 3.18. Заполнение массива случайными числами

```
Module Module1
    Sub Main()
        Const N = 10
```

```
        Dim A(10) As Integer
        Dim J, I, Flag As Integer
        J = 1
        While J <= N
            A(J) = Int(15 * Rnd())
            Flag = 0
            For I = 1 To J — 1
                If A(J) = A(I) Then
                    Flag = 1
                    Exit For
                End If
            Next
            If Flag = 0 Then J = J + 1
        End While
        For J = 1 To N
            Console.WriteLine(A(J))
        Next
        Console.Read()
  End Sub
End Module
```

Нахождение суммы группы элементов массива

Дан массив целых чисел. Необходимо найти группы из трех подряд идущих элементов, сумма значений которых максимальна. Требуется вывести на экран индекс первого элемента каждой группы и сумму группы элементов. Один из вариантов программы показан в листинге 3.19. Результат работы такой программы приведен на рис. 3.16.

Листинг 3.19. Нахождение группы элементов массива с максимальной суммой

```
Module Module1
    Sub Main()
        Const N = 20
        Dim A(20) As Integer
        Dim J, M1, M2 As Integer
        J = 1
        Randomize()
        For J = 1 To N
            A(J) = Int(5 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
        M1 = A(1) + A(2) + A(3)
```

```
        For J = 2 To N - 2
            M2 = A(J) + A(J + 1) + A(J + 2)
            If M2 > M1 Then
                M1 = M2
            End If
        Next
        Console.WriteLine()
        For J = 1 To N - 2
            M2 = A(J) + A(J + 1) + A(J + 2)
            If M1 = M2 Then
                Console.Write(J)
                Console.Write(" ")
                Console.WriteLine(M2)
            End If
        Next
        Console.Read()
  End Sub
End Module
```

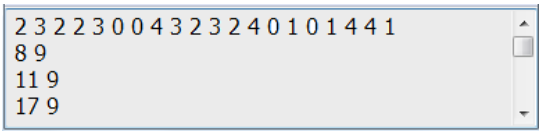

Рис. 3.16. Вывод информации в программе, приведенной в листинге 3.19

Задания из ЕГЭ

В этом разделе приведены готовые листинги программ, которые вам следует проанализировать. Создания каких-либо программных разработок здесь не требуется. В последующих задачах желательно находить правильные ответы без использования компьютера. В последнем разделе главы приведены правильные ответы.

ЗАДАЧА 3.1

Значения двух одномерных массивов `A` и `B`, размером 100 элементов каждый, задаются с помощью представленного в листинге 3.20 фрагмента программы. Вопрос: сколько элементов массива `B` будут иметь отрицательные значения?

Листинг 3.20. Фрагмент к задаче 3.1

```
For I=1 To 100
    A(I)=50 - I
Next
For I=1 to 100
   B(I)=A(I) + 49
Next
```

ЗАДАЧА 3.2

Значения элементов массива A задаются с помощью вложенного цикла во фрагменте, представленном в листинге 3.21. Сколько элементов массива будут кратны 4?

Листинг 3.21. Фрагмент к задаче 3.2

```
For I=1 To 8
    A(I)=I
    A(I)= A(I) + I*(8-I)
Next
```

ЗАДАЧА 3.3

Значения элементов массива A размером 5 задаются с помощью цикла во фрагменте, представленном в листинге 3.22. Сколько элементов массива будут кратны 3?

Листинг 3.22. Фрагмент к задаче 3.3

```
For I=1 To 5
    A(I,J)= I * (6-I)
Next
```

ЗАДАЧА 3.4

Значения элементов массива размером 5 задаются с помощью вложенного цикла во фрагменте, представленном в листинге 3.23. Сколько элементов массива будут иметь значения больше 3?

Листинг 3.23. Фрагмент к задаче 3.4

```
For I=1 To 5
   A(I)= I*(I-2)
Next
```

ЗАДАЧА 3.5

Значения элементов массива размером 5 задаются с помощью цикла во фрагменте, представленном в листинге 3.24. Сколько элементов массива будут иметь отрицательные значения?

Листинг 3.24. Фрагмент к задаче 3.5

```
For I=1 To 5
   If ((I Mod 3) = 1) Or ((I Mod 2) = 0)  Then
       A(I)= I*(5-I)
```

```
   Else
      A(I)= I*(4-I)
End If
Next
```

ЗАДАЧА 3.6

Дан фрагмент программы (листинг 3.25), обрабатывающей одномерный массив A с индексами от 0 до 10. Определить, чему будут равны элементы массива A после выполнения данного фрагмента программы.

Листинг 3.25. Фрагмент к задаче 3.6

```
For I=1 To 10
       A(I)= 7+I
Next
For I=0 To 4
     A(10-I)= A(I)-1
     A(I)= A(10-I)-1
Next
```

ЗАДАЧА 3.7

Дан фрагмент программы (листинг 3.26), обрабатывающей одномерный массив A с индексами от 0 до 10. Определить, чему будут равны элементы массива A после выполнения данного фрагмента программы.

Листинг 3.26. Фрагмент к задаче 3.7

```
For I=0 To 10
     A(I)= 2+I
Next
For I=0 To 4
     A(10-I)= A(I)-1
     A(I)= A(I)-2
Next
```

Ответы к заданиям из ЕГЭ

Задача 3.1

Проанализируем ситуацию. Во втором цикле отрицательные значения элементов массива B будут в случае, если значения массива A(I) будут иметь значения меньшие, чем –49. Это следует из того, что к значению каждого элемента массива A добавляется 49. Из верхнего цикла видно, что среди элементов массива A таковым

является только одно значение (–50), которое получается при `I = 100`. Следовательно, и в массиве `B` будет одно отрицательное значение.

Задача 3.2

Проанализируем массив:

- при `I=1` значение `A(I)` равно 9;
- при `I=2` значение `A(I)` равно 14;
- при `I=3` значение `A(I)` равно 18;
- при `I=4` значение `A(I)` равно 20;
- при `I=5` значение `A(I)` равно 20;
- при `I=6` значение `A(I)` равно 18;
- при `I=7` значение `A(I)` равно 14;
- при `I=8` значение `A(I)` равно 8.

Всего в массиве 3 элемента, кратных 4.

Задача 3.3

Требуется проанализировать результат вычисления каждого элемента массива:

- при `I=1` значение `A(I)` равно 5;
- при `I=2` значение `A(I)` равно 8;
- при `I=3` значение `A(I)` равно 9;
- при `I=4` значение `A(I)` равно 8;
- при `I=5` значение `A(I)` равно 5.

Таким образом, в массиве один элемент, кратный 3.

Задача 3.4

Проанализируем массив:

- при `I=1` значение `A(I)` равно –1;
- при `I=2` значение `A(I)` равно 0;
- при `I=3` значение `A(I)` равно 3;
- при `I=4` значение `A(I)` равно 8;
- при `I=5` значение `A(I)` равно 15.

Ответ: два элемента массива будут иметь значения больше 3.

Задача 3.5

- `A(I)` при `I=1` значение `A(I)` равно 4;
- при `I=2` значение `A(I)` равно 6;
- при `I=3` значение `A(I)` равно 3;

- при `I=4` значение `A(I)` равно 4;
- при `I=5` значение `A(I)` равно –5.

Ответ: один элемент массива будет иметь отрицательное значение.

Задача 3.6

Сначала массив заполняется последовательными числами от 7 до 17 (всего 11 элементов массива). Далее средний элемент массива (имеющий индекс 5) остается неизменным. Первые и последние 5 значений меняются местами и корректируются. Изменение элементов массива представляет собой последовательное присваивание пяти элементам, отсчитываемым с конца, значений на единицу меньших, чем значения в симметричных ячейках (относительно центрального элемента) массива, но отсчитываемых с его начала. Значения элементов начала массива затем сразу же заменяются на значения в симметричных элементах (относительно центрального элемента), уменьшенные еще на одну единицу.

Ответ: после заполнения массива получаем: 5 6 7 8 9 12 10 9 8 7 6.

Задача 3.7

Сначала массив заполняется последовательными числами от 2 до 12 (всего 11 элементов массива). Далее средний элемент массива (имеющий индекс 5) остается неизменным. Первые и последние 5 значений меняются местами.

Изменение элементов массива представляет собой последовательное присваивание пяти элементам, отсчитываемым с конца, значений на единицу меньших, чем значения в симметричных ячейках (относительно центрального элемента) массива, но отсчитываемых с его начала. Значения элементов начала массива затем сразу же заменяются на значения, уменьшенные на 2.

Ответ: после заполнения массива получаем: 0 1 2 3 4 7 5 4 3 2 1.

ГЛАВА 4

Двумерные массивы

Кроме одномерных массивов, в практических задачах часто используются и двумерные массивы. Двумерный массив представляет собой таблицу из однотипных элементов, организованную по строкам и столбцам. Элемент такого массива записывается так: `A(I,J)`, где первый индекс `I` представляет собой номер строки, а второй индекс `J` является номером столбца. Местоположение каждого элемента массива в памяти компьютера определяется этими индексами.

В языке Бейсик используется следующее описание двумерного массива:

```
Dim имя_массива (максимальный индекс строки, максимальный индекс столбца)of тип элементов
```

Данное описание вполне аналогичное описанию одномерного массива.

Нахождение суммы элементов массива

Исходная ситуация традиционна для работы с двумерными массивами: дан массив `A(I,J)`, где индексы `I`, `J` принимают значения от 1 до `N`. В данном случае будем считать элементы массива целыми числами (имеющими тип `Integer`). Сумма элементов двумерного массива вычисляется по следующей формуле:

$$S = \sum_{I=1}^{N} \sum_{J=1}^{N} A[I,J]. \qquad (4.1)$$

Алгоритм вычисления `S` по соотношению (4.1) достаточно простой, и программа, его реализующая, не требует пояснения (как и ряд других программ, рассматриваемых в примерах этой главы). В листинге 4.1 приведена реализация решения данной задачи в системе Visual Basic 2010. Для определенности мы установили максимальные индексы строк и столбцов равными 5. Как и в разработках предыдущих глав, в данной программе обеспечено первоначальное заполнение элементов массива с помощью датчика случайных чисел.

Результат работы такой программы приведен на рис. 4.1.

Листинг 4.1. Вычисление суммы значений элементов двумерного массива

```
Module Module1
    Sub Main()
        Const N = 5
        Dim A(5, 5) As Integer
        Dim I, J, S As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To N
                A(I, J) = Int(5 * Rnd())
                Console.Write(A(I, J))
                Console.Write(" ")
            Next
            Console.WriteLine()
        Next
        S = 0
        For I = 1 To N
            For J = 1 To N
                S = S + A(I, J)
            Next
        Next
        Console.Write("Сумма элементов двумерного массива равна ")
        Console.WriteLine(S)
        Console.Read()
 End Sub
End Module
```

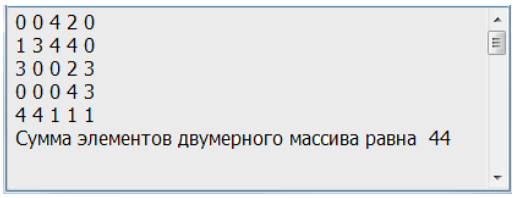

Рис. 4.1. Вывод информации в программе, приведенной в листинге 4.1

Сумма элементов с заданными свойствами

Несколько изменим формулировку предыдущего задания. Будем считать, что требуется просуммировать только те элементы, значения которых являются нечетными числами, кроме того, эти значения должны быть больше трех. В листинге 4.2 приведена программа, реализующая решение данной задачи.

Результат работы такой программы приведен на рис. 4.2.

Листинг 4.2. Вычисление суммы значений элементов массива при условии

```
Module Module1
    Sub Main()
        Const N = 5
        Dim A(5, 5) As Integer
        Dim I, J, S As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To N
                A(I, J) = Int(8 * Rnd())
                Console.Write(A(I, J))
                Console.Write(" ")
            Next
            Console.WriteLine()
        Next
        S = 0
        For I = 1 To N
            For J = 1 To N
                If A(I, J) Mod 2 = 1 And A(I, J) > 3 Then
                    S = S + A(I, J)
                End If
            Next
        Next
        Console.Write("Сумма элементов равна ")
        Console.WriteLine(S)
        Console.Read()
    End Sub
End Module
```

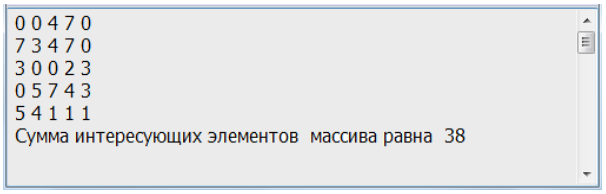

Рис. 4.2. Вывод информации в программе, приведенной в листинге 4.2

Расчет среднего арифметического в строках

В данном случае нам требуется вычислить среднее арифметическое значение отдельно по каждой строке двумерного массива. В листинге 4.3 приведена программа, реализующая решение данной задачи. Здесь мы использовали переменную `S` — для среднего арифметического интересующих нас элементов.

Результат работы такой программы приведен на рис. 4.3.

Листинг 4.3. Вычисление среднего арифметического

```
Module Module1
    Sub Main()
        Const N = 5
        Dim A(5, 5) As Integer
        Dim I, J As Integer
        Dim S As Single
        Randomize()
        For I = 1 To N
            For J = 1 To N
                A(I, J) = Int(10 * Rnd())
                Console.Write(A(I, J))
                Console.Write(" ")
            Next
            Console.WriteLine()
        Next
        For I = 1 To N
            S = 0
            For J = 1 To N
                S = S + A(I, J)
            Next
            S = S / N
            Console.Write("Строка ")
            Console.Write(I)
            Console.Write(". Среднее арифметическое в строке равно ")
            Console.WriteLine(S)
        Next
        Console.Read()
    End Sub
End Module
```

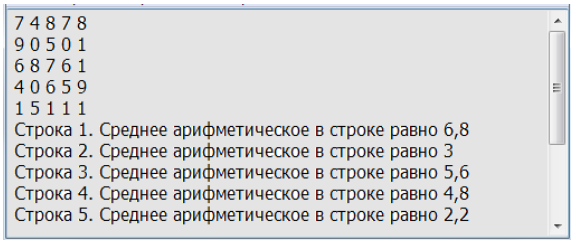

Рис. 4.3. Вывод информации в программе, приведенной в листинге 4.3

Поиск минимального элемента

В листинге 4.4 приведена программа поиска минимального элемента в двумерном массиве (N на M элементов), заполненном целыми числами. Под поиском подразумевается нахождение номеров (индексов) строки и столбца необходимого элемента в двумерном массиве. При этом если минимальных элементов несколько, то достаточно найти индексы только одного из них.

Результат работы такой программы приведен на рис. 4.4.

Листинг 4.4. Поиск индексов минимального элемента в двумерном массиве

```
Module Module1
    Sub Main()
        Const N = 5
        Const M = 7
        Dim A(5, 7) As Integer
        Dim I, J, Minimum, Imin, Jmin As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(10 * Rnd())
                Console.Write(A(I, J))
                Console.Write(" ")
            Next
            Console.WriteLine()
        Next
        Minimum = A(1, 1)
        Imin = 1
        Jmin = 1
        For I = 1 To N
            For J = 1 To M
                If A(I, J) < Minimum Then
                    Minimum = A(I, J)
                    Imin = I
                    Jmin = J
                End If
            Next
        Next
        Console.Write("I= ")
        Console.Write(Imin)
        Console.Write("J= ")
        Console.Write(Jmin)
        Console.Read()
    End Sub
End Module
```

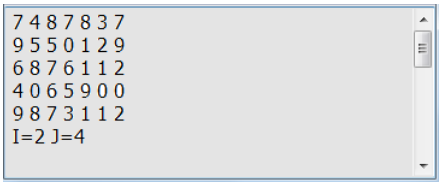

Рис. 4.4. Вывод информации в программе, приведенной в листинге 4.4

Здесь и в предыдущих примерах при объявлении массива мы сразу задавали его размерность, т. е. сразу отводили в памяти необходимое место. Однако, как было указано в *главе 3*, можно выделять память в процессе выполнения (динамически). Для демонстрации этого ресурса изменим предыдущую разработку в соответствии с листингом 4.5.

Листинг 4.5. Поиск индексов минимального элемента в двумерном массиве (вариант 2)

```
Module Module1
    Sub Main()
        Const N = 5
        Const M = 7
        Dim A(,) As Integer
        Dim I, J, Minimum, Imin, Jmin As Integer
        ReDim A(N, M)
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(10 * Rnd())
                Console.Write(A(I, J))
                Console.Write(" ")
            Next
            Console.WriteLine()
        Next
        Minimum = A(1, 1)
        Imin = 1
        Jmin = 1
        For I = 1 To N
            For J = 1 To M
                If A(I, J) < Minimum Then
                    Minimum = A(I, J)
                    Imin = I
                    Jmin = J
                End If
            Next
        Next
        Console.Write("I= ")
        Console.Write(Imin)
        Console.Write("J= ")
```

```
        Console.Write(Jmin)
        Console.Read()
    End Sub
End Module
```

Изменим предыдущую программу так, чтобы в результате на экран выводились индексы всех минимальных элементов. В этом случае (листинг 4.6) нам потребуется дополнительный цикл для просмотра всех элементов.

Результат работы такой программы приведен на рис. 4.5.

Листинг 4.6. Поиск индексов всех минимальных элементов в двумерном массиве

```
Module Module1
    Sub Main()
        Const N = 5
        Const M = 7
        Dim A( , ) As Integer
        Dim I, J, Minimum As Integer
        ReDim A(N, M)
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(10 * Rnd())
                Console.Write(A(I, J))
                Console.Write(" ")
            Next
            Console.WriteLine()
        Next
        Minimum = A(1, 1)
        For I = 1 To N
          For J = 1 To M
                If A(I, J) < Minimum Then
                    Minimum = A(I, J)
                End If
            Next
        Next
        For I = 1 To N
            For J = 1 To M
                If A(I, J) = Minimum Then
                    Console.Write("I= ")
                    Console.Write(I)
                    Console.Write("J= ")
                    Console.WriteLine(J)
                End If
            Next
        Next
```

```
        Console.Read()
    End Sub
End Module
```

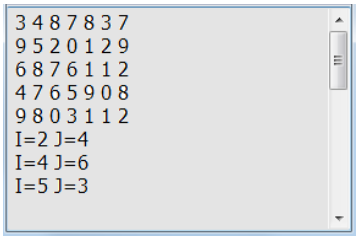

Рис. 4.5. Вывод информации в программе, приведенной в листинге 4.5

Поиск номера строки с минимальной суммой

Рассмотрим задачу поиска номера строки массива, для которой сумма элементов минимальна. Текст программы показан в листинге 4.7.

Результат работы такой программы приведен на рис. 4.6.

Листинг 4.7. Поиск номера строки с минимальной суммой элементов

```
Module Module1
    Sub Main()
        Const N = 5
        Const M = 7
        Dim A(5, 7) As Integer
        Dim I, J, Imin, Sum, SumMin As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(10 * Rnd())
                Console.Write(A(I, J))
                Console.Write(" ")
            Next
            Console.WriteLine()
        Next
        For I = 1 To N
            Sum = 0
            For J = 1 To M
                Sum = Sum + A(I, J)
                If I = 1 Then
                    Imin = 1
                    SumMin = Sum
```

```
                Else
                    If (SumMin > Sum) Then
                        SumMin = Sum
                        Imin = I
                    End If
                End If
            Next
        Next
        Console.Write("Номер строки с минимальной суммой равен ")
        Console.WriteLine(Imin)
        Console.Read()
  End Sub
End Module
```

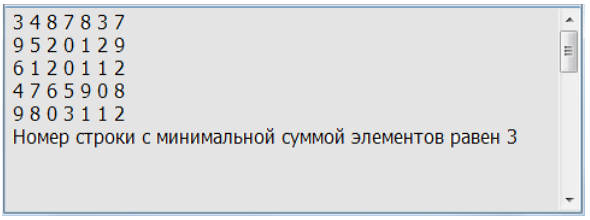

Рис. 4.6. Вывод информации в программе, приведенной в листинге 4.7

Подсчет числа учащихся

В двумерном массиве хранится информация о количестве учеников в каждом классе всех параллелей школы (параллели нумеруются от 1 до 11). Так, в каждой строке двумерного массива сначала располагается число классов в определенной параллели, а затем последующие элементы содержат число учащихся в каждом классе данной параллели. Пример одной из строк двумерного массива выглядит так:

4 25 23 27 28,

что означает четыре класса в параллели и, соответственно, 25 учеников в первом из классов параллели, 23 ученика во втором классе параллели и т. д.

Необходимо написать программу, которая позволит автоматически определить общее число учащихся в классах с 5-й по 10-ю параллель включительно. Для определенности будем считать, что в каждой параллели не более пяти классов. В листинге 4.8 приведена программа, решающая данную задачу. Для заполнения информации о числе классов в параллелях и числа учащихся мы использовали датчик случайных чисел. Число учащихся в классе варьируется от 20 до 30.

Листинг 4.8. Подсчет числа учащихся в параллелях

```
Module Module1
    Sub Main()
        Const N = 11
```

```
        Const M = 6
        Dim A(11, 6) As Integer
        Dim I,J,Schet As Integer
        Randomize()
        For I = 1 To N
             A(I, 1) = Int(5 * Rnd())+1
        Next
        For I=1 To N
           For J=1 To A(I,1)
               A(I,J+1)=20+ Int(11 * Rnd())
         Next
       Next
       Schet=0
       For I=5 to 10
           For J=1 to A(I,1)
              Schet=Schet + A(I,J+1)
           Next
       Next
      Console.Write("Число учащихся равно ")
       Console.WriteLine(Schet)
       Console.Read()
   End Sub
End Module
```

Изменим программу так, чтобы подсчет осуществлялся по интервалу параллелей, значения которых вводятся с клавиатуры. В листинге 4.9 приведена необходимая программа.

Листинг 4.9. Подсчет числа учащихся по произвольному интервалу параллелей

```
Module Module1
    Sub Main()
        Const N = 11
        Const M = 6
        Dim A(11, 6) As Integer
        Dim I,J,Schet,P1,P2 As Integer
        Console.WriteLine("Введите значение P1")
        P1 = Console.ReadLine()
        Console.WriteLine("Введите значение P2")
        P2 = Console.ReadLine()
        For I = 1 To N
             A(I, 1) = Int(5 * Rnd())+1
        Next
        For I=1 To N
             For J=1 To A(I,1)
                 A(I,J+1)=20+ Int(11 * Rnd())
             Next
```

```
        Next
        Schet=0
        For I=P1 to P2
           For J=1 to A(I,1)
               Schet=Schet+ A(I,J+1)
           Next
        Next
        Console.Write("Число учащихся равно ")
        Console.WriteLine(Schet)
        Console.Read()
End Sub
End Module
```

Определение результата турнира

В двумерном массиве (N на N элементов) хранится информация о результатах хоккейного турнира. В турнире участвовали N команд и элемент массива A(I, J) содержит число очков, которые набрала I-я команда в игре на своем поле во встрече с J-й командой. Соответственно, элемент A(J, I) содержит число очков, которые набрала J-я команда при встрече с I-й на своем поле. Будем считать, что за победу команде дается 2 очка, в случае ничьей — одно очко, а при поражении команда очков не получает.

Необходимо определить чемпиона (или несколько команд, набравших максимальное количество очков). В листинге 4.10 приведена программа, позволяющая определить победителя (или победителей) турнира. Здесь мы сначала обеспечиваем формирование результатов турнира с помощью датчика случайных чисел. После этого выполняется подсчет числа очков, набранных первой командой. И этот результат принимается за максимум, который фиксируется в переменной MaxBalls. Далее последовательно вычисляются набранные очки другими командами. В результате мы получаем максимальное количество очков по итогам турнира, которые набрала команда-чемпион. При этом в дополнительном массиве B фиксируются результаты, набранные каждой командой.

Листинг 4.10. Определение победителя турнира

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10, 10) As Integer
        Dim B(10) As Integer
        Dim I, J, MaxBalls, Vr_Max As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To N
```

```
                If I <> J Then
                    A(I, J) = Int(3 * Rnd())
                Else
                    A(I, J) = 0
                End If
               Console.Write(A(I, J))
               Console.Write(" ")
            Next
               Console.WriteLine()
        Next
        MaxBalls = 0
        For I = 2 To N
            MaxBalls = MaxBalls + A(1, I)
        Next
        For I = 2 To N
            MaxBalls = MaxBalls + 2 — A(I, 1)
        Next
        B(1) = MaxBalls
        For I = 2 To N
            Vr_Max = 0
            For J = 1 To N
                If I <> J Then
                    Vr_Max = Vr_Max + A(I, J)
                End If
            Next
            For J = 1 To N
                If I <> J Then
                    Vr_Max = Vr_Max + 2 — A(J, I)
                End If
            Next
            B(I) = Vr_Max
            If Vr_Max > MaxBalls Then
                MaxBalls = Vr_Max
            End If
        Next
        For I = 1 To N
            If B(I) = MaxBalls Then
                Console.WriteLine(I)
            End If
        Next
        Console.Read()
    End Sub
End Module
```

Расчет доходов по отделу

В двумерном массиве хранится информация о доходах отдела (10 человек) за каждый месяц текущего года. Необходимо подсчитать общую сумму по отделу за каждый квартал. В листинге 4.11 приведена необходимая программа, где после внесения оплат осуществляется группировка данных по кварталам. Отметим назначение двух переменных:

- `L1` — для обозначения первого месяца квартала;
- `L2` — для обозначения последнего месяца квартала.

Результат работы такой программы приведен на рис. 4.7.

Листинг 4.11. Расчет доходов по кварталам

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 12
        Dim A(10, 12) As Integer
        Dim I, J, L1, L2, L, Sum As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(500 * Rnd()) + 7000
            Next
        Next
        For I = 1 To 4
            Sum = 0
            L1 = 1 + (I - 1) * 3
            L2 = 3 + (I - 1) * 3
            For J = 1 To N
                For L = L1 To L2
                    Sum = Sum + A(J, L)
                Next
            Next
            Console.Write(I)
            Console.Write(" ")
            Console.WriteLine(Sum)
        Next
        Console.Read()
    End Sub
End Module
```

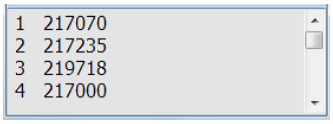

Рис. 4.7. Вывод информации в программе, приведенной в листинге 4.11

Анализ средней зарплаты сотрудников

В двумерном массиве хранится информация о доходах отдела (10 человек) за каждый месяц в течение года. Необходимо подсчитать число сотрудников, у которых доходы за год оказались выше средних по отделу. В листинге 4.12 приведена необходимая программа.

Листинг 4.12. Подсчет числа сотрудников по условию

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 12
        Dim A(10, 12) As Integer
        Dim I, J, Col As Integer
        Dim Srednee, Sr As Single
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(500 * Rnd()) + 7000
            Next
        Next
        Srednee = 0
        For I = 1 To N
            For J = 1 To M
                Srednee = Srednee + A(I, J)
            Next
        Next
        Srednee = Srednee / 10
        Col = 0
        For I = 1 To N
            Sr = 0
            For J = 1 To M
                Sr = Sr + A(I, J)
            Next
            If Sr > Srednee Then
                Col = Col + 1
            End If
        Next
        Console.Write("Количество = ")
        Console.WriteLine(Col)
        Console.Read()
    End Sub
End Module
```

В листинге 4.13 приведен более экономичный вариант предыдущей программы. А именно заполнение данными массива выполняется вместе с подсчетом среднего значения.

Листинг 4.13. Подсчет числа сотрудников по условию

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 12
        Dim A(10, 12) As Integer
        Dim I, J, Col As Integer
        Dim Srednee, Sr As Single
        Randomize()
        Srednee = 0
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(500 * Rnd()) + 7000
                Srednee = Srednee + A(I, J)
            Next
        Next
        Srednee = Srednee / 10
        Col = 0
        For I = 1 To N
            Sr = 0
            For J = 1 To M
                Sr = Sr + A(I, J)
            Next
            If Sr > Srednee Then
                Col = Col + 1
            End If
        Next
        Console.Write("Количество = ")
        Console.WriteLine(Col)
        Console.Read()
    End Sub
End Module
```

Подсчет элементов по условию

Задан двумерный числовой массив размерности `N` на `N`. Требуется подсчитать число положительных элементов, размещенных выше главной диагонали, что располагаются от элемента `A(1,1)` до элемента `A(N,N)`. В листинге 4.14 приведена необходимая программа.

Просмотр элементов, расположенных выше главной диагонали, реализуется с помощью следующего вложенного цикла:

```
For I = 1 To N
    For J = I + 1 To N
```

Листинг 4.14. Подсчет числа элементов по условию

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10, 10) As Integer
        Dim I, J, Cols As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To N
                A(I, J) = -50 + Int(100 * Rnd())
            Next
        Next
        Cols = 0
        For I = 1 To N
            For J = I + 1 To N
                If A(I, J) > 0 Then
                    Cols = Cols + 1
                End If
            Next
        Next
        Console.Write("Количество = ")
        Console.WriteLine(Cols)
        Console.Read()
    End Sub
End Module
```

Подсчет суммы элементов по условию

Задан двумерный числовой массив размерности `N` на `N`. Требуется подсчитать сумму отрицательных элементов, размещенных выше главной диагонали, которая располагается от элемента `A(1,N)` до элемента `A(N,1)`. В листинге 4.15 приведена необходимая программа.

Листинг 4.15. Подсчет суммы элементов по условию

```
Module Module1
    Sub Main()
        Const N = 10
        Dim A(10, 10) As Integer
        Dim I, J, Sum As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To N
                A(I, J) = -50 + Int(100 * Rnd())
            Next
        Next
```

```
        Sum = 0
        For I = 1 To N
            For J = 1 To N — I
                If A(I, J) < 0 Then
                    Sum = Sum + 1
                End If
            Next
        Next
        Console.WriteLine(Sum)
        Console.Read()
    End Sub
End Module
```

Нахождение индексов элементов

Задан двумерный числовой массив размерностью N на M. Нужно вывести на экран индексы первых (начиная с начальных столбцов) положительных элементов в каждой строке. В листинге 4.16 приведена необходимая программа.

Листинг 4.16. Нахождение индексов первых положительных элементов

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 10
        Dim A(10, 20) As Integer
        Dim I, J As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = -50 + Int(100 * Rnd())
            Next
        Next
        For I = 1 To N
            For J = 1 To M
                If A(I, J) > 0 Then
                    Console.Write(I)
                    Console.Write(" ")
                    Console.WriteLine(J)
                    Exit For
                End If
            Next
        Next
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще одну похожую ситуацию. Пусть задан двумерный числовой массив размерностью N на M. Необходимо вывести на экран индексы последних отрицательных элементов в каждой строке. В листинге 4.17 приведена соответствующая этой задаче программа.

Листинг 4.17. Нахождение индексов отрицательных элементов

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 20
        Dim A(10, 20) As Integer
        Dim I, J As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = -50 + Int(100 * Rnd())
            Next
        Next
        For I = 1 To N
            For J = M To 1 Step -1
                If A(I, J) < 0 Then
                    Console.Write(I)
                    Console.Write(" ")
                    Console.WriteLine(J)
                    Exit For
                End If
            Next
        Next
        Console.Read()
    End Sub
End Module
```

Нахождение уникальных элементов

Задан двумерный числовой массив размерностью N на M. Требуется вывести на экран только уникальные элементы в каждой строке. Так, если в определенной строке есть одинаковые элементы, то их выводить не надо. В листинге 4.18 приведена необходимая программа. Здесь перед выводом очередного элемента мы предварительно анализируем предыдущие элементы в данной строке.

Листинг 4.18. Нахождение уникальных элементов в каждой строке

```
Module Module1
    Sub Main()
        Const N = 10
```

```
        Const M = 10
        Dim A(10, 10) As Integer
        Dim I, J, L, Flag As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(10 * Rnd())
            Next
        Next
        For I = 1 To N
            For J = 1 To M
                Flag = 0
                For L = 1 To J — 1
                    If A(I, J) = A(I, L) Then
                        Flag = 1
                        Exit For
                    End If
                Next
                If Flag = 0 Then
                    Console.Write(" ")
                    Console.Write(A(I, J))
                End If
            Next
            Console.WriteLine()
        Next
        Console.Read()
    End Sub
End Module
```

Анализ тестирования

В двумерном массиве размерностью N на M хранится информация о результатах тестирования среди учащихся по пяти темам. Требуется вывести номера участников (номера строк двумерного массива), которые в сумме набрали более 100 баллов. В листинге 4.19 приведена необходимая для решения этой задачи программа.

Листинг 4.19. Анализ тестирования учащихся

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 5
        Dim A(10, 5) As Integer
        Dim I, J, Sum As Integer
        Randomize()
```

```
        For I = 1 To N
            For J = 1 To M
                A(I, J) = Int(50 * Rnd())
            Next
        Next
        For I = 1 To N
            Sum = 0
            For J = 1 To M
                Sum = Sum + A(I, J)
            Next
            If Sum > 100 Then
                Console.WriteLine(I)
            End If
        Next
        Console.Read()
    End Sub
End Module
```

Изменение знака элементов

В двумерном массиве размерностью `N` на `M` элементов требуется сделать следующее преобразование: в каждой строке у последнего отрицательного элемента следует поменять знак. В листинге 4.20 приведена необходимая для решения данной задачи программа.

Листинг 4.20. Изменение знака у правых крайних отрицательных элементов

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 20
        Dim A(10, 20) As Integer
        Dim I, J As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = -50 + Int(100 * Rnd())
            Next
        Next
        For I = 1 To N
            For J = M To 1 Step -1
                If A(I, J) < 0 Then
                    A(I, J) = -A(I, J)
                    Exit For
                End If
```

```
            Next
        Next
    End Sub
End Module
```

Изменение элементов по условию

В двумерном массиве размерностью N на M элементов требуется сделать следующее преобразование: в каждой строке максимальный элемент умножить на 10. В листинге 4.21 приведена необходимая для этого программа.

Листинг 4.21. Изменение элементов по условию

```
Module Module1
    Sub Main()
        Const N = 10
        Const M = 20
        Dim A(10, 20) As Integer
        Dim I, J, Maxsim As Integer
        Randomize()
        For I = 1 To N
            For J = 1 To M
                A(I, J) = -50 + Int(100 * Rnd())
            Next
        Next
         For I = 1 To N
            Maxsim = A(I, 1)
            For J = 2 To M
                If A(I, J) > Maxsim Then
                    Maxsim = A(I, J)
                End If
            Next
            For J = 1 To M
                If A(I, J) = Maxsim Then
                    A(I, J) = A(I, J) * 10
                End If
            Next
        Next
    End Sub
End Module
```

Тур коня на шахматной доске

Рассмотрим задачу, которая заключается в разработке программы, моделирующей ходы коня на шахматной доске. Сформулируем словесное описание разработки. Пользователь выбирает произвольное положение шахматной фигуры, а затем про-

грамма автоматически продвигает коня по шахматной доске с условием, что дважды побывать на одной клетке конь не может. Таким образом, маршрут коня завершается, когда нет возможности перехода с текущего поля ни на одну, еще не посещенную клетку.

В качестве аналога шахматной доски выберем двумерный массив 8 на 8 элементов.

Как известно, в шахматах конь может пойти из заданного поля максимум на восемь других полей доски. Если фигура расположена на краю доски, то число возможных ходов уменьшается. Также если поле для хода уже занято, то конь пойти на него не может.

Варианты ходов шахматного коня с произвольного поля показаны на рис. 4.8, при этом нумерация вариантов в дальнейшем будет отражаться в алгоритме маршрута именно в таком виде. Например, вариант с номером 2 связан с перемещением фигуры на одну клетку вверх и две клетки вправо.

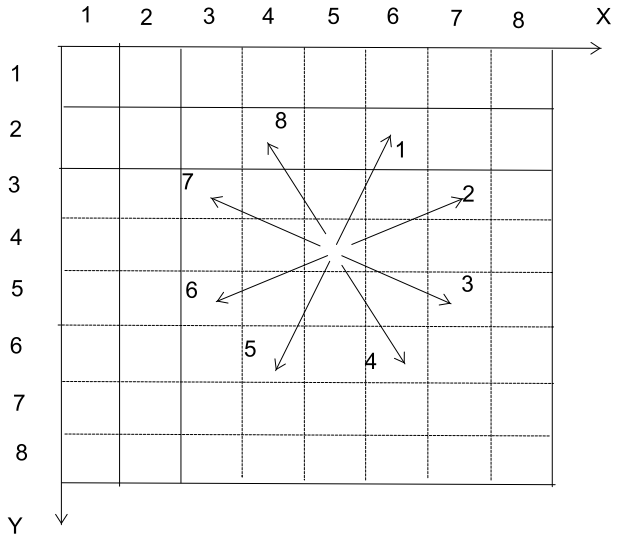

Рис. 4.8. Возможные ходы коня

Сформулируем алгоритм выбора очередного хода. В цикле последовательно просматриваются возможные варианты ходов из текущего поля. Просмотр возможных полей осуществляется последовательно, начиная с первого варианта. Если очередной рассматриваемый вариант хода возможен (данная клетка располагается в пределах доски и конь на этой клетке еще не был), то выполняется перемещение фигуры в соответствии с этим вариантом. Если ход невозможен, то просматривается следующий вариант и далее, вплоть до восьмого. Если ни один ход из текущей клетки невозможен, то тур шахматного коня на этом завершается.

Для программного запоминания ходов коня нам потребуется двумерный массив `A(N, N)`. Возможные ходы коня будем называть вариантами, а для обозначения вариантов введем переменную `K`, которая может принимать значения от 1 до 8 (табл. 4.1).

Таблица 4.1. *Данные для выполнения хода шахматного коня*

K	Новое значение Y	Новое значение X	Значение элемента массива Dy(K)	Значение элемента массива Dx(K)
1	Y-2	X+1	-2	1
2	Y-1	X+2	-1	2
3	Y+1	X+2	1	2
4	Y+2	X+1	2	1
5	Y+2	X-1	2	-1
6	Y+1	X-2	1	-2
7	Y-1	X-2	-1	-2
8	Y-2	X-1	-2	-1

Например, при третьем варианте хода (K=3) к текущему значению координаты X добавляется значение элемента массива Dx(3), а к текущему значению координаты Y добавляется значение Dy(3).

Программная реализация представлена в листинге 4.22.

Листинг 4.22. Тур шахматного коня

```
Module Module1

    Sub Main()
        Dim Dx() As Integer = {0, 1, 2, 2, 1, -1, -2, -2, -1}
        Dim Dy() As Integer = {0, -2, -1, 1, 2, 2, 1, -1, -2}
        Dim Mass(8, 8) As Integer
        Dim I, J, K, X, Y, VarX, VarY, Nom As Integer
        For J = 1 To 8
            For I = 1 To 8
                Mass(I, J) = 0
            Next
        Next
        Console.WriteLine("Ввод номера поля по горизонтали")
        X = Console.ReadLine()
        Console.WriteLine("Ввод номера поля по вертикали ")
        Y = Console.ReadLine()
        Nom = 1
        K = 1
        Mass(Y, X) = Nom
        Do
            VarX = X + Dx(K)
            VarY = Y + Dy(K)
            If (VarX < 1) Or (VarX > 8) Or (VarY < 1) Or (VarY > 8) Then
                K = K + 1
```

```
            Else
                If Mass(VarY, VarX) > 0 Then
                    K = K + 1
                Else
                    X = VarX
                    Y = VarY
                    Nom = Nom + 1
                    Mass(Y, X) = Nom
                    K = 1
                End If
            End If
        Loop While K <= 8
        For I = 1 To 8
            Console.WriteLine()
            For J = 1 To 8
                If Mass(I, J) = 0 Then
                    Console.Write("   ")
                ElseIf Mass(I, J) <= 9 Then
                    Console.Write("  ")
                    Console.Write(Mass(I, J))
                Else
                    Console.Write(" ")
                    Console.Write(Mass(I, J))
                End If
            Next
        Next
        Console.Read()
    End Sub
End Module
```

Задания из ЕГЭ

В заданиях данного раздела требуется проанализировать программный код без использования компьютера и сделать правильный вывод. В последнем разделе главы приведены правильные ответы.

ЗАДАЧА 4.1

Дан фрагмент программы, который представлен в листинге 4.23. Чему равно значение `C(4,3)`, если перед приведенными командами значение `C(4,3)=10`.

Листинг 4.23. Фрагмент программы к задаче 4.1

```
For N=1 To 6
 For M=1 To 5
     C(N,M)= C(N,M)+2*N-M
  Next
Next
```

ЗАДАЧА 4.2

Значения элементов двумерного массива A размером 5 на 5 задаются с помощью вложенного цикла во фрагменте, представленном в листинге 4.24. Сколько элементов массива будут больше 10?

Листинг 4.24. Фрагмент к заданию 4.2

```
For I=1 To 5
  For J=1 To 5
     A(I,J)= I * J
  Next
Next
```

ЗАДАЧА 4.3

Значения элементов двумерного массива A размером 5 на 5 задаются с помощью вложенного цикла во фрагменте, представленном в листинге 4.25. Сколько элементов массива будут больше 2?

Листинг 4.25. Фрагмент к задаче 4.3

```
For I=1 To 5
  For J=1 To 5
     A(I,J)= I Mod J
  Next
Next
```

ЗАДАЧА 4.4

Значения элементов двумерного массива размером 7 на 7 задаются с помощью вложенного цикла во фрагменте, представленном в листинге 4.26. Сколько элементов массива будут иметь значения больше 0?

Листинг 4.26. Фрагмент к задаче 4.4

```
For I=1 To 7
  For J=1 To 7
     B(I,J)= J - I
  Next
Next
```

ЗАДАЧА 4.5

Значения элементов двумерного массива A размером 4 на 4 первоначально были равны нулю. Затем значения элементов меняются с помощью вложенного оператора цикла во фрагменте программы, представленном в листинге 4.27. Сколько элементов массива будут равны 1?

Листинг 4.27. Фрагмент к задаче 4.5

```
For N=1 To 4
  For K=N to 4
      A(N,K)= A(N,K) + 1
      A(K,N)= A(K,N) + 1
      Next
Next
```

ЗАДАЧА 4.6

Дан фрагмент программы, который представлен в листинге 4.28. Чему равно значение `C(3,5)` после выполнения данного фрагмента.

Листинг 4.28. Фрагмент программы к задаче 4.6

```
C(3,4)= 5
C(2,5)= 7
C(2,4)= 2
For N=3 To 4
   For M=4 To 5
       C(N,M)= C(N-1,M)+ C(N,M)
   Next
Next
```

ЗАДАЧА 4.7

Значения элементов двумерного массива `A` размером 4 на 4 задаются с помощью вложенного цикла во фрагменте, представленном в листинге 4.29. Сколько элементов массива будут меньше 5?

Листинг 4.29. Фрагмент к заданию 4.7

```
For I=1 To 4
    For J=1 To 4
        A(I,J)= I * J+I
     Next
Next
```

ЗАДАЧА 4.8

Значения элементов двумерного массива `A` размером 3 на 3 задаются с помощью вложенного цикла во фрагменте, представленном в листинге 4.30. Сколько элементов массива будут больше 2?

Листинг 4.30. Фрагмент к задаче 4.8

```
For I=1 To 3
   For J=1 To 3

       A(I,J)= I \ J Mod J
  Next
Next
```

ЗАДАЧА 4.9

Значения элементов двумерного массива размером 5 на 5 задаются с помощью вложенного цикла во фрагменте, представленном в листинге 4.31. Сколько элементов массива будут иметь значения, равные 0?

Листинг 4.31. Фрагмент к задаче 4.9

```
For I=1 To 5
   For J=1 To 5
     B(I,J)= J Mod I
  Next
Next
```

ЗАДАЧА 4.10

Значения элементов двумерного массива `A` размером 4 на 4 первоначально были равны единице. Затем значения элементов меняются с помощью вложенного оператора цикла во фрагменте программы, представленном в листинге 4.32. Сколько элементов массива будут равны 0?

Листинг 4.32. Фрагмент к задаче 4.10

```
For N=1 To 4
   For K=1 To 4
     A(N,K)= A(K,N) — 1
   Next
Next
```

ЗАДАЧА 4.11

Дан фрагмент программы (листинг 4.33), обрабатывающей двумерный массив `A` размером `N` на `N`. Представим массив в виде квадратной таблицы, в которой для элемента массива `A(I,J)` величина `I` является номером строки, а величина `J` номером столбца, в котором расположен элемент. Что делает рассматриваемый алгоритм:

- ❒ меняет местами элементы двух диагоналей;
- ❒ меняет местами элементы диагонали и `K`-й строки;

- меняет местами элементы диагонали и K-го столбца;
- меняет местами элементы K-й строки и K-го столбца.

Листинг 4.33. Фрагмент к задаче 4.11

```
K=1
For I=1 to N
   C= A(I,K)
   A(I,K)= A(K,I)
   A(K,I)= C
Next
```

Ответы к задачам и заданиям из ЕГЭ

Задача 4.1

Здесь требуется только подставить числа (N=4 и M=3) и выполнить арифметическое вычисление: 10 + (2 * 4 – 3) = 15.

Задача 4.2

Всего в массиве 25 элементов и требуется проанализировать результат вычисления каждого элемента массива. Так, результаты больше 10 дают произведения: 5 на 5; 4 на 5; 5 на 4; 4 на 4; 5 на 3; 3 на 5; 4 на 3; 3 на 4. В результате получаем, что число интересующих нас элементов массива равно 8.

Задача 4.3

Всего в массиве 25 элементов и требуется проанализировать результат вычисления остатка от деления I на J. Так, результаты больше 2 дают остатки от деления: 3 на 4; 3 на 5; 4 на 5. В результате получаем ответ 3.

Задача 4.4

Проанализируем двумерный массив по строкам. В первой строке (при I=1) таких элементов насчитывается 6 штук, во второй (при I=2) таких элементов 5 штук. Продолжая анализ следующих строк, получим ответы: 4, 3, 2, 1, 0.

В итоге получается, что в массиве число положительных элементов равно 6 + 5 + 4 + 3 + 2 + 1 = 21 значение.

Задача 4.5

Любой способ решения заключается в последовательном анализе выполнения циклов. В табл. 4.2 приведен один из вариантов такого анализа. Здесь первые два столбца соответствуют очередным значениям счетчиков циклов, а последующие два других столбца соответствуют значениям A(N,K) и A(K,N), которые вычисляют-

ся в цикле (заметим, что вложенный цикл вычисляется от N до 4). Таким образом, всего в массиве 6 элементов, которые равны 1.

Таблица 4.2. *Решение задачи 4.5*

N	K	A(N,K)	A(K,N)
1	1	A(1,1)=1	A(1,1)=2
1	2	A(1,2)=1	A(2,1)=1
1	3	A(1,3)=1	A(3,1)=1
1	4	A(1,4)=1	A(4,1)=1
2	2	A(2,2)=1	A(2,2)=2
2	3	A(2,3)=1	A(3,2)=1
2	4	A(2,4)=1	A(4,2)=1
3	3	A(3,3)=1	A(3,3)=2
3	4	A(3,4)=1	A(4,3)=1
4	4	A(4,4)=1	A(4,4)=2

Однако можно предложить и другой вариант анализа (табл. 4.3). В этом случае мы формируем таблицу размером 4 на 4 (визуальное представление массива 4 на 4) и начинаем последовательно выполнять шаги, указанные в циклах. После очередного шага в соответствии с циклами мы корректируем значения рассматриваемых элементов. После заполнения таблицы простой пересчет приводит к тому же значению, равному 6.

Таблица 4.3. *Второй вариант решения задачи 4.5*

N	K=1	K=2	K=3	K=4
N =1	A(1,1)=2	A(1,2)=1	A(1,3)=1	A(1,4)=1
N =2	A(2,1)=0	A(2,2)=2	A(2,3)=1	A(2,4)=1
N =3	A(3,1)=0	A(3,2)=0	A(3,3)=2	A(3,4)=1
N =4	A(4,1)=0	A(4,2)=0	A(4,3)=0	A(4,4)=2

Задача 4.6

Здесь требуется проанализировать работу алгоритма. Так как нас интересует элемент из третьей строки (элемент C(3,5)), то рассмотрим выполнение внутреннего цикла (по M) при N=3. Так, при M=4 получим:

```
C(3,4)= C(2,4)+ C(3,4),
```

что после подстановки приводит к результату, равному 7.

Следующее выполнение внутреннего цикла приводит к результату:

```
C(3,5)= C(2,5)+ C(3,5),
```

что дает ответ 14.

Задача 4.7

Всего в массиве 16 элементов и требуется проанализировать результат вычисления каждого элемента массива. Так, результат, меньший 7, дают значения:

```
I=1, J=1;
I=1, J=2;
I=2, J=1;
I=2, J=2;
I=1, J=3;
I=3, J=1.
```

В итоге получаем, что число интересующих нас элементов массива равно 6.

Задача 4.8

Всего в массиве 25 элементов и требуется проанализировать результат вычисления выражения, где используется целочисленное деление и остаток от деления.

Так, вычисления дают:

```
1 \ 1 mod 1 = 0;
1 \ 2 mod 2 = 2;
1 \ 3 mod 3 = 3;
2 \ 1 mod 1 = 0;
2 \ 2 mod 2 = 2;
2 \ 3 mod 3 = 3;
3 \ 1 mod 1 = 0;
3 \ 2 mod 2 = 2;
3 \ 3 mod 3 = 3.
```

Из этого можно сделать вывод о том, что в массиве имеются три элемента, значения которых больше 2.

Задача 4.9

Проанализируем двумерный массив по строкам. В первой строке (при `I=1`) таких элементов (равных 0) насчитывается 5 штук, во второй (при `I=2`) 2 штуки. Продолжая анализ следующих строк, получим: 1, 1, 1. В итоге: 5 + 2 + 1 + 1 + 1 = 10 элементов массива имеют нулевые значения.

Задача 4.10

Рассмотрим вариант анализа, основанный на данных табл. 4.4. В этом случае мы формируем таблицу размером 4 на 4 (визуальное представление массива 4 на 4) и начинаем последовательно выполнять шаги, указанные в циклах. После очередного шага в соответствии с циклами корректируем значения рассматриваемых элементов. После заполнения таблицы простой пересчет приводит к результату: 10 элементов массива имеют нулевые значения.

Таблица 4.4. *Решение задачи 4.10*

N	K=1	K=2	K=3	K=4
N =1	A(1,1)= 0	A(1,2)=0	A(1,3)=0	A(1,4)=0
N =2	A(2,1)=-1	A(2,2)=0	A(2,3)=0	A(2,4)=0
N =3	A(3,1)=-1	A(3,2)=-1	A(3,3)=0	A(3,4)=0
N =4	A(4,1)=-1	A(4,2)=-1	A(4,3)=-1	A(4,4)=0

Задача 4.11

Программа выполняет вычисление в одном цикле по переменной `I`. Значение индекса `K` в программе не меняется. Рассмотрим элемент `A(2,1)`. Этот элемент расположен в 1-м столбце 2-й строки. Как видно из алгоритма, его значение меняется на значение элемента `A(1,2)`. Соответственно в `A(1,2)` записывается исходное значение `A(2,1)`. Аналогично меняются местами элементы `A(3,1)` и `A(1,3)` и т. д. Таким образом, выполняется замена элементов первого столбца на элементы первой строки. В результате подходит версия ответа — обмен значений элементов `K`-й строки и `K`-го столбца.

ГЛАВА 5

Строки

Значительная часть информации, с которой мы работаем, представляет собой текст. Текст фактически является набором символов. Для работы с такими данными в Бейсике имеется специальный тип данных — `String`. Если в программе создать переменную данного типа, то в ней можно хранить текстовую строку длиной от 0 до двух миллиардов символов в кодировке Unicode [1].

В этой главе мы рассмотрим различные приемы работы со строками. При этом познакомимся как со стандартными функциями, предназначенными для работы со строками, так разработаем несколько своих алгоритмов обработки строк.

В *предыдущих главах* мы уже упоминали о таком типе данных, как строка. Также был рассмотрен ряд примеров, касающихся несложных действий над строками. Здесь эта тема будет продолжена и мы разберем приемы работы со строковыми данными. Также на практических примерах познакомимся с типовыми заданиями, касающимися работы со строками.

Действия над строками

Над строками допустимы следующие операции:

- сцепление (соединение или по-другому сложение нескольких строк);
- присваивание;
- отношения, которые используются для сравнения;
- ввод/вывод.

Для иллюстрации действий над строками в листинге 5.1 приведен пример использования ввода/вывода, присваивания и сложения строковых переменных. Сложение строк и присваивание используется в следующей программной строке данного листинга:

```
D = A + " " + B + " " + " " + C
```

Здесь выполняется соединение пяти строк (трех строк входных данных и двух строк, каждая из которых состоит из одного пробела.

Листинг 5.1. Пример действий над строками

```
Module Module1
    Sub Main()
        Dim A, B, C, D As String
        Console.WriteLine("Введите фамилию")
        A = Console.ReadLine()
        Console.WriteLine("Введите имя")
        B = Console.ReadLine()
        Console.WriteLine("Введите отчество")
        C = Console.ReadLine()
        D = A + " " + B + " " + " " + C
        Console.WriteLine(D)
        Console.Read()
    End Sub
End Module
```

Частое действие, встречающееся в программном коде, связано со сравнением строк. Здесь никакой новизны относительно ранее рассмотренных примеров с использованием сравнения (встречалось в предыдущих главах) нет. Операции отношения (=, <, >, <>, <=, >=) позволяют выполнить сравнение двух строковых операндов и используются при проверке условий. Сравнение строк осуществляется слева направо до первого несовпадающего символа. В этом случае большей считается та строка, в которой код первого несовпадающего символа больше. Если же строки совпадают по длине и содержат одни и те же символы, то они считаются равными. Примеры выражений с использованием сравнений строк:

- `"Олег " > "Олег"` — данное выражение истинно (результат — `True`), т. к. в левой части выражения присутствует дополнительный пробел;
- `"Петя"="Вася"` — данное выражение ложно (результат — `False`).

В листинге 5.2 приведена программа, демонстрирующая данные действия. Здесь введена логическая переменная `L`, в которую последовательно заносится результат сравнения строк. Так, сначала выполняется оператор

```
L = A > B,
```

который приводит к занесению в переменную `L` значения `True`. В конструкции

```
L = A = B
```

сначала выполняется проверка на равенство двух строк (`A` и `B`), а затем результат (`False`) заносится в переменную `L`.

Листинг 5.2. Пример действий над строками

```
Module Module1
    Sub Main()
        Dim L As Boolean
```

```
        Dim A, B As String
        A = "Петя "
        B = "Петя"
        L = A > B
        Console.WriteLine(L)
        A = "Петя"
        B = "Вася"
        L = A = B
        Console.WriteLine(L)
        Console.Read()
    End Sub
End Module
```

Рассмотрим пример, связанный с вводом с клавиатуры набора (массива) строк. В листинге 5.3 приведена разработка, в которой при повторном вводе уже встречавшейся строки работа программы завершается. Каждая новая уникальная строка записывается в очередной элемент массива `A(I)`. Здесь мы использовали цикл с предусловием `While` и переменную `Flag`. Изначально в эту переменную записывается 0, при обнаружении повторно введенной строки вносится 1. Цикл по вводу строк выполняется до тех пор, пока истинно условие:

```
While Flag = 0
```

В заключительной части программы выполняется вывод всех ранее введенных строк с помощью цикла:

```
For I = 1 To N
   Console.WriteLine(A(I))
Next
```

На рис. 5.1 показан результат вывода информации при вводе в программу трех строк.

Листинг 5.3. Введение строк с проверкой на идентичность

```
Module Module1
    Sub Main()
        Dim A(100) As String
        Dim B As String
        Dim N, I, Flag As Integer
        Flag = 0
        N = 0
        While Flag = 0
            Console.WriteLine("Введите строку")
            B = Console.ReadLine()
            For I = 1 To N
                If B = A(I) Then Flag = 1
            Next
```

```
            If Flag = 0 Then
                N = N + 1
                A(N) = B
            End If
        End While
        Console.WriteLine("Введенные строки:")
        For I = 1 To N
            Console.WriteLine(A(I))
        Next
        Console.Read()
    End Sub
End Module
```

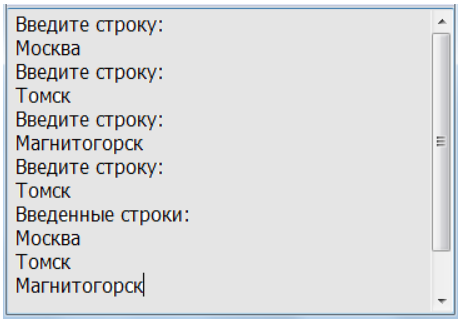

Рис. 5.1. Вывод информации в программе, приведенной в листинге 5.3

Работа со строками как с элементами массивов

Строка фактически представляет собой массив символов. Для доступа к конкретному символу необходимо указать имя, выбранное для строки, а также в круглых скобках указать номер символа. Если фактическая длина строки составляет `N` символов, то символы строки расположены в массиве с индексами от 0 до `N-1` включительно. В листинге 5.4 приведена разработка, которая выводит на экран индексы элементов массива и сами символы (элементы массива). Для вычисления длины строки мы воспользовались функцией `Len`.

На рис. 5.2 показан результат выполнения данной программы.

> ***Примечание***
>
> В предыдущем примере также использовались круглые скобки. Однако там мы работали с массивом строк, где каждый элемент массива представлял строку в целом. Здесь же мы осуществляем доступ к символам.

Листинг 5.4. Доступ к символам строки с помощью массива

```
Module Module1
    Sub Main()
        Dim A As String
```

```
        Dim I As Integer
        Dim C As Char
        A = "Бейсик"
        For I = 0 To Len(A) — 1
            C = A(I)
            Console.Write("Индекс = ")
            Console.Write(I)
            Console.Write(" Элемент = ")
            Console.WriteLine(C)
        Next
        Console.Read()
    End Sub
End Module
```

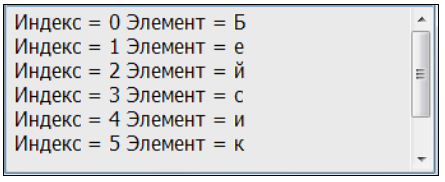

Рис. 5.2. Вывод информации в программе, приведенной в листинге 5.4

Рассмотрим еще один пример на данную тему. Так, необходимо по введенной с клавиатуры строке определить, является ли она записью целого числа в десятичной системе счисления. Программа (листинг 5.5) учитывает, что коды чисел расположены плотно и наша задача заключается в проверке: попадает ли каждый введенный символ в интервал от 0 до 9. Здесь мы использовали цикл с предусловием, действие которого состоит в проверке одновременного выполнения двух составляющих условий:

- не превысили ли мы число просматриваемых символов в строке;
- не оказался ли очередной символ символом, не являющимся цифрой.

Если очередной просматриваемый символ оказался не цифрой, то это фиксируется в переменной `Flag`. А именно: если очередной просматриваемый символ оказался не цифрой, то в качестве значения в переменную `Flag` заносится единица. В этом случае условие в операторе цикла ложно и выполнение цикла завершается.

В заключительной части программы осуществляется вывод соответствующего сообщения в зависимости от значения переменной `Flag`.

Листинг 5.5. Проверка строки, введенной с клавиатуры

```
Module Module1
    Sub Main()
        Dim A As String
        Dim N, Flag As Integer
```

```
        Console.WriteLine("Введите строку")
        A = Console.ReadLine()
        N = 0
        Flag = 0
        While N < Len(A) And Flag = 0
            If A(N) < "0" Or A(N) > "9" Then
                Flag = 1
            Else
                N = N + 1
            End If
        End While
        If Flag = 1 Then
            Console.WriteLine("Строка не является целым числом!")
         Else
            Console.WriteLine("Строка является целым числом!")
        End If
        Console.Read()
    End Sub
End Module
```

В тексте этого листинга использована стандартная функция работы со строками `Len`, которая вычисляет длину строки `A`, указанной в качестве параметра.

Учитывая распространенность проверки вводимой информации на ее отношение к числовому типу, логично присутствие стандартной функции для осуществления такой проверки. Это функция `IsNumeric`, которая выдает значение `True` в случае, если исходная информация является числом. В противном случае функция в качестве своего значения выдает значение `False`.

В листинге 5.6 приведена программная разработка, реализующая проверку информации на ее отношение к числовому типу.

Листинг 5.6. Проверка строки, введенной с клавиатуры (вариант 2)

```
Module Module1
    Sub Main()
        Dim A As String
        Console.WriteLine("Введите строку")
        A = Console.ReadLine()
        If IsNumeric(A) = True Then
            Console.WriteLine("Строка является целым числом!")
        Else
            Console.WriteLine("Строка не является целым числом!")
        End If
        Console.Read()
    End Sub
End Module
```

Копирование фрагмента строки

Рассмотрим ситуацию, когда из исходной строки необходимо выделить фрагмент и создать из него другую строку. Фактически задача заключается в том, чтобы из строки выделить подстроку. Так, в листинге 5.7 представлена необходимая разработка. На этот раз, вместо работы со строкой как с элементами массива мы воспользовались знакомой стандартной функцией `Mid (Str, L, N)`, где:

- `Str` — исходная строка;
- `L` — позиция выделяемой подстроки (заметим, что начальный символ расположен в единичной позиции);
- `N` — число выделяемых символов.

На рис. 5.3 приведено окно вывода информации в результате выполнения данной программы.

Листинг 5.7. Выделение фрагмента строки (вариант 1)

```
Module Module1
    Sub Main()
        Dim A As String
        Dim B As String
        Dim N, L As Integer
        Console.WriteLine("Введите строку")
        A = Console.ReadLine()
        Console.WriteLine("Введите количество выделяемых символов")
        N = Console.ReadLine()
        Console.WriteLine("Введите начало выделяемых символов")
        L = Console.ReadLine()
        B = Mid(A, L, N)
        Console.WriteLine(B)
        Console.Read()
    End Sub
End Module
```

Еще один вариант программной разработки, связанной с выделением фрагмента строки, представлен в листинге 5.8. Здесь мы воспользовались возможностью доступа к символам строки с помощью элементов массива.

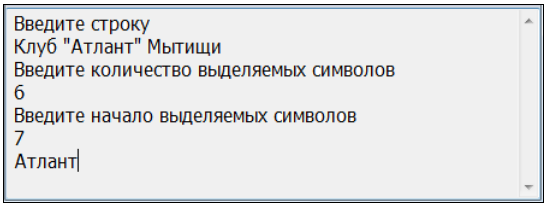

Рис. 5.3. Вывод информации в программе, приведенной в листинге 5.7

На рис. 5.4 приведено окно вывода информации в результате выполнения данной программы.

Листинг 5.8. Выделение фрагмента строки (вариант 2)

```
Module Module1
    Sub Main()
        Dim A As String
        Dim B As String
        Dim C As Char
        Dim N, L, I As Integer
        Console.WriteLine("Введите строку")
        A = Console.ReadLine()
        Console.WriteLine("Введите количество выделяемых символов")
        N = Console.ReadLine()
        Console.WriteLine("Введите начало фрагмента символов")
        L = Console.ReadLine()
        B = ""
        For I = L To L + N — 1
            C = A(I)
            B = B + C
        Next
        Console.WriteLine(B)
        Console.Read()
    End Sub
End Module
```

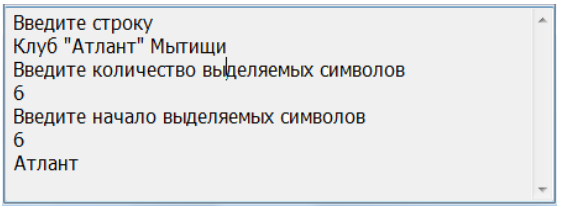

Рис. 5.4. Вывод информации в программе, приведенной в листинге 5.8

Подсчет количества слов в строке

Необходимо подсчитать количество слов в строке, введенной с клавиатуры. Подразумевается, что слова друг от друга могут отделяться одним или несколькими пробелами. При этом несколько пробелов могут располагаться в начале, в середине или в конце строки. Текст необходимой программы приведен в листинге 5.9.

В этой программе мы вводим переменную `s` для хранения строки, введенной с клавиатуры. Также для технических действий нам потребуются три целочисленные переменные:

- `I` — для индексации элементов массива;
- `L` — для хранения длины введенной строки;
- `M` — для размещения счетчика слов.

После ввода набора слов с помощью стандартной функции `Len` вычисляется длина введенной строки. Далее наша задача заключается в последовательном анализе соседних элементов массива `S(I)`, начиная с нулевого.

С помощью цикла `While` мы последовательно просматриваем символы с начального до предпоследнего. В данном цикле если текущий символ не является пробелом, а последующий, наоборот, является пробелом, то в счетчик слов (`M`) добавляется единица. После цикла важно проверить последний символ, и если он не является пробелом, то к счетчику слов добавляется еще одна единица.

Листинг 5.9. Подсчет числа слов в строке

```
Module Module1
    Sub Main()
        Dim S As String
        Dim I, L, M As Integer
        Console.WriteLine("Введите строку")
        S = Console.ReadLine()
        L = Len(S)
        M = 0
        I = 0
        While I < (L - 1)
            If S(I) <> " " And S(I + 1) = " " Then
                M = M + 1
            End If
            I = I + 1
        End While
        If S(L - 1) <> " " Then
            M = M + 1
        End If
       Console.Write("Количество слов ")
        Console.WriteLine(M)
        Console.Read()
    End Sub
End Module
```

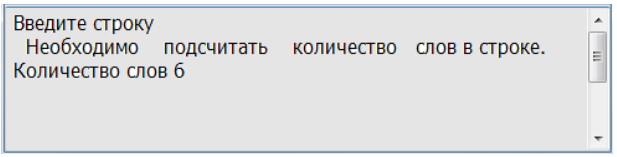

Рис. 5.5. Вывод информации в программе, приведенной в листинге 5.9

Листинг 5.10 содержит текст программы подсчета слов в строке с использованием функции `Mid`.

Листинг 5.10. Подсчет числа слов в строке (вариант 2)

```
Module Module1
    Sub Main()
        Dim S As String
        Dim I, L, M As Integer
        Console.WriteLine("Введите строку")
        S = Console.ReadLine()
        L = Len(S)
        M = 0
        I = 1
        While I < L
            If Mid(S, I, 1) <> " " And Mid(S, I + 1, 1) = " " Then
                M = M + 1
            End If
            I = I + 1
        End While
        If Mid(S, L, 1) <> " " Then
            M = M + 1
        End If
        Console.Write("Количество слов ")
        Console.WriteLine(M)
        Console.Read()
    End Sub
End Module
```

Подсчет количества символов фрагмента в строке

Необходимо подсчитать число символов во введенной строке, которые соответствуют одному из символов, присутствующих в предварительно сформированном наборе символов. Например, если сформированный набор представляет собой комбинацию: `abc` (три символа), то программа должна подсчитать общее число символов из данного списка во введенной строке. Текст необходимой программы приведен в листинге 5.11.

Листинг 5.11. Подсчет символов из фрагмента в строке

```
Module Module1
    Sub Main()
        Dim Nabor As String
        Dim S As String
        Dim I, J, L, M, N As Integer
```

```
        Console.WriteLine("Ввести набор интересующих символов в строке")
        Nabor = Console.ReadLine()
        N = Len(Nabor)
        Console.WriteLine("Ввести строку ")
        S = Console.ReadLine()
        L = Len(S)
        M = 0
        For I = 0 To L — 1
            For J = 0 To N — 1
                If S(I) = Nabor(J) Then
                    M = M + 1
                End If
            Next
        Next
        Console.WriteLine(M)
        Console.Read()
    End Sub
End Module
```

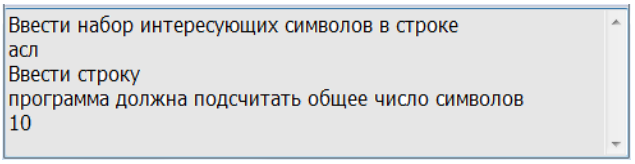

Рис. 5.6. Вывод информации в программе, приведенной в листинге 5.11

Нахождение суммы цифр

Требуется подсчитать сумму цифр в строке, а также вывести все имеющиеся цифры в строке на экран (листинг 5.12). Учитывая, что коды цифр расположены плотно в возрастающем порядке, для получения очередной цифры необходимо вычислить разность:

```
Asc(A(N)) — Asc("0").
```

Результат вывода информации программой, приведенной в листинге 5.12, показан на рис. 5.7.

Листинг 5.12. Подсчет суммы цифр в строке

```
Module Module1
    Sub Main()
    Dim A As String
        Dim N, Sum As Integer
        Console.WriteLine("Ввести строку")
        A = Console.ReadLine()
```

```
        N = 0
        Sum = 0
        Console.Write("Цифры: ")
        While N < Len(A)
            If (A(N) >= "0") And (A(N) <= "9") Then
                Sum = Sum + Asc(A(N)) — Asc("0")
                Console.Write(A(N))
            End If
            N = N + 1
        End While
        Console.WriteLine()
        Console.Write("Сумма цифр равна ")
        Console.WriteLine(Sum)
        Console.Read()
    End Sub
End Module
```

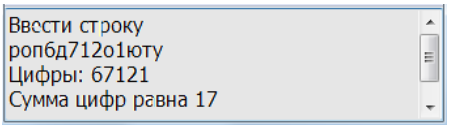

Рис. 5.7. Вывод информации в программе, приведенной в листинге 5.12

Примеры использования функций обработки строк

Кроме уже использованных в данном разделе функций имеется еще ряд интересных функций.

Так, для преобразования символов строки к определенному регистру используются:

- `LCase` — преобразует строку к нижнему регистру;
- `UCase` — преобразует строку к верхнему регистру.

В листинге 5.13 показан пример с использованием данных функций, а на рис. 5.8 показан результат диалога с программой.

Листинг 5.13. Преобразование строки к нижнему или верхнему регистру

```
Module Module1
    Sub Main()
        Dim A As String
        Console.WriteLine("Ввести строку")
        A = Console.ReadLine()
        Console.WriteLine("Преобразование строки к нижнему регистру: ")
        Console.WriteLine(LCase(A))
```

```
        Console.WriteLine("Преобразование строки к верхнему  регистру: ")
        Console.WriteLine(UCase(A))
        Console.Read()
    End Sub
End Module
```

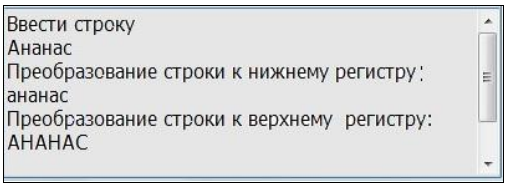

Рис. 5.8. Вывод информации в программе, приведенной в листинге 5.13

Для выделения фрагментов строк используются:

- `Left (A,N)` — возвращает подстроку, состоящую из заданного числа первых символов `N` исходной строки `A`;
- `Right (A,N)` — возвращает подстроку, состоящую из заданного числа последних символов `N` исходной строки `A`.

В листинге 5.14 показан пример с использованием данных функций, а на рис. 5.9 показан результат диалога с программой на экране.

Листинг 5.14. Выделение частей строки

```
Module Module1
    Sub Main()
        Dim A As String
        Console.WriteLine("Ввести строку")
        A = Console.ReadLine()
        Console.WriteLine("Левая часть строки: ")
        Console.WriteLine(Left(A, 5))
        Console.WriteLine("Правая часть строки:  ")
        Console.WriteLine(Right(A, 5))
        Console.Read()
    End Sub
End Module
```

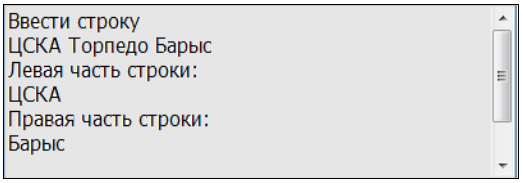

Рис. 5.9. Вывод информации в программе, приведенной в листинге 5.14

Для исключения пробелов фрагментов строк используются функции:

- `LTrim` — возвращает копию строки без пробелов в начале;
- `RTrim` — возвращает копию строки без пробелов в конце.

В листинге 5.15 показан пример с использованием данных функций, а на рис. 5.10 — результат диалога с программой на экране.

Листинг 5.15. Удаление левых и правых пробелов из строки

```
Module Module1
    Sub Main()
        Dim A As String
        Console.WriteLine("Ввести строку")
        A = Console.ReadLine()
        Console.Write("Число символов в исходной строке :")
        Console.WriteLine(Len(A))
        Console.Write("Число символов после отброса пробелов слева: ")
        Console.WriteLine(Len(LTrim(A)))
        Console.Write("Число символов после отброса пробелов справа:  ")
        Console.WriteLine(Len(RTrim(A)))
        Console.Read()
    End Sub
End Module
```

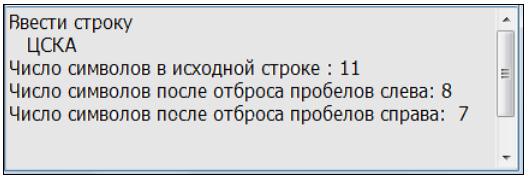

Рис. 5.10. Вывод информации в программе, приведенной в листинге 5.15

ГЛАВА 6

Процедуры и функции

Большие программы не принято разрабатывать в виде единого целого. Эффективнее большинство разработок разделять на отдельные функциональные компоненты. Для этого в языке Бейсик, как и в других языках программирования, имеется важный ресурс — процедуры. *Процедура* — это именованная, логически завершенная группа операторов, которую можно вызвать для выполнения любое количество раз из различных мест программы. Каждая процедура представляет собой самостоятельный фрагмент программного кода. Этот фрагмент, как правило, связан с основной программой с помощью параметров (входных и выходных).

Понятно, что когда программный код программы не очень большой (30—40 строк), то и необходимости выделять в нем отдельные самостоятельные фрагменты нет. Однако решение серьезных алгоритмических задач всегда требует существенно большего количества программных строк. И писать такие программы без разделения их на отдельные фрагменты совершенно не эффективно.

Разбиение программ на процедуры разумно по двум причинам. Во-первых, каждая процедура существует в памяти в единственном экземпляре, а вызывать ее можно многократно. Процедуры, в свою очередь, можно использовать при создании других процедур. Вторая причина касается методики нисходящего программирования. В этом случае алгоритм представляется в виде совокупности процедур, реализующих самостоятельные части алгоритма.

Процедуры подразделяются на две основные категории:

- процедуры `Sub`;
- процедуры `Function`, которые принято называть *функциями*, и далее мы так и будем поступать.

Отличие их друг от друга незначительное и заключается в том, что `Function` выдает значение, являющее результатом ее работы. Это значение можно использовать в выражениях. В качестве примера среди стандартных функций следует упомянуть тригонометрические функции (синус и косинус), с которыми мы уже встречались в этой книге.

Еще одна классификация — все процедуры и функции подразделяются на две группы: встроенные и определенные пользователем. *Встроенные* (стандартные)

процедуры и функции могут вызываться по имени без предварительного описания. Их наличие существенно облегчает разработку прикладных программ. Однако в большинстве случаев некоторые специфические действия требуют написания собственных *пользовательских* процедур и функций.

Вызов пользовательских функций и процедур реализуется простым упоминанием их имени, что похоже на вызов стандартных процедур и функций. Однако пользовательские процедуры и функции необходимо предварительно описать (как известно, любое имя в программе должно быть определено). Описать процедуру или функцию — это значит определить заголовок и тело *(см. далее)*.

Организация процедур

Процедура включает *заголовок* и *тело процедуры*. Заголовок состоит из зарезервированного слова `Sub`, *имени процедуры* и необязательного, заключенного в круглые скобки, списка формальных параметров с указанием типа каждого параметра:

```
Sub Имя_процедуры (Формальные параметры)
```

Также имени процедуры может предшествовать параметр *Уровень_доступности*, с помощью которого указывается, доступна ли процедура другим частям программы. Но в дальнейших примерах этим параметром мы пользоваться не будем.

Приведем пример двух заголовков процедур, соответственно, без параметров и с включением параметров:

- `Sub Linia();`
- `Sub Linia(ByVal Sim As Char, ByVal Cols As Integer).`

Передача параметров в процедуру может осуществляться двумя способами:

- по значению (`ByVal`);
- по ссылке (`ByRef`).

В случае передачи параметра по значению передается не сама переменная, а ее копия. Поэтому изменение параметра в процедуре затрагивает не переменную, а ее копию.

При передаче параметров по ссылке процедура получает доступ к области памяти, в которой эта переменная хранится, в результате чего при изменении в процедуре параметра происходит изменение значения переменной.

При выборе способа передачи параметра (по ссылке или по значению) решающим критерием является — должен ли изменяться параметр в процедуре. При передаче параметра по ссылке появляется возможность возвращения в вызывающую процедуру измененного значения параметра.

Имя процедуры должно быть уникально, т. е. его нельзя использовать повторно в программе для названия процедур.

> ***Примечание***
>
> Параметры процедуры должны отделяться друг от друга запятой.

Для обращения к процедуре используется *оператор вызова процедуры*. Он состоит из имени процедуры и списка *фактических* параметров, заключенных в круглые скобки и отделенных друг от друга запятыми. Если процедуре не передается никаких значений, то список параметров отсутствует.

Параметры обеспечивают механизм замены, который позволяет выполнять процедуру с различными исходными данными. Между фактическими параметрами в операторе вызова процедуры и формальными параметрами в заголовке описания процедуры устанавливается однозначное соответствие в результате их перебора слева направо. Количество и тип формальных параметров соответствуют количеству и типу фактических параметров. Заметим, что соответствующие друг другу параметры *не обязательно* должны одинаково обозначаться.

Использование имени процедуры в тексте вызывающей процедуры приводит к вызову данной процедуры (к выполнению программного кода, содержащегося в ней). После выполнения последнего оператора, имеющегося в процедуре, управление передается в основную программу (вызывающую процедуру). В этом случае будут выполняться операторы, следующие за строкой вызова процедуры. Таким образом, если отдельные фрагменты программного кода оформить как процедуры, то в основной программе вместо них останутся только операторы их вызова и основная программа станет существенно компактнее.

Любая процедура может, в свою очередь, иметь *вложенные процедуры*. При этом каждая процедура имеет совокупность имен. Считается, что все описанные имена видимы внутри процедуры и невидимы снаружи. При входе в процедуру становятся доступны не только имена, определенные внутри нее, но и все имена, указанные ранее (имена верхнего уровня). Эти объекты называются *глобальными*, в отличие от *локальных*, определенных внутри процедуры.

Примеры использования процедур

Теперь после рассмотрения необходимых теоретических сведений мы разберем типовые примеры использования процедур в практическом программировании.

Формирование разделяющей линии

Рассмотрим для начала наиболее простую организацию процедуры — без включения параметров. Пример будет касаться добавления оформления при выводе информации на экране. А именно, необходимо обеспечить формирование обрамляющей горизонтальной линии. В качестве линий будем использовать последовательность символов подчеркивания (_).

Формирование такой линии мы обеспечим с помощью процедуры `Linia`. В процедуре мы создали локальную переменную `J` и использовали цикл `For`. Полный текст процедуры и вызывающей (основной) процедуры представлен в листинге 6.1. В результате после выполнения программы на экране будет отображена горизонтальная линия из 50 символов.

Листинг 6.1. Использование процедуры для формирования линии

```
Module Module1
    Sub Linia()
        Dim J As Integer
        For J = 1 To 50
            Console.Write("_")
        Next
        Console.WriteLine()
    End Sub
    Sub Main()
        Linia()
        Console.Read()
    End Sub
End Module
```

Поясним теперь более детально содержание листинга 6.1. В начале программы после заголовка модуля располагается процедура, начинающаяся с ключевого слова `Sub`, за которым следует ее название — `Linia`. После названия процедуры до обозначения ее завершения `End Sub` размещается текст. Структура текста процедуры традиционна — область описания переменных и тело процедуры. Содержательной компонентой процедуры является выполнение цикла по выводу 50 символов подчеркивания и перевод на следующую строку.

В основной программе (процедуре `Main`) первым выполняемым оператором является вызов процедуры `Linia`, что автоматически приводит к выполнению программного кода, содержащегося в процедуре `Linia`. В результате на экране отображается горизонтальная линия. Рассмотренная ситуация достаточно простая, т. к. из основной программы в процедуру мы не передавали никаких параметров.

Передача символа рисования линии

Теперь усложним ситуацию предыдущего примера — обеспечим передачу в процедуру `Linia` кода символа для рисования линии. В этом случае линия необязательно будет оформлена с помощью символа нижнего подчеркивания. Можно выбрать звездочку или другой понравившийся символ. Текст процедуры вместе с основной программой представлен в листинге 6.2.

Листинг 6.2. Передача кода символа для рисования

```
Module Module1
    Sub Linia(ByVal Sim As Char)
        Dim J As Integer
        For J = 1 To 50
            Console.Write(Sim)
        Next
```

```
        Console.WriteLine()
    End Sub
    Sub Main()
        Linia("*")
        Linia("/")
        Console.Read()
    End Sub
End Module
```

В этом случае заголовок процедуры `Sub Linia(ByVal Sim As Char)` включает ключевое слово `ByVal`, которое говорит о том, что входной параметр в процедуру передается по значению.

Кроме этого в скобках указывается тип формального параметра (параметры). В примере, приведенном в листинге 6.2, мы использовали один параметр `Sim` типа `Char`.

В процедуре можно использовать формальные параметры так, как будто у нас есть переменные указанных типов.

При вызове процедуры `Linia` (см. листинг 6.2) необходимо в качестве параметра указать символ, которым собираемся отобразить линию. В нашем случае мы использовали два разных символа: `Linia("*")` и `Linia("/")`.

Передача двух параметров в процедуру

В предыдущем примере в качестве фактического параметра из основной программы в процедуру передавалось значение константы (тот или иной символ). Однако в качестве фактических параметров могут выступать и переменные. В листинге 6.3 приведен пример, в котором в подпрограмму передается два параметра:

- символ для рисования линии;
- число повторений этого символа при рисовании.

Значение второго параметра мы вводим с клавиатуры и записываем в переменную `N`. Далее значение этой переменной передается в параметр `Cols` процедуры `Linia`.

Результат вывода информации на экран показан на рис. 6.1.

Листинг 6.3. Передача значений переменных в процедуру

```
Module Module1
    Sub Linia(ByVal Sim As Char, ByVal Cols As Integer)
        Dim J As Integer
        For J = 1 To Cols
            Console.Write(Sim)
        Next
        Console.WriteLine()
    End Sub
```

```
    Sub Main()
        Dim A As Char
        Dim N As Integer
        Console.WriteLine("Введите количество выводимых символов")
        N = Console.ReadLine()
        A = "*"
        Linia(A, N)
        Console.Read()
    End Sub
End Module
```

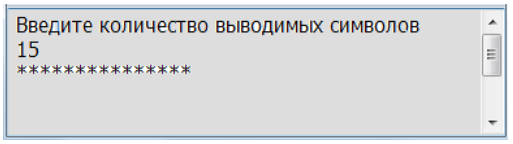

Рис. 6.1. Вывод информации в программе, приведенной в листинге 6.3

Процедура анализа четности числа

Рассмотрим уже знакомую задачу: необходимо проанализировать, является ли введенное с клавиатуры целое числом четным. Но на этот раз ввод интересующего нас числа реализуем в основной процедуре, а сам анализ четности проведем в процедуре, которую назовем `Chislo`. В листинге 6.4 представлена необходимая программная разработка.

Результат вывода информации на экран показан на рис. 6.2.

Листинг 6.4. Анализ четности целого числа

```
Module Module1
    Sub Chislo(ByVal Z As Integer)
        If Z Mod 2 = 0 Then
            Console.WriteLine("Число четное")
        Else
            Console.WriteLine("Число нечетное")
        End If
    End Sub
    Sub Main()
        Dim L As Integer
        Console.WriteLine("Введите число")
        L = Console.ReadLine()
        Chislo(L)
        Console.Read()
    End Sub
End Module
```

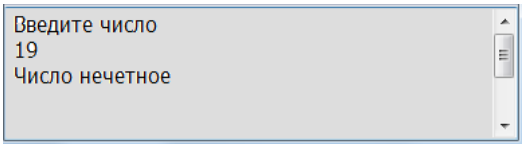

Рис. 6.2. Вывод информации в программе, приведенной в листинге 6.4

Передача параметров через глобальные переменные

В рассмотренных ранее примерах мы использовали передачу информации из основной программы в процедуру с помощью формальных и фактических параметров. Это наиболее стандартный вариант действий. Однако существует еще один ресурс, который можно использовать. Он связан с применением глобальных параметров.

Так, если описать переменные сразу после заголовка модуля, то все они будут доступны в операторах последующих процедур:

- основной процедуры `Main`;
- в процедуре `Linia`.

В листинге 6.5 приведен пример, в котором используются глобальные переменные для передачи символа (`Sim`) и числа повторений (`Cols`) этого символа при рисовании.

Результат вывода информации на экран показан на рис. 6.3.

Листинг 6.5. Пример использования глобальных переменных

```
Module Module1
    Dim Sim As Char
    Dim Cols As Integer
    Sub Linia()
        Dim J As Integer
        For J = 1 To Cols
            Console.Write(Sim)
        Next
        Console.WriteLine()
    End Sub
    Sub Main()
        Sim = "!"
        Cols = 15
        Linia()
        Console.Read()
    End Sub
End Module
```

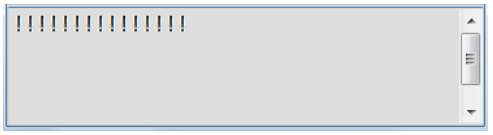

Рис. 6.3. Вывод информации в программе, приведенной в листинге 6.5

Глобальное описание массива

Рассмотрим пример, в котором будет продемонстрирована технология работы с массивами при использовании их глобального описания. В листинге 6.6 приведен пример, в котором с помощью процедур реализуется ввод значений массива, их вывод на экран и суммирование элементов.

Для этого мы описали массив `A`, вместе с переменной `Sum` и константой `N` сразу после заголовка модуля. После этого в модуле располагается четыре процедуры:

- `InputMass` — для ввода значений элементов массива;
- `OutMass` — для вывода значений элементов массива;
- `SumMass` — для суммирования значений элементов массива;
- `Main` — основная процедура.

Результат вывода информации на экран показан на рис. 6.4.

Листинг 6.6. Работа с массивами с использованием глобальных переменных

```
Module Module1
    Const N = 5
    Dim A(5) As Integer
    Dim Sum As Integer
    Sub InputMass()
        Dim I As Integer
        For I = 1 To N
            Console.WriteLine("Ввести элемент массива")
            A(I) = Console.ReadLine()
        Next
    End Sub
    Sub OutMass()
        Dim I As Integer
        For I = 1 To N
            Console.Write("    I=")
            Console.Write(I)
            Console.Write(" A=")
            Console.Write(A(I))
        Next
        Console.WriteLine()
    End Sub
```

```
    Sub SumMass()
        Dim I As Integer
        Sum = 0
        For I = 1 To N
            Sum = Sum + A(I)
        Next
        Console.Write("Сумма равна ")
        Console.WriteLine(Sum)
    End Sub
    Sub Main()
        InputMass()
        OutMass()
        SumMass()
        Console.Read()
    End Sub
End Module
```

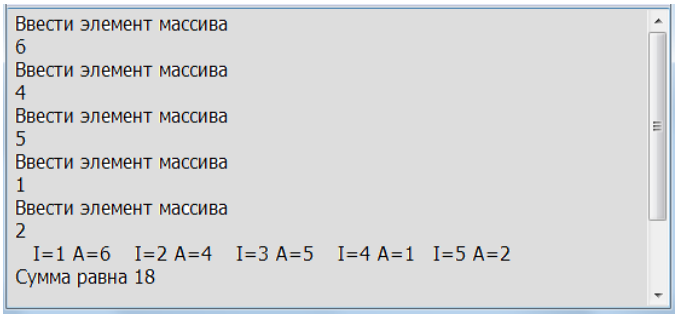

Рис. 6.4. Вывод информации в программе, приведенной в листинге 6.6

Рассмотрим еще один пример на тему работы с массивами. Необходимо с помощью процедур обеспечить:

- заполнение массива;
- поиск максимального значения в массиве и вывод его на экран.

В листинге 6.7 приведена полная разработка, включающая основную программу и две необходимые процедуры.

Результат вывода информации на экран показан на рис. 6.5.

Листинг 6.7. Поиск максимального значения в массиве с использованием процедур

```
Module Module1
    Const N = 5
    Dim A(5) As Integer
    Dim Maxsimum As Integer
```

```
    Sub FormMass()
        Dim I As Integer
        Randomize()
        Console.WriteLine("Массив :")
        For I = 1 To N
            A(I) = Int(100 * Rnd())
            Console.Write(A(I))
            Console.Write(" ")
        Next
        Console.WriteLine()
    End Sub
    Sub PoiskMax()
        Dim I As Integer
        Maxsimum = A(1)
        For I = 2 To N
            If A(I) > Maxsimum Then Maxsimum = A(I)
        Next
    End Sub
    Sub Main()
        FormMass()
        PoiskMax()
        Console.Write("Максимум ")
        Console.Write(Maxsimum)
        Console.Read()
    End Sub
End Module
```

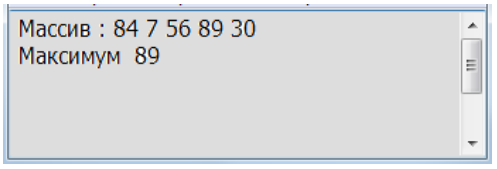

Рис. 6.5. Вывод информации в программе, приведенной в листинге 6.7

Передача параметров через ссылку

Особенность предыдущих примеров данной главы была связана с тем, что информация через параметры передавалась в одном направлении — из основной программы в процедуру. Если же требуется возвратить данные из процедуры в основную программу, то рассмотренный механизм не работает. Это связано с тем, что в данном случае осуществляется передача параметров по значению.

Существует и другой механизм, называемый *передачей параметров по ссылке* (по адресу). В этом случае в процедуру передается не значение переменной, а ее местоположение в памяти (адрес ячейки памяти, которая хранит данную переменную). После этого в процедуре можно легко изменить содержимое данного адреса.

В листинге 6.8 приведен вариант поиска максимального значения в массиве с использованием передачи параметров по ссылке.

При описании параметра, передаваемого по ссылке, используется ключевое слово `ByRef`. Сам массив `A`, а также переменные `N` и `Maxsimum` описываются в процедуре `Main`.

Результат вывода информации на экран показан на рис. 6.6.

Листинг 6.8. Пример передачи массива в процедуру по ссылке

```
Module Module1
    Sub FormMass(ByRef B() As Integer, ByVal L As Integer)
        Dim I As Integer
        Console.WriteLine("Массив :")
        For I = 1 To L
            B(I) = Int(100 * Rnd())
            Console.Write(B(I))
            Console.Write(" ")
        Next
    End Sub
    Sub PoiskMax(ByRef B() As Integer, ByVal L As Integer,
                 ByRef M As Integer)
        Dim I As Integer
        M = B(1)
        For I = 2 To L
            If B(I) > M Then M = B(I)
        Next
    End Sub
    Sub Main()
        Dim N = 5
        Dim A(5) As Integer
        Dim Maxsimum As Integer
        Randomize()
        FormMass(A, N)
        PoiskMax(A, N, Maxsimum)
        Console.WriteLine()
        Console.Write("Максимум ")
        Console.Write(Maxsimum)
        Console.Read()
    End Sub
End Module
```

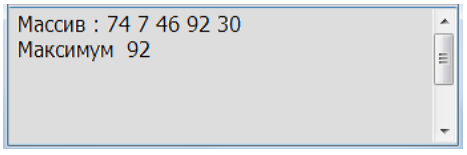

Рис. 6.6. Вывод информации в программе, приведенной в листинге 6.8

Вычисление факториала

В листинге 6.9 приведен пример вычисления факториала числа с использованием разработанной нами процедуры `Factorial`. После вычисления значение факториала передается в процедуру `Main` из процедуры `Factorial` через параметр `M`.

Результат вывода информации на экран показан на рис. 6.7.

Листинг 6.9. Вычисление факториала числа

```
Module Module1
    Sub Factorial(ByRef M As Long, ByVal N As Integer)
        Dim I As Integer
        M = 1
        For I = 1 To N
            M = M * I
        Next
    End Sub
    Sub Main()
        Dim A As Long
        Dim B As Integer
        Console.WriteLine("Введите число")
        B = Console.ReadLine()
        Factorial(A, B)
        Console.Write("Число=")
        Console.Write(B)
        Console.Write("    Факториал=")
        Console.Write(A)
        Console.Read()
    End Sub
End Module
```

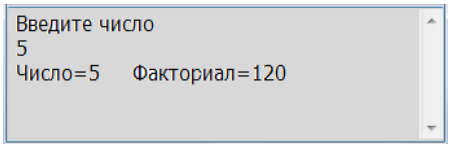

Рис. 6.7. Вывод информации в программе, приведенной в листинге 6.9

Вычисление математических функций

Рассмотрим пример, где требуется с помощью процедуры вычислить значения нескольких математических функций, а результат вычисления передать в основную процедуру. В листинге 6.10 приведен такой пример для вычисления значений следующих математических функций:

- $y = \sin(x) + 4$;
- $z = \cos(3x) - 1$;
- $w = 2x - 7$.

Вычисленные значения трех функций передаются в процедуру `Main` по ссылке.

Результат вывода информации на экран показан на рис. 6.8.

Листинг 6.10. Вычисление значений математических функций

```
Module Module1
    Sub Func(ByVal X As Single, ByRef Y As Single,
             ByRef Z As Single, ByRef W As Single)
        Y = Math.Sin(X) + 4
        Z = Math.Cos(3 * X)
        W = 2 * X — 7
    End Sub
    Sub Main()
        Dim Y, Z, W, X As Single
        Console.Write("Введите значение X   ")
        X = Console.ReadLine()
        Func(X, Y, Z, W)
        Console.Write("Y=")
        Console.WriteLine(Y)
        Console.Write("Z=")
        Console.WriteLine(Z)
        Console.Write("W=")
        Console.Write(W)
        Console.Read()
    End Sub
End Module
```

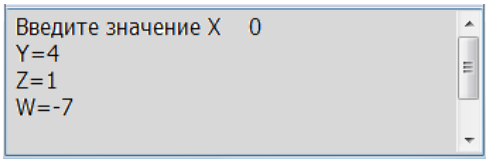

Рис. 6.8. Вывод информации в программе, приведенной в листинге 6.10

Обмен значений переменных

Необходимо разработать процедуру, которая будет осуществлять обмен значений двух переменных. Далее с помощью этой процедуры требуется разместить в порядке возрастания три числа, которые вводятся с клавиатуры. Разработка приведена в листинге 6.11.

Результат вывода информации на экран показан на рис. 6.9.

Листинг 6.11. Перестановка чисел

```
Module Module1
    Sub Obmen(ByRef X As Integer, ByRef Y As Integer)
        Dim Z As Integer
        Z = X
        X = Y
        Y = Z
    End Sub
    Sub Main()
        Dim A, B, C As Integer
        Console.Write("Введите значение A   ")
        A = Console.ReadLine()
        Console.Write("Введите значение B  ")
        B = Console.ReadLine()
        Console.Write("Введите значение C   ")
        C = Console.ReadLine()
        If A > B Then Obmen(A, B)
        If B > C Then Obmen(B, C)
        If A > B Then Obmen(A, B)
        Console.Write("A=")
        Console.Write(A)
        Console.Write("   B=")
        Console.Write(B)
        Console.Write("   C=")
        Console.Write(C)
        Console.Read()
    End Sub
End Module
```

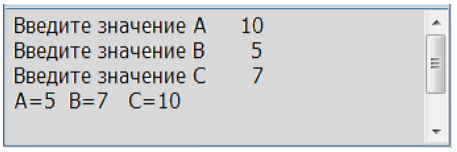

Рис. 6.9. Вывод информации в программе, приведенной в листинге 6.11

Анализ чисел

С помощью процедуры реализуем программу, которая будет анализировать: является ли число простым или нет. Процедура будет иметь два параметра:

- число, которое следует проанализировать;
- переменную, представляющую собой результат, получаемый из процедуры основной программой.

В листинге 6.12 показана разработка, включающая основную процедуру и процедуру, выполняющую проверку чисел.

Результат вывода информации на экран показан на рис. 6.10.

Листинг 6.12. Анализ простых чисел

```
Module Module1
    Sub Analiz(ByVal N As Integer, ByRef Res As Boolean)
        Dim J As Integer
        Res = True
        For J = 2 To N — 1
            If N Mod J = 0 Then
                Res = False
                Exit For
            End If
        Next
    End Sub
    Sub Main()
        Dim X As Integer
        Dim F As Boolean
        Console.Write("Введите целое положительное число   ")
        X = Console.ReadLine()
        Analiz(X, F)
        Console.Write("Число простое — ")
        Console.Write(F)
        Console.Read()
    End Sub
End Module
```

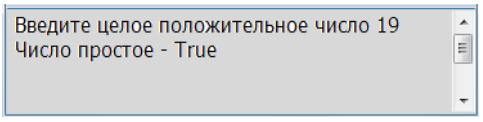

Рис. 6.10. Вывод информации в программе, приведенной в листинге 6.12

Функции пользователя

В большинстве случаев результатом процедуры является только одно значение. И в этом случае вычислительные действия лучше оформить не как процедуру, а как функцию. Организация функции аналогична процедуре, но имеются два отличия:

- функция передает в программу результат своей работы — единственное значение, носителем которого является имя самой функции;
- имя функции может входить в выражение как операнд (функция возвращает результат в точку своего вызова).

Функция, определенная пользователем, состоит из заголовка и тела функции. Заголовок содержит зарезервированное слово `Function` и имя функции. Кроме того, после имени функции в круглых скобках указывается список формальных параметров. А главным отличием от синтаксиса процедуры является указание типа возвращаемого значения.

Тело функции по своей структуре аналогично обычной процедуре.

Заметим, что в разделе операторов функции должен находиться, по крайней мере, один оператор, который присваивает ее имени значение, возвращаемое как результат работы функции.

Далее рассмотрим ряд практических примеров программирования с использованием функций.

Вычисление наибольшего значения

Необходимо разработать функцию, которая вычисляет максимальное из двух вещественных чисел, полученных в качестве аргумента. Требуется также привести вызывающую процедуру. В листинге 6.13 данная разработка представлена целиком.

Листинг 6.13. Пример функции вычисления наибольшего значения

```
Module Module1
    Function Comp(ByVal A As Integer, ByVal B As Integer)
        if A > B then
            Comp = A
        Else
            Comp = B
        End If
    End Function
    Sub Main()
        Dim X1, X2 As Integer
        Console.Write("Введите целое число X1  ")
        X1 = Console.ReadLine()
        Console.Write("Введите целое число X2  ")
        X2 = Console.ReadLine()
        Console.Write("Наибольшее число = ")
        Console.Write(Comp(X1, X2))
        Console.Read()
    End Sub
End Module
```

Теперь небольшое усложнение. Необходимо разработать другую программу с использованием сформированной функции `Comp`. В данном случае мы должны будем выбирать максимальное из трех чисел вводимых с клавиатуры. Данная разработка представлена в листинге 6.14.

Листинг 6.14. Выбор наибольшего значения среди трех переменных

```
Module Module1
    Function Comp(ByVal A As Integer, ByVal B As Integer)
        If A > B Then
            Comp = A
        Else
            Comp = B
        End If
    End Function
    Sub Main()
        Dim X1, X2, X3 As Integer
        Console.Write("Введите целое число X1  ")
        X1 = Console.ReadLine()
        Console.Write("Введите целое число X2  ")
        X2 = Console.ReadLine()
        Console.Write("Введите целое число X3  ")
        X3 = Console.ReadLine()
        Console.Write("Наибольшее число = ")
        Console.Write(Comp(Comp(X1, X2), X3))
        Console.Read()
    End Sub
End Module
```

Вычисление процента

Необходимо разработать функцию, которая вычисляет процент от числа. Эти данные (число и значение в процентах, которое требуется вычислить) передаются в качестве параметров. В листинге 6.15 данная разработка представлена вместе с вызывающей процедурой.

Листинг 6.15. Функция вычисления значения процента от числа

```
Module Module1
    Function Protsent(ByVal Chislo As Single,
                      ByVal Prots As Single) As Single
        Protsent = Chislo * Prots / 100
    End Function
    Sub Main()
        Dim A, B As Integer
        Console.Write("Введите  число  ")
        A = Console.ReadLine()
        Console.Write("Введите процент, который необходимо вычислить  ")
        B = Console.ReadLine()
        Console.Write("Процент = ")
        Console.Write(Protsent(A, B))
```

```
        Console.Read()
    End Sub
End Module
```

Расчет дохода по вкладу

Нужно разработать функцию, которая вычисляет доход по вкладу. Данные (сумма вклада, годовая ставка по вкладу и срок в днях) передаются в качестве параметров. В листинге 6.16 данная разработка представлена вместе с вызывающей программой.

Листинг 6.16. Функция вычисления дохода по вкладу

```
Module Module1
    Function Protsent(ByVal Summa As Single,
             ByVal Prots As Single, ByVal Srok As Integer) As Single
        Protsent = Summa * (Prots / 100) * (Srok / 365)
    End Function
    Sub Main()
        Dim A, B, C As Integer
        Console.Write("Введите  сумму  ")
        A = Console.ReadLine()
        Console.Write("Введите процент, который необходимо вычислить  ")
        B = Console.ReadLine()
        Console.Write("Введите на какой срок вклад (в днях)  ")
        C = Console.ReadLine()
        Console.Write("Доход по вкладу = ")
        Console.Write(Format(Protsent(A, B, C), "0.0006"))
        Console.Read()
    End Sub
End Module
```

Анализ текста

Требуется разработать функцию, которая анализирует строку (вводимую с клавиатуры), и если в ней присутствует английская заглавная буква, то выдаваемое функцией значение должно иметь значение `True`. В противном случае функция выдает значение `False`. Данная разработка представлена в листинге 6.17.

Листинг 6.17. Анализ строки текста

```
Module Module1
    Function Books(ByVal S As String) As Boolean
        Dim L, I As Integer
        L = Len(S)
```

```
        Books = False
        For I = 1 To L
            If Mid(S, I, 1) >= "A" And Mid(S, I, 1) <= "Z" Then
                Books = True
                Exit For
            End If
     Next
    End Function
    Sub Main()
        Dim A As String
        Console.Write("Введите строку  ")
        A = Console.ReadLine()
        Console.Write("В строке есть заглавные английские буквы — ")
        Console.Write(Books(A))
        Console.Read()
    End Sub
End Module
```

Функция поиска минимума в одномерном массиве

В *главе 3* мы рассматривали задачу поиска максимального значения в одномерном массиве. В листинге 6.18 приведено решение аналогичной задачи — поиска минимального значения, но при этом основной алгоритм оформлен в виде функции.

Листинг 6.18. Функция поиска минимального значения в массиве

```
Module Module1
    Const N = 10
    Function Poisk(ByRef M() As Integer) As Integer
        Dim J, Min As Integer
        Min = M(1)
        For J = 2 To N
            If M(J) < Min Then
                Min = M(J)
            End If
        Next
        Poisk = Min
    End Function
    Sub Main()
        Dim A(10) As Integer
        Dim J As Integer
        Randomize()
        For J = 1 To N
            A(J) = Int(100 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
```

```
        Console.WriteLine()
        Console.Write("Минимум= ")
        Console.Write(Poisk(A))
        Console.Read()
        End Sub
End Module
```

Функция подсчета соседних элементов массива

Необходимо разработать функцию подсчета максимального количества идущих подряд возрастающих элементов в одномерном массиве. В *главе 3* мы решали подобную задачу, где было необходимо найти максимальное количество идущих подряд одинаковых элементов. Здесь требуется разбить алгоритм на две части и разработать вместе с функцией еще и основную программу. Программа, решающая данную задачу, представлена в листинге 6.19.

Листинг 6.19. Функция анализа элементов массива

```
Module Module1
    Const N = 10
    Function Poisk(ByRef M() As Integer) As Integer
        Dim J, Nums, MaxNums As Integer
        Nums = 1
        MaxNums = 1
        For J = 2 To N
            If M(J) > M(J — 1) Then
                Nums = Nums + 1
            Else
                If Nums > MaxNums Then
                    MaxNums = Nums
                    Nums = 1
                End If
            End If
        Next
        If Nums > MaxNums Then
            MaxNums = Nums
        End If
        Poisk = MaxNums
    End Function
    Sub Main()
        Dim J As Integer
        Dim A(10) As Integer
        For J = 1 To N
            A(J) = Int(100 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
```

```
        Console.WriteLine()
        Console.Write("Число возрастающих элементов= ")
        Console.Write(Poisk(A))
        Console.Read()
    End Sub
End Module
```

Функция изменения значений элементов массива

Необходимо разработать функцию, осуществляющую изменение определенных значений в одномерном массиве. Так, если в массиве есть элементы, равные нулю, то их следует заменить на такое значение элемента массива, которое наиболее близко к нулю. В качестве своего значения функция должна выдать значение `True`, если такая замена произошла, и `False` — в противном случае. В листинге 6.20 приведена соответствующая программная разработка.

Листинг 6.20. Изменение значений элементов в массиве

```
Module Module1
    Const N = 10
    Function Zamena(ByRef A() As SByte) As Boolean
        Dim J As Integer
        Dim Delta As SByte
        Zamena = False
        Delta = 127
        For J = 1 To N
            If Math.Abs(A(J)) < Math.Abs(Delta) And A(J) <> 0 Then
                Delta = A(J)
            End If
        Next
        For J = 1 To N
            If A(J) = 0 Then
                A(J) = Delta
                Zamena = True
            End If
        Next
     End Function
    Sub Main()
        Dim J As Integer
        Dim A(10) As SByte
        For J = 1 To N
            A(J) = -2 + Int(5 * Rnd())
            Console.Write(A(J))
            Console.Write(" ")
        Next
        Console.WriteLine()
        Console.Write("Замена элементов ")
        Console.WriteLine(Zamena(A))
```

```
        For J = 1 To N
            Console.Write(A(J))
            Console.Write(" ")
        Next
        Console.Read()
    End Sub
End Module
```

Функция вычисления суммы элементов двумерного массива

Рассмотрим теперь пример работы с двумерным массивом. Необходимо разработать функцию, которая будет суммировать элементы, значения которых являются четными числами. В листинге 6.21 приведена программа, реализующая решение данной задачи.

Листинг 6.21. Функция вычисления суммы значений элементов массива

```
Module Module1
    Const N = 10
    Function Raschet(ByRef A(,) As Integer) As Integer
        Dim I, J, Sum As Integer
        Sum = 0
        For I = 1 To N
            For J = 1 To N
                If A(I, J) Mod 2 = 0 Then
                    Sum = Sum + A(I, J)
                End If
            Next
        Next
        Raschet = Sum
    End Function
    Sub Main()
        Dim I, J As Integer
        Dim A(10, 10) As Integer
        For I = 1 To N
            For J = 1 To N
                A(I, J) = Int(5 * Rnd())
            Next
        Next
        Console.WriteLine()
        Console.Write("Сумма элементов ")
        Console.WriteLine(Raschet(A))
        Console.Read()
    End Sub
End Module
```

ГЛАВА 7

Математические вычисления

В этой главе внимание будет сконцентрировано на задачах, касающихся разнообразных математических вычислений. В билетах Единого государственного экзамена присутствуют такие задания, которые в определенной степени связаны с разделами математики. Конечно, глубоких знаний в математической области они не требуют, однако вопросы, связанные со стандартными функциями и решением несложных уравнений, необходимо знать.

Расчет значений функции

Требуется программно вычислить значения функции в указанном диапазоне изменения аргумента. А именно необходимо построить работающую программу, которая вычисляет значения функции $y = \sin(x+5) - x$. Диапазон изменения аргумента составляет от 1 до 5, а шаг приращения аргумента равен 0,3.

Для начала необходимо подсчитать число точек (значений аргумента), в которых следует вычислять значения функции:

$$N = \mathrm{Int}\left(\frac{X_{\max} - X_{\min}}{Dx} + 1\right), \tag{7.1}$$

где:

- $X_{\max}$ — максимальное значение аргумента;
- $X_{\min}$ — минимальное значение аргумента;
- Dx — шаг приращения аргумента;
- Int — целая часть числа.

В листинге 7.1 приведена необходимая программа для вычисления значений функции. Здесь в цикле от 1 до N последовательно вычисляются значения функции по формуле $y = \sin(x+5) - x$.

Листинг 7.1. Программа вычисления значений функции

```
Module Module1
    Sub Main()
        Const Xmax = 5
        Const Xmin = 1
        Const Dx = 0.3
        Dim X, Y As Single
        Dim J, N As Integer
        N = Int((Xmax — Xmin) / Dx + 1)
        X = Xmin
        For J = 1 To N
            Y = Math.Sin(X + 5) — X
            Console.Write(" X=")
            Console.Write(Format(X, "0.0"))
            Console.Write(" Y=")
            Console.WriteLine(Format(Y, "0.000"))
            X = X + Dx
        Next
        Console.Read()
    End Sub
End Module
```

Численное интегрирование

Некоторые школьники знакомы с интегралами и методикой их вычисления. Однако на практике часто встречаются определенные интегралы (напомним, что в отличие от неопределенных у них указываются пределы интегрирования), которые либо нельзя вычислить аналитическими методами, либо вычисление требует очень больших усилий. В этом случае можно использовать численные методы, что обеспечивает вместо точного значения приближенное. Задача, которую мы здесь рассмотрим, заключается в вычислении интеграла:

$$Z = \int_{A}^{B} F(x)dx, \tag{7.2}$$

где Z — искомое значение, которое фактически является площадью фигуры, ограниченной функцией $F(x)$, осью абсцисс и прямыми $x = A$ и $x = B$.

На рис. 7.1 эта задача проиллюстрирована графически.

Существуют методы приближенного вычисления определенного интеграла, которые основываются на замене площади сложной фигуры (рис. 7.1) на конечный набор элементарных участков. Тогда площадь исходной фигуры легко вычисляется как сумма площадей элементарных участков.

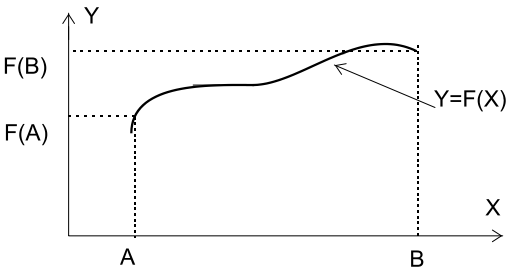

Рис. 7.1. График функции формулы (7.2)

Наиболее простой вариант связан с численным интегрированием по методу прямоугольников. В этом случае интервал от A до B делится точками $x_1, x_2, ..., x_N$ на N равных частей (рис. 7.2). При этом выполняются следующие условия:

- $x_1 = A$;
- $x_N = B$;
- $h = (B - A) / N$.

В этом случае очередную точку по оси абсцисс можно определить так:

$$x_i = x_1 + (i - 1) \cdot h .$$

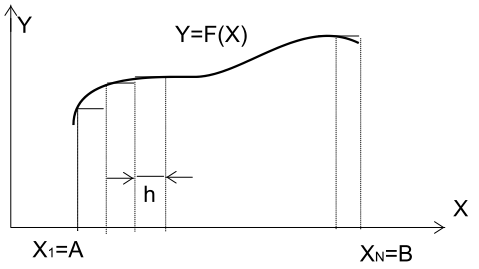

Рис. 7.2. График функции формулы (7.2) с разбивкой на прямоугольники

В соответствии с рис. 7.2 из каждой точки на оси абсцисс проводится перпендикуляр до пересечения с кривой $F(x)$. После этого кривая подынтегральной функции заменяется на ломаную линию, отрезки которой параллельны оси абсцисс. Площадь такой ступенчатой фигуры можно найти как сумму площадей прямоугольников, стороны которых равны h и y_i. Таким образом, площадь отдельного прямоугольника вычисляется:

$$S_i = y_i \cdot h .$$

В этом случае интеграл (7.2) можно приближенно заменить следующим рядом:

$$Z = h \sum_{i=1}^{N-1} F(x_i) . \qquad (7.3)$$

Запрограммировать полученное соотношение достаточно просто, и в качестве примера приведем вычисление интеграла для следующих исходных данных:

- $F(x)=x^3+\cos(x)+1;$
- $A=1;$
- $B=3.$

В листинге 7.2 приведена необходимая программа для вычисления значений функции. Здесь в цикле от 1 до N последовательно вычисляются значения функции и суммируются площади прямоугольников. В качестве N в приведенной программе мы выбрали значение 100.

Листинг 7.2. Программа вычисления значений функции

```
Module Module1
    Function Integral(ByVal A As Single, ByVal B As Single,
                      ByVal N As Integer) As Single
        Dim S, H, Xi, Fi As Single
        Dim I As Integer
        H = (B — A) / N
        S = 0
        For I = 1 To N — 1
            Xi = A + H * (I — 1)
            Fi = Xi ^3 + Math.Cos(Xi) + 1
            S = S + Fi * H
        Next
        Integral = S
    End Function
Sub Main()
        Console.Write(" Интеграл равен ")
        Console.Write(Integral(1, 3, 100))
        Console.Read()
    End Sub
End Module
```

Однако использование замены функции совокупностью прямоугольников дает невысокую точность. Для ее повышения требуется существенно увеличивать число точек N.

Гораздо большую точность обеспечивает метод трапеций. Если провести ординаты во всех точках деления линии и заменить каждую из полученных "криволинейных" трапеций прямолинейной (рис. 7.3), то приближенное значение интеграла будет равно сумме площадей прямолинейных трапеций. Площадь отдельной трапеции вычисляется:

$$Z_i=\frac{y_i+y_{i+1}}{2}h,$$

где i изменяется от 1 до $N-1.$

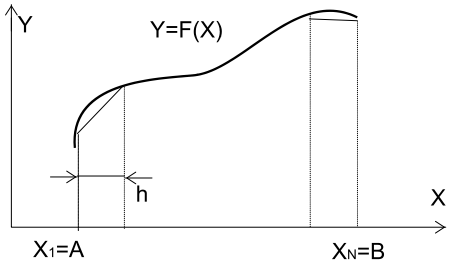

Рис. 7.3. График функции формулы (7.2) с разбивкой на трапеции

Тогда площадь искомой фигуры будет равна сумме всех трапеций:

$$Z = \int_A^B F(x)dx = \frac{h}{2}\sum_{i=1}^{N-1}(y_i + y_{i+1}) = h \cdot \left(\frac{y_1 + y_N}{2} + \sum_{i=2}^{N-1} y_i \right).$$

В результате можно записать формулу трапеций для численного интегрирования:

$$S = h \cdot \left(\frac{F(A) + F(B)}{2} + \sum_{i=2}^{N-1} F(x_i) \right). \qquad (7.4)$$

Теперь применим соотношение (7.4) при вычислении интеграла для следующих исходных данных:

- $F(x) = x^2 \cdot \cos(x)$;
- $A = -1$;
- $B = 1$.

В листинге 7.3 приведена необходимая программа для вычисления значений данной функции. Здесь в цикле от 1 до N последовательно вычисляются значения функции. Для N в приведенной программе мы выбрали значение 100.

Листинг 7.3. Пример вычисления интеграла по методу трапеций

```
Module Module1
 Function Integral(ByVal A As Single, ByVal B As Single,
                   ByVal N As Integer) As Single
        Dim S, H, Xi, Fi, FA, FB As Single
        Dim I As Integer
        H = (B — A) / N
        S = 0
        For I = 2 To N — 1
            Xi = A + H * (I — 1)
            Fi = Xi * Xi + Math.Cos(Xi)
            S = S + Fi
        Next
```

```
        FA = A * A * Math.Cos(A)
        FB = B * B * Math.Cos(B)
        Integral = H * ((FA + FB) / 2 + S)
    End Function
    Sub Main()
        Console.Write(" Интеграл равен ")
        Console.Write(Integral(1, 3, 100))
        Console.Read()
    End Sub
End Module
```

Решение уравнений

В билетах Единого государственного экзамена имеется много заданий, касающихся решений уравнений. В этом разделе мы рассмотрим ряд задач в том виде, как они были представлены в экзаменационных билетах.

ПРИМЕР 1

Требуется написать программу, которая решает уравнение `AX+B=0` относительно `X` при любых числах `A` и `B`, введенных с клавиатуры. Все числа считаются действительными.

Программист сделал в программе (листинг 7.4) ошибки.

Вам предлагается последовательно выполнить следующие задания:

- указать, где имеется ошибка, связанная с вводом информации;
- найти другие ошибки в листинге 7.4 и обосновать их;
- доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.4. Программа решения уравнения `AX+B=0` с наличием ошибок

```
Module Module1
    Sub Main()
        Dim A, B, X As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
        B = Console.ReadLine()
        Console.Write("Введите X ")
        X = Console.ReadLine()
        If B = 0 Then
            Console.Write("X=0 ")
        Else
            If A = 0 Then
                Console.Write("решений нет")
```

```
            Else
                Console.Write(-B / A)
            End If
        End If
        Console.Read()
    End Sub
End Module
```

Выполнение заданий

1. Учитывая то обстоятельство, что мы решаем уравнение относительно X, вводить значение X с клавиатуры не требуется. Поэтому нужно исключить следующий фрагмент строк:

```
Console.Write("Введите X ")
X = Console.ReadLine()
```

2. Теперь перейдем непосредственно к решению уравнения. Рассмотрим следующие варианты:
   - если A ≠ 0, то в этом случае X = –B/A;
   - если A = 0 и B = 0, то в качестве решения подойдет любое значение X;
   - если только A = 0, то решений уравнения нет. Это и нужно отразить в программном решении уравнения.
3. Из рассмотрения листинга 7.4 вытекает, что при равных нулю коэффициентах A и B в качестве решения уравнения получается ноль, однако на самом деле в этом случае решением является любое значение X. С учетом сказанного, возможная коррекция программы представлена в листинге 7.5.

Листинг 7.5. Исправленный вариант листинга 7.4

```
Module Module1
    Sub Main()
        Dim A, B, X As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
        B = Console.ReadLine()
        If A = 0 And B = 0 Then
            Console.Write("Любое значение X ")
        Else
            If A = 0 Then
                Console.Write("решений нет")
            Else
                Console.Write(-B / A)
            End If
        End If
```

```
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 2

Рассмотрим еще один пример, связанный с решением уравнения. Требуется написать программу, которая решает уравнение `|X|-2=-A` относительно `X` для любого числа `A`, вводимого с клавиатуры. Все числа считаются действительными.

Программист сделал в программе (листинг 7.6) ошибки.

Вам предлагается последовательно выполнить следующие задания:

- указать, где имеется ошибка, связанная с вводом информации;
- найти другие ошибки в листинге 7.6 и обосновать их;
- доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.6. Программа решения уравнения `|X|-2=-A` с наличием ошибок

```
Module Module1
    Sub Main()
        Dim A, X As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите X ")
        X = Console.ReadLine()
        If (2 — A) < 0 Then
            Console.Write(2 — A)
        Else
            Console.Write("решений нет")
        End If
        Console.Read()
    End Sub
End Module
```

Выполнение заданий

1. Здесь, как и в примере 1, мы решаем уравнение относительно `X`, значит, вводить значение `X` с клавиатуры не требуется. Поэтому фрагмент

   ```
   Console.Write("Введите X ")
   X = Console.ReadLine()
   ```

   нужно убрать.

2. А теперь перед коррекцией программы рассмотрим решение уравнения. Модуль числа (в данном случае `|X|`) не может быть отрицательным, поэтому `2-A` должно быть больше или равно 0. Таким образом, после ввода с клавиатуры числа `A` необходимо проверить выражение `2-A`, и если оно отрицательное, то следует

вывести сообщение о том, что решений уравнения нет. В противном случае имеются два решения:

- `2-A`;
- `A-2`.

С учетом сказанного возможная коррекция программы представлена в листинге 7.7.

Листинг 7.7. Исправленный вариант листинга 7.6

```
Module Module1
    Sub Main()
        Dim A, X As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        If (2 — A) < 0 Then
            Console.Write("решений нет")
        Else
            Console.WriteLine(2 — A)
            Console.Write(A — 2)
        End If
        Console.Read()
    End Sub
End Module
```

ПРИМЕР 3

Рассмотрим еще один пример на данную тему. Требуется решить уравнение: `A|X| = B` относительно `X` для любых чисел `A` и `B`, введенных с клавиатуры. Все рассматриваемые числа считаются действительными.

Программист поторопился и написал программу неправильно (листинг 7.8).

Вам предлагается последовательно выполнить следующие задания:

- ❒ указать, где имеется ошибка, связанная с вводом информации;
- ❒ найти другие ошибки в листинге 7.8 и обосновать их;
- ❒ доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.8. Программа решения уравнения `A|X| = B` с наличием ошибок

```
Module Module1
    Sub Main()
        Dim A, B, X As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
```

```
        B = Console.ReadLine()
        Console.Write("Введите X ")
        X = Console.ReadLine()
        If A = 0 And B = 0 Then
            Console.Write("Любое число")
        Else
            Console.WriteLine(B / A)
            Console.Write(-B / A)
        End If
        Console.Read()
    End Sub

End Module
```

Выполнение заданий

1. Учитывая, что мы решаем уравнение относительно `X`, то вводить его значение с клавиатуры не требуется. Поэтому фрагмент

   ```
   Console.Write("Введите X ")
   X = Console.ReadLine()
   ```

   нужно убрать.

2. Перед коррекцией программы рассмотрим сначала решение уравнения:
   - если `A = 0`, то при `B = 0` решением является любое `X`;
   - если `A = 0` и `B ≠ 0`, то уравнение решений не имеет.
3. Рассмотрим теперь ситуацию, когда `A ≠ 0`. Мы должны учесть, что `|X|` не может быть отрицательным. В связи с этим `B/A` должно принимать значения не меньше нуля. Таким образом, после ввода с клавиатуры параметров `A` и `B` необходимо проанализировать выражение `B/A`, и если оно отрицательное, то уравнение решений не имеет. Если `B = 0`, то `X = 0`. В противном случае решений два:
   - `B/A`;
   - `-B/A`.

С учетом сказанного возможная коррекция программы представлена в листинге 7.9.

Листинг 7.9. Исправленный вариант листинга 7.8

```
Module Module1
    Sub Main()
        Dim A, B, X As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
        B = Console.ReadLine()
        If A = 0 And B = 0 Then
            Console.Write("Любое число")
```

```
        Else
            If A = 0 Then
                Console.WriteLine("Решений нет")
            Else
                If (B / A) < 0 Then
                    Console.WriteLine("Решений нет")
                Else
                    If B = 0 Then
                        Console.WriteLine("Решение X=0")
                    Else
                        Console.WriteLine("2 Решения:")
                        Console.WriteLine(B / A)
                        Console.WriteLine(-B / A)
                    End If
                End If
            End If
        End If
        Console.Read()
    End Sub
End Module
```

Квадратное уравнение

Это задание также встречалось в ЕГЭ в прошлые годы. Требовалось написать программу, которая решает уравнение $AX^2 + BX + C = 0$ относительно X для действительных чисел A, B, C, введенных с клавиатуры, о которых заведомо известно, что $A \neq 0$, $B \neq 0$, $C \neq 0$. Была написана программа, представленная в листинге 7.10, в которой программист сделал ошибки.

Вам предлагается последовательно выполнить следующие задания:

- указать, где имеется ошибка, связанная с вводом информации;
- найти другие ошибки в листинге 7.10 и обосновать их;
- доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.10. Программа решения квадратного уравнения с наличием ошибок

```
Module Module1
    Sub Main()
        Dim A, B, C, D, X1, X2 As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
        B = Console.ReadLine()
        Console.Write("Введите C ")
```

```
        C = Console.ReadLine()
        Console.Write("Введите X1 ")
        X1 = Console.ReadLine()
        Console.Write("Введите X2 ")
        X2 = Console.ReadLine()
        D = B * B — 4 * A * C
        If D > 0 Then
            X1 = (-B + Math.Sqrt(D)) / (2 * A)
            X2 = (-B — Math.Sqrt(D)) / (2 * A)
            Console.Write("X1=")
            Console.WriteLine(X1)
            Console.Write("X2=")
            Console.WriteLine(X2)
        Else
            Console.Write("Действительных корней нет")
        End If
        Console.Read()
    End Sub
End Module
```

Выполнение заданий

1. Учитывая, что мы решаем уравнение относительно `X`, вводить с клавиатуры его значение не требуется. Поэтому строки

   ```
   Console.Write("Введите X1 ")
   X1 = Console.ReadLine()
   Console.Write("Введите X2 ")
   X2 = Console.ReadLine()
   ```

   следует исключить.

2. В программе не рассматривается случай, когда дискриминант (`D`) равен нулю. Тогда решение уравнения должно выглядеть так:

   `X = -B/(2A).`

   В приведенной программе этот случай не рассматривается, и при `D = 0` программа выдает, что действительных корней нет. Поэтому, задав `A=1`, `B=2`, `C=1`, мы получим согласно программе листинга 7.10 неправильное решение.

С учетом сказанного исправленный вариант программы представлен в листинге 7.11.

Листинг 7.11. Исправленный вариант программы листинга 7.10

```
Module Module1
    Sub Main()
        Dim A, B, C, D, X1, X2 As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
```

```
        Console.Write("Введите B ")
        B = Console.ReadLine()
        Console.Write("Введите C ")
        C = Console.ReadLine()
        D = B * B — 4 * A * C
        If D > 0 Then
            X1 = (-B + Math.Sqrt(D)) / (2 * A)
            X2 = (-B — Math.Sqrt(D)) / (2 * A)
            Console.Write("X1=")
            Console.WriteLine(X1)
            Console.Write("X2=")
            Console.WriteLine(X2)
        Else
            If D = 0 Then
                X1 = (-B) / (2 * A)
                Console.Write("X1=")
                Console.WriteLine(X1)
                Console.Write("X2=")
                Console.WriteLine(X1)
            Else
                Console.Write("Действительных корней нет")
            End If
        End If
        Console.Read()
    End Sub
End Module
```

Решение неравенства

Еще одно задание из Единого государственного экзамена прошлых лет. Требуется написать программу, которая решает неравенство `AX+B>0` относительно `X` для любых чисел `A` и `B`, вводимых с клавиатуры. Все числа считаются действительными. Программист поторопился и написал программу некорректно (листинг 7.12).

Вам предлагается последовательно выполнить следующие задания:

- ❐ указать, где имеется ошибка, связанная с вводом информации;
- ❐ найти другие ошибки в листинге 7.12 и обосновать их;
- ❐ доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.12. Программа решения неравенства с наличием ошибок

```
Module Module1
    Sub Main()
        Dim A, B, X As Single
```

```
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
        B = Console.ReadLine()
        Console.Write("Введите X ")
        X = Console.ReadLine()
        If A = 0 Then
            Console.Write("Любое число")
        Else
            If A > 0 Then
                X = (-B) / (2 * A)
                Console.Write("X > ")
                Console.WriteLine(-B / A)
            Else
                Console.Write("X < ")
                Console.WriteLine(-B / A)
            End If
        End If
        Console.Read()
    End Sub
End Module
```

Выполнение заданий

1. Как и в предыдущих примерах, ввод значения переменной `X` является излишним. Поэтому операторы

   ```
   Console.Write("Введите X ")
   X = Console.ReadLine()
   ```

   следует исключить.

2. При `A=0` программа выдает сообщение о том, что решением является любое число. Однако на самом деле решений неравенства нет.

   Сначала необходимо решить неравенство `AX + B > 0`, и в качестве первого шага выполним очевидное преобразование: `AX > -B`. А теперь рассмотрим следующие ситуации:

   - если `A = 0`, то решением является любое `X`, при условии, что `B > 0`. Это связано с тем, что в этом случае неравенство принимает вид: `0 > -B`;
   - если `A = 0` и `B ≤ 0`, то решений нет. Это также связано с тем, что неравенство принимает вид: `0 > -B`;
   - если `A > 0`, то в этом случае решением неравенства является: `X > -B/A`;
   - если `A < 0`, то решением является: `X < -B/A`.

В листинге 7.13 приведен вариант программной разработки, который соответствует нашим выкладкам.

Листинг 7.13. Исправленный вариант программы листинга 7.12

```
Module Module1
    Sub Main()
        Dim A, B As Single
        Console.Write("Введите A ")
        A = Console.ReadLine()
        Console.Write("Введите B ")
        B = Console.ReadLine()
        If A = 0 Then
            If B > 0 Then
               Console.Write("Любое число является решением неравенства")
            Else
                Console.Write("Решений нет")
            End If
        Else
            If A > 0 Then
                Console.Write("X > ")
                Console.WriteLine(-B / A)
            Else
                Console.Write("X < ")
                Console.WriteLine(-B / A)
            End If
        End If
            Console.Read()
    End Sub
End Module
```

Определение принадлежности множеству

На рис. 7.4 приведен набор фигур:

- круг радиуса 10;
- прямая, параллельная оси ординат, проходящая через точку `X = 5`;
- прямая, проведенная через начало координат под углом 45 градусов к оси абсцисс.

Необходимо написать программу, позволяющую проверить попадание точки в заштрихованную область.

Здесь нужно просто аккуратно записать условие попадания точки в одну из двух областей.

В листинге 7.14 приведен вариант программной разработки.

Листинг 7.14. Проверка попадания точки в область

```
Module Module1
    Sub Main()
        Dim A, X, Y As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        A = Math.Sqrt(X ^ 2 + Y ^ 2)
        If ((A <= 10) And (Y >= (-X)) And (X <= 0)) Or ((X >= 5) _
                 And (Y <= 0) And (A <= 10)) Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

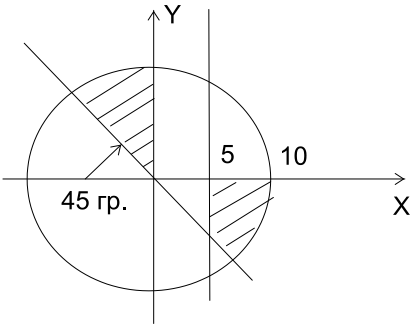

Рис. 7.4. Иллюстрация графической области к листингу 7.14

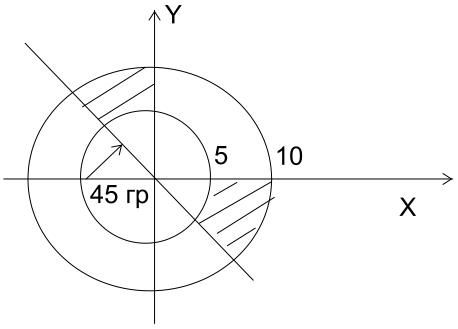

Рис. 7.5. Иллюстрация графической области к листингу 7.15

На рис. 7.5 приведен другой набор фигур:

- ❒ круг радиуса 10;
- ❒ круг радиуса 5;
- ❒ прямая, проведенная через начало координат под углом 45 градусов к оси абсцисс.

Необходимо написать программу, позволяющую проверить попадание точки в заштрихованную область, состоящую из двух фрагментов.

Здесь также следует аккуратно записать условие попадания точки в одну из двух областей. В листинге 7.15 приведен вариант программной разработки.

Листинг 7.15. Проверка попадания точки в область

```
Module Module1
    Sub Main()
        Dim A, X, Y As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        A = Math.Sqrt(X ^ 2 + Y ^ 2)
        If (A <= 10) And (Y >= 5) And (((Y >= (-X)) And (X <= 0)) _
                Or ((X >= (-Y)) And (Y <= 0))) Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

И еще одна ситуация (рис. 7.6) с проверкой попадания точки в заштрихованную область. В листинге 7.16 приведен вариант программной разработки.

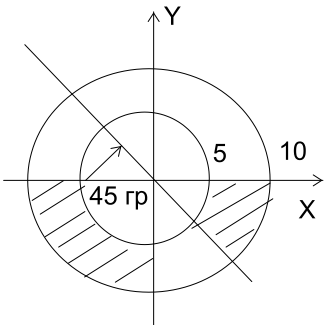

Рис. 7.6. Иллюстрация графической области к листингу 7.16

Листинг 7.16. Проверка попадания точки в область

```
Module Module1
    Sub Main()
        Dim A, X, Y As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        A = Math.Sqrt(X ^ 2 + Y ^ 2)
```

```
        If (A <= 10) And (A >= 5) And (((Y <= 0) And (X <= 0)) _
             Or ((X >= (-Y)) And (Y <= 0))) Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

Последняя ситуация на рассматриваемую тему приведена на рис. 7.7. Один круг радиуса 10 с центром в начале координат, а второй — радиуса 5 с центром в точке (10,0). В листинге 7.17 отражен вариант программной разработки.

Листинг 7.17. Проверка попадания точки в область

```
Module Module1
    Sub Main()
        Dim X, Y, A, B As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        A = Math.Sqrt(X ^ 2 + Y ^ 2)
        B= Math.Sqrt ((X-10) ^ 2 + Y ^ 2)
        If A<=10 And B<=5 Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

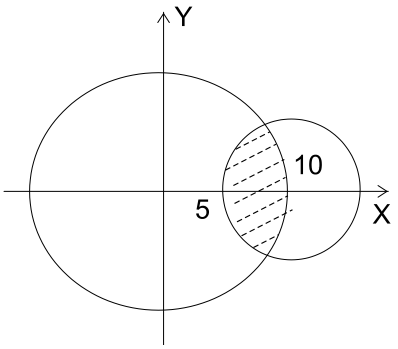

Рис. 7.7. Иллюстрация графической области к листингу 7.17

Метод Монте-Карло

В предыдущих главах мы уже познакомились со стандартной функцией `Rnd`, которая позволяет программно формировать случайные числа. Фактически эта и аналогичные функции в других языках программирования являются *датчиками случайных чисел*.

В математике известен метод Монте-Карло, который позволяет программным способом с помощью датчика случайных чисел решать сложные вычислительные задачи. Сам метод достаточно прост, далее мы его поясним.

Допустим, нам необходимо вычислить площадь сложной фигуры (рис. 7.8). Для этого будем случайным образом формировать точки внутри прямоугольника, окружающего фигуру. На рисунке данный прямоугольник ограничен точками `B1`, `B2`, `B3`, `B4`. Площадь данного прямоугольника равна произведению длин сторон (на рисунке это `C` и `D`). После формирования случайным образом `N` точек в области рассматриваемого прямоугольника какая-то их часть (обозначим это количество символом `M`) попадает в фигуру `S`. В соответствии с методом Монте-Карло формула площади сложной фигуры такова:

$$S = \frac{M}{N} \cdot (C \cdot D). \qquad (7.5)$$

Фактически все содержание метода Монте-Карло заключается в этом соотношении. При этом сам метод не ограничивается только геометрическими задачами вычисления площади. И далее рассмотрим ряд практических примеров, касающихся различных тем, в которых данный метод позволяет быстро найти решение.

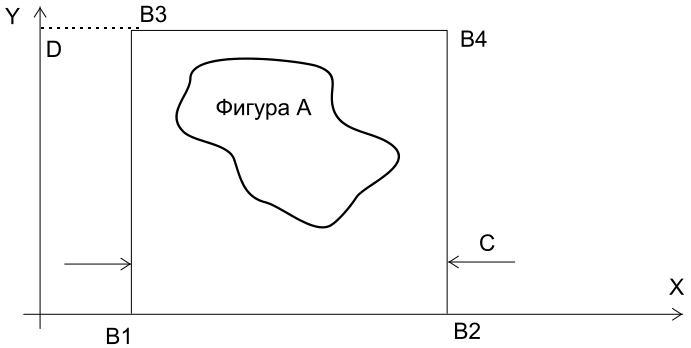

Рис. 7.8. Сложная фигура для вычисления площади

Вычисление площади фигуры

Начнем с уже сформулированной задачи вычисления площади фигуры. Так, необходимо вычислить площадь фрагмента, заключенного внутри линий, которые определяются следующими зависимостями:

$$y = \cos(x),$$
$$y = x^2.$$

Для решения организуем формирование точек в области, ограниченной следующими соотношениями:

$$-2 \le x \le 2,$$

$$0 \le y \le 1.$$

Примечание

В данном случае мы выбрали фигуру, площадь которой легко вычисляется.

Если графически представить исходные линии, то интересующая нас область как раз расположена внутри данного прямоугольника. В листинге 7.18 приведена программная разработка, решающая данную задачу. В программе мы случайным образом формируем 30000 точек, месторасположение которых затем анализируется. Площадь самого прямоугольника, внутри которого формируются точки, равна 4 см^2. В решении задачи мы используем соотношение (7.5).

Листинг 7.18. Использование метода Монте-Карло для вычисления площади фигуры

```
Module Module1
    Sub Main()
        Dim I, M As Integer
        Dim S, Y, Y1, Y2, X As Single
        M = 0
        For I = 1 To 30000
            X = -2 + 4 * Rnd()
            Y = Rnd()
            Y1 = Math.Cos(X)
            Y2 = X ^ 2
            If Y >= Y2 And Y <= Y1 Then
                M = M + 1
            End If
        Next
        S = (M / 30000) * 4
        Console.Write("Площадь равна ")
        Console.Write(S)
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще одну задачу на данную тему. Пусть необходимо вычислить площадь фигуры, ограниченной параболой $y = x^2$, осью абсцисс и прямой $x = 2$.

Организуем формирование точек в области, ограниченной следующими соотношениями:

$$0 \le x \le 2,$$

$$0 \le y \le 4.$$

Если графически представить исходные линии, то интересующая нас область как раз расположена внутри данного прямоугольника. В листинге 7.19 приведена программная разработка, решающая данную задачу. В программе мы случайным образом формируем 30000 точек, месторасположение которых затем анализируется. Площадь самого прямоугольника, внутри которого формируются точки, равна 8 см^2.

Листинг 7.19. Использование метода Монте-Карло для вычисления площади фигуры

```
Module Module1
    Sub Main()
        Dim I,M As Integer
        Dim S,Y,Y1,X As Single
        M=0
        For I=1 To 30000
          X= 2*Rnd
          Y= 4*Rnd
          Y1= X^2
     If Y<=Y1 Then
       M=M+1
    End If
   S=(M/30000)*8
   Console.Write("Площадь равна ")
   Console.Write(S)
   Console.Read()
    End Sub
End Module
```

Моделирование бросания игрального кубика

Необходимо смоделировать бросание игрального кубика. По итогам 10000 бросаний требуется отобразить на экране относительную частоту выпадания каждой грани (в какой доле случаев из 10000 выпала та или иная грань).

Для подсчета относительных частот выпадания каждой грани мы используем массив `A` из 6 элементов. Так, элемент `A(1)` будет содержать значение доли случаев выпадания единицы, `A(2)` — двойки и т. д.

Для моделирования подбрасывания кубика мы воспользовались датчиком случайных чисел в следующей редакции:

```
X = 1+ Int(6*Rnd).
```

В результате такого использования в переменной `X` будут случайно и равновероятно формироваться целые числа от 1 до 6, что соответствует граням кубика.

В листинге 7.20 приведена программная разработка, решающая данную задачу.

Листинг 7.20. Статистика подбрасываний кубика

```
Module Module1
    Sub Main()
        Dim I, X As Integer
        Dim A(6) As Single
        For I = 1 To 6
            A(I) = 0
        Next
        For I = 1 To 10000
            X = 1 + Int(6 * Rnd())
            A(X) = A(X) + 1
        Next
        For I = 1 To 6
            A(I) = A(I) / 10000
            Console.Write("I= ")
            Console.Write(I)
            Console.Write("A= ")
            Console.WriteLine(A(I))
        Next
        Console.Read()
    End Sub
End Module
```

Статистика подбрасывания монет

Необходимо смоделировать эксперимент по подбрасыванию двух монет: подсчитать статистику комбинаций:

- орел, орел;
- орел, решка;
- решка, решка.

Разработка учитывает уже рассмотренные в предыдущем примере технические приемы, поэтому какого-либо комментария не требуется. В листинге 7.21 приведена программная разработка, решающая данную задачу.

Листинг 7.21. Статистика подбрасываний двух монет

```
Module Module1
    Sub Main()
Dim I, X, Y As Integer
        Dim A(3) As Single
        For I = 1 To 3
            A(I) = 0
        Next
```

```
        For I = 1 To 10000
            X = Int(2 * Rnd())
            Y = Int(2 * Rnd())
            If X = 0 And Y = 0 Then
                A(1) = A(1) + 1
            Else
                If X = 1 And Y = 1 Then
                    A(2) = A(2) + 1
                Else
                    A(3) = A(3) + 1
                End If
            End If
        Next
        Console.Write("Доля комбинаций (орел, орел)  равна ")
        Console.WriteLine(A(1) / 10000)
        Console.Write("Доля комбинаций (решка, решка)  равна ")
        Console.WriteLine(A(2) / 10000)
        Console.Write("Доля комбинаций (орел, решка)  равна ")
        Console.Write(A(3) / 10000)
        Console.Read()
    End Sub
End Module
```

Типовые задания из ЕГЭ

В этом разделе приведены типовые задачи из билетов Единого государственного экзамена. В задачах присутствуют готовые листинги программ, которые вам необходимо проанализировать. Также требуется предложить свой правильный вариант программной разработки.

ЗАДАЧА 7.1

Напишите программу, которая проверяет, удовлетворяет ли пара чисел $(x; y)$ двойному неравенству $2 < x^2 + y^2 < 10$.

Программист поторопился и написал программу неправильно (листинг 7.22).

Вам предлагается последовательно выполнить следующие задания:

- ❒ найти ошибки в листинге 7.22 и обосновать их;
- ❒ доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.22. Программа с ошибками к задаче 7.1

```
Module Module1
    Sub Main()
        Dim  X, Y, S As Single
        Console.Write("Введите координату X точки ")
```

```
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        S=X^2 + Y^2
        If S > 2 Or S < 1 Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

Выполнение заданий

В данном случае должны выполняться одновременно два условия, поэтому в операторе условия необходимо использовать ключевое слово `And`.

Вместо строки

```
If S > 2 Or S < 10 Then
```

нужно написать:

```
If S>2 And S<10 Then
```

Программа согласно листингу 7.22 ошибочно (при `X=5` и `Y=5`) выдаст сообщение, что пара точек удовлетворяет неравенству.

В программе используются два одинаковых сообщения о выводе информации.

С учетом сказанного исправленный вариант программы представлен в листинге 7.23.

Листинг 7.23. Исправленный вариант программы листинга 7.22

```
Module Module1
    Sub Main()
        Dim X, Y, S As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        S = X ^ 2 + Y ^ 2
        If (S > 2) And (S < 10) Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 7.2

Напишите программу, которая проверяет, удовлетворяет ли пара чисел $(x; y)$ двойному неравенству $-4 < x^2 - y^2 < 7$.

Программист поторопился и написал программу неправильно (листинг 7.24).

Вам предлагается последовательно выполнить следующие задания:

- ❒ найти ошибки в листинге 7.24 и обосновать их;
- ❒ доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.24. Программа с ошибками к задаче 7.2

```
Module Module1
    Sub Main()
       Dim X, Y, S As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        S = Y ^ 2 — X ^ 2
        If S > -4 And S < 7 Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

Выполнение заданий

В листинге 7.24 неправильно написано выражение, составляющее компоненту неравенства:

```
S=Y^2 — X^2
```

Нужно написать:

```
S=X^2 — Y^2
```

Программа согласно листингу 7.24 ошибочно (при `X=0` и `Y=10`) выдаст сообщение, что пара точек удовлетворяет неравенству.

В программе используются два одинаковых сообщения о выводе информации.

С учетом сказанного исправленный вариант программы представлен в листинге 7.25.

Листинг 7.25. Исправленный вариант программы листинга 7.24

```
Module Module1
    Sub Main()
       Dim X, Y, S As Single
```

```
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        S = X ^ 2 — Y ^ 2
        If S > -4 And S < 7 Then
            Console.Write("Да, попадает")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
    End Sub
End Module
```

ЗАДАЧА 7.3

Напишите программу, которая проверяет, удовлетворяет ли пара чисел $(x; y)$ двойному неравенству $5 \le x^2 + y^2 \le 9$.

Программист поторопился и написал программу неправильно (листинг 7.26).

Вам предлагается последовательно выполнить следующие задания:

- найти ошибки в листинге 7.26 и обосновать их;
- доработать программу так, чтобы не было случаев ее неправильной работы.

Листинг 7.26. Программа с ошибками к задаче 7.3

```
Module Module1
    Sub Main()
        Dim X, Y, S As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        S = X ^ 2 + Y ^ 2
        If (S >= 5) Or (S <= 9) Then
            Console.Write("Удовлетворяет")
        Else
            Console.Write("Нет")
        End If
        Console.Read()
        Console.Read()
    End Sub
End Module
```

Выполнение заданий

В листинге 7.26 неправильно написано условие:

```
(S >= 5) Or (S <= 9)
```

Нужно написать:

```
If (S >= 5) And ( S <= 9)
```

Программа согласно листингу 7.26 ошибочно при `X=1` и `Y=10` выдаст сообщение, что пара точек удовлетворяет неравенству.

В программе используются два одинаковых сообщения о выводе информации.

С учетом сказанного исправленный вариант программы представлен в листинге 7.27.

Листинг 7.27. Исправленный вариант программы листинга 7.26

```
Module Module1
    Sub Main()
        Dim X, Y, S As Single
        Console.Write("Введите координату X точки ")
        X = Console.ReadLine()
        Console.Write("Введите координату Y точки ")
        Y = Console.ReadLine()
        S = X ^ 2 + Y ^ 2
        If (S >= 5) And (S <= 9) Then
            Console.Write("Удовлетворяет")
        Else
            Console.Write("Нет")
        End If
    End Sub
End Module
```

ГЛАВА 8

Обработка данных

Эта глава непосредственно связана с наиболее трудоемкой компонентой Единого государственного экзамена. В целом программирование и алгоритмизация задач являются основным содержанием части *C* билетов Единого государственного экзамена, где содержатся четыре задания, которые ориентированы на проверку умения составлять и анализировать алгоритмы. При этом данная часть традиционно вызывает наибольшие трудности у учащихся. Это вполне объяснимо, т. к. здесь требуется как самостоятельное написание достаточно сложных программ, так и анализ и коррекция уже готовых, часто непростых разработок. И данная глава преследует цель подготовить учащихся именно к части *C* билетов Единого государственного экзамена. Глава состоит в основном из подробного рассмотрения заданий выносимых на Единый государственный экзамен в прошлые годы. Вместе с этим в данной главе приводится разбор типовых заданий по программированию алгоритмов.

Анализ тестирования учащихся

Рассмотрим следующую задачу. Пусть в некотором вузе студенты проходят предварительное тестирование, по результатам которого они могут быть допущены к ранней сдаче экзаменов. Тестирование проводится по трем темам. По каждой теме учащийся может набрать от 0 до 100 баллов. При этом к ранней сдаче экзамена допускаются учащиеся, набравшие по результатам тестирования не менее 30 баллов по каждой из трех тем. Известно, что общее количество участников тестирования не превосходит 300.

Необходимо обеспечить ввод начальных данных в программу. При этом в первой строке вводится количество студентов, принимавших участие в тестировании (`N`).

Далее следуют `N` вводимых строк, имеющих следующий формат:

<фамилия> <имя> <баллы>,

где:

- *фамилия* — это информация о фамилии студента, являющаяся строкой (в плане типа данных);
- *имя* — строка (также речь идет о типе данных);

- *баллы* — три целых числа, разделенных пробелами (эти числа соответствуют баллам, набранным по результатам тестирования по каждой из трех тем).

При этом фамилия и имя, а также имя и баллы разделены одним пробелом.

Пример возможной введенной строки с информацией по одному студенту:

```
Петренко Наталья 58 75 35
```

Разработанная программа должна выводить на экран фамилии и имена студентов, допущенных к сдаче экзамена. При этом фамилии учащихся (вместе с именами) можно выводить в произвольном порядке.

Перед написанием программы поясним алгоритм необходимых действий для решения данной задачи. После ввода `N` (числа студентов) необходимо организовать цикл, в котором последовательно вводятся строки с данными в определенном ранее формате (в каждой строке вносятся фамилия, имя и баллы). Для хранения информации о фамилии и имени предназначен массив `Mass` размерностью 300 элементов (в качестве типа данных элементов массива укажем `String` — строку). Таким образом, в элемент массива `Mass` с индексом `J` заносится информация о фамилии и имени `J`-го студента. Еще один массив (имя которого выберем — `Indicator`) также из трехсот элементов предназначен для фиксирования прохождения (или не прохождения) каждым учащимся всех трех тем. Условно будем считать, что если `J`-м студентом все три темы успешно пройдены (по каждой набрано не менее 30 баллов), то в элементе `Indicator(J)` это отмечается значением 1, в противном случае в данный элемент массива `Indicator(J)` записывается 0. Описанные действия представляют собой часть решения поставленной задачи. Они заключаются во вводе и первоначальной обработке входной информации.

На следующем этапе решения поставленной задачи осуществляется вывод на экран информации об учащихся, допущенных к ранней сдаче экзамена. Для этого используется информация, сформированная на предыдущем этапе в массивах `Mass` и `Indicator`. Данный этап решения задачи сводится к организации нового цикла по перебору всех учащихся. При этом если у очередного `J`-го учащегося значение элемента массива `Indicator(J)` равно 1, то на экран выводится элемент массива `Mass(J)`, содержащий фамилию и имя учащегося. Таким образом, в результате описанных действий мы достигаем поставленной цели.

В листинге 8.1 приведена программа, функционирующая в соответствии с данным алгоритмом. Для подсчета числа символов в очередной введенной строке с информацией по учащемуся мы воспользовались стандартной функцией `Len`. Далее в программе выполняется последовательное извлечение символов из введенной строки с помощью еще одной стандартной функции `Mid`. Заметим, что для извлечения информации о фамилии, имени и баллах здесь мы использовали цикл с проверкой условия в конце цикла.

Листинг 8.1. Анализ тестирования учащихся (вариант 1)

```
Module Module1
Sub Main()
  Dim Mass(300) As String
```

```
Dim Indicator(300) As Integer
Dim M1, M2, M3 As String ' Переменные для хранения результатов тестов
                         ' по трем темам
Dim N, I, J, Num As Integer
Dim C As String ' Переменная для хранения вводимой строки
Dim Cs As Char  ' Переменная для хранения очередного символа
                ' из введенной строки
N = Console.ReadLine() ' Считывание числа учащихся
For I = 1 To N  ' Цикл для считывания строк с информацией
  C = Console.ReadLine() ' Считывание строки с данными об очередном
                         '  учащемся
  Num = Len(C) ' Вычисление числа символов во введенной строке
  Mass(i)=""   ' Очистка элемента массива перед последующим внесением
               ' информации о фамилии и имени учащегося
  M1 = ""      ' Начальная инициализация переменных
  M2 = ""      ' для считывания баллов по трем темам
  M3 = ""
  J = 1 ' Переменная J используется в качестве счетчика символов во
        ' введенной строке
  Do ' Извлечение информации о фамилии и внесение ее в элемент
     '  массива Mass
       Cs = Mid(C, J, 1)
       J = J + 1
       Mass(I) = Mass(I) + Cs
  Loop While Cs <> " "
  Do  ' Извлечение информации об имени и добавление ее в
      ' элемент массива Mass
      Cs = Mid(C, J, 1)
      J = J + 1
      Mass(I) = Mass(I) + Cs
  Loop While Cs <> " "
  Do ' Извлечение информации о баллах, набранных по первой теме,
     ' и внесение их в переменную M1
     Cs = Mid(C, J, 1)
     J = J + 1
     M1 = M1 + Cs
  Loop While Cs <> " "
  Do ' Извлечение информации о баллах, набранных по второй теме,
     ' и внесение их в переменную M2
     Cs = Mid(C, J, 1)
     J = J + 1
     M2 = M2 + Cs
  Loop While Cs <> " "
  M3 = Mid(C, J, Num — J + 1) ' Извлечение информации о баллах,
     '  набранных по третьей теме, и внесение их в переменную M3
  ' Проверка условия допуска учащегося к сдаче экзамена
  If Val(M1) < 30 Or Val(M2) < 30 Or Val(M3) < 30 Then
     Indicator(I) = 0
```

```
    Else
       Indicator(I) = 1
    End If
   Next ' Завершение цикла считывания строк
   For I = 1 To N ' Цикл вывода на экран списка учащихся,
                  ' допущенных к ранней сдачи экзамена
      If Indicator(I) = 1 Then
         Console.WriteLine(Mass(i))
      End If
   Next
   Console.Read()
  End Sub
End Module
```

В результате выполнения программы, приведенной в листинге 8.1, мы увидим на экране набор строк, в каждой из которых присутствует фамилия и имя учащегося, допущенного к ранней сдаче экзамена.

Можно предложить следующее улучшение данной программы. Понятно, что если по какой-либо теме у очередного учащегося балл оказался меньше 30, то количество баллов, набранных им по последующим темам, можно уже не проверять. В листинге 8.2 приведена программа для такого варианта построения алгоритма.

Листинг 8.2. Анализ тестирования учащихся (вариант 2)

```
Module Module1
Sub Main()
  Dim Mass(300) As String
  Dim Indicator(300) As Integer
  Dim M1, M2, M3 As String' Переменные для хранения результатов тестов
                           ' по трем темам
  Dim N, I, J, Num As Integer
  Dim C As String ' Переменная для хранения вводимой строки
  Dim Cs As Char  ' Переменная для хранения очередного символа
                  ' из введенной строки
  N = Console.ReadLine()    ' Считывание числа учащихся
  For I = 1 To N            ' Цикл для считывания строк с информацией
    C = Console.ReadLine() ' Считывание строки с данными об очередном
                           '  учащемся
    Num = Len(C) ' Вычисление числа символов во введенной строке
    Mass(I)=""   ' Очистка элемента массива для последующего внесения
                 ' информации о фамилии и имени учащегося
    M1 = ""
    M2 = ""
    M3 = ""
    J = 1 ' Переменная J используется в качестве счетчика символов во
          ' введенной строке
```

```
 Do    ' Извлечение информации о фамилии и внесение ее в элемент
       ' массива Mass
      Cs = Mid(C, J, 1)
      J = J + 1
      Mass(I) = Mass(I) + Cs
 Loop While Cs <> " "
 Do    ' Извлечение информации об имени и добавление ее в
       ' элемент массива Mass
     Cs = Mid(C, J, 1)
     J = J + 1
     Mass(I) = Mass(I) + Cs
 Loop While Cs <> " "
 Do    ' Извлечение информации о баллах, набранных по первой теме,
       ' и внесение их в переменную M1
    Cs = Mid(C, J, 1)
    J = J + 1
    M1 = M1 + Cs
 Loop While Cs <> " "
 If Val(M1) < 30 Then ' Если первая тема не пройдена, то сразу
                      ' отмечаем это в элементе Indicator(I)
    Indicator(I) = 0
 Else    ' Если по первой теме набрано не менее 30-ти баллов
    Do   ' Извлечение информации о баллах, набранных по второй теме,
         ' и внесение их в переменную M2
       Cs = Mid(C, J, 1)
       J = J + 1
       M2 = M2 + Cs
    Loop While Cs <> " "
    If Val(M2) < 30 Then ' Если вторая тема не пройдена, то
                         ' отмечаем это в элементе Indicator(I)
          Indicator(I) = 0
    Else ' Если по второй теме набрано не менее 30-ти баллов
          M3 = Mid(C, J, Num — J + 1)
          If Val(M3) < 30 Then
               Indicator(I) = 0
          Else
               Indicator(I) = 1
          End If
     End If
   End If      ' Окончание просмотра тем
Next           ' Завершение цикла считывания строк
For I = 1 To N ' Цикл вывода на экран списка учащихся,
               ' допущенных к ранней сдачи экзамена
   If Indicator(I) = 1 Then
      Console.WriteLine(Mass(I))
   End If
 Next
```

```
    Console.Read()
  End Sub
End Module
```

В программах, приведенных в листингах 8.1 и 8.2, мы использовали массивы для хранения информации о фамилиях (с именами) учащихся и их допуске к ранней сдаче экзамена. Однако, учитывая, что по условию задания требуется только вывод на экран студентов, допущенных к досрочной сдаче экзамена, то можно обойтись без хранения в памяти лишних массивов. В этом случае вывод на экран фамилий учащихся реализуется в цикле считывания строк исходных данных. После считывания строки осуществляется весь необходимый анализ ее содержимого. В листинге 8.3 приведен данный вариант решения поставленной задачи. Здесь мы вместо массива `Mass` воспользовались просто переменной строкового типа с аналогичным именем.

> ***Примечание***
>
> Недостаток данного варианта решения задачи заключается в том, что вывод на экран учащихся, допущенных к ранней сдаче экзамена, выполняется вперемежку с отображением вводимых строк.

Листинг 8.3. Анализ тестирования учащихся (вариант 3)

```
Module Module1
Sub Main()
  Dim Mass As String
  Dim M1, M2, M3 As String
  Dim N, I, J, Num As Integer
  Dim C As String
  Dim Cs As Char
  N = Console.ReadLine()   ' Считывание числа учащихся
  For I = 1 To N  ' Цикл для считывания входных строк с информацией
    C = Console.ReadLine() ' Считывание строки с данными об очередном
                           ' учащемся
    Num = Len(C) ' Вычисление числа символов во введенной строке
    Mass=""      ' Инициализация переменной для последующего внесения
                 ' информации о фамилии и имени учащегося
    M1 = ""
    M2 = ""
    M3 = ""
    J = 1 ' Переменная J используется в качестве счетчика символов во
          ' введенной строке
    Do  ' Извлечение информации о фамилии и внесение ее в переменную Mass
        Cs = Mid(C, J, 1)
        J = J + 1
        Mass = Mass + Cs
    Loop While Cs <> " "
```

```
    Do  ' Извлечение информации об имени и добавление его в Mass
        Cs = Mid(C, J, 1)
        J = J + 1
        Mass = Mass + Cs
    Loop While Cs <> " "
    Do  ' Извлечение информации о баллах, набранных по первой теме,
        ' и внесение ее в переменную M1
       Cs = Mid(C, J, 1)
       J = J + 1
       M1 = M1 + Cs
    Loop While Cs <> " "
    Do  ' Извлечение информации о баллах, набранных по второй теме,
        ' и внесение их в переменную M2
       Cs = Mid(C, J, 1)
       J = J + 1
       M2 = M2 + Cs
    Loop While Cs <> " "
    M3 = Mid(C, J, Num — J + 1) ' Извлечение информации о баллах,
          ' набранных по третьей теме, и внесение их в переменную M3
          ' Проверка условия допуска учащегося к сдаче экзамена
    If Val(M1) >= 30 And Val(M2) >= 30 And Val(M3) >= 30 Then
          Console.Write("Допущен(а) ")
          Console.WriteLine(Mass)
    End If
   Next   ' Завершение цикла считывания строк
   Console.Read()
  End Sub
End Module
```

Отчет по олимпиаде

В этой задаче на вход программы подаются сведения о номерах школ учащихся, которые участвовали в олимпиаде. При этом в первой строке вводится количество учащихся (N). Далее следуют N строк, имеющих следующий формат:

<фамилия> <инициалы> <номер школы>,

где:

- *фамилия* — фамилия учащегося (тип данных — строка);
- *инициалы* — строка, состоящая из 4-х символов (буква, точка, буква, точка);
- *номер школы* — однозначное или двузначное число (от 1 до 99).

При этом фамилия и инициалы, а также инициалы и номер школы разделены одним пробелом.

Один из вариантов ввода строки об участнике с клавиатуры выглядит так:

```
Петренко Н.С. 55
```

В программе необходимо обеспечить вывод на экран информации о том, в какой школе было меньше всего участников (таких школ может быть несколько). При этом необходимо отразить информацию только по школам, пославшим хотя бы одного участника. Следует также учесть, что `N<1000`.

Для работы нам понадобится целочисленный массив `Mass` из 99 элементов: элемент с индексом 1 будет хранить сведения о количестве участников из школы с номером 1, элемент с индексом 2 будет хранить сведения о количестве участников из школы с номером 2 и т. д.

Во все элементы массива `Mass` мы предварительно заносим нули (перед началом подсчета от каждой школы 0 участников). Как и в предыдущем примере, после ввода числа учащихся `N` осуществляется последовательный ввод `N` строк указанного ранее формата. Так как нас интересует только из какой школы был каждый участник (фамилии с инициалами участников нам не важны), то следует просто прибавить единицу в определенный элемент массива. Например, если в очередной вводимой строке считан учащийся из школы с номером 25, то в элемент массива `Mass` с индексом 25 добавляется единица. После такой обработки всех строк с данными необходимо найти минимальный элемент в массиве (только при этом необходимо учесть, что значение элемента не должно равняться нулю). В завершающей части алгоритма с помощью еще одного цикла на экран выводятся школы, удовлетворяющие указанному условию. В листинге 8.4 представлена программа, которая реализует рассмотренный алгоритм.

Листинг 8.4. Формирование отчета по олимпиаде (вариант 1)

```
Module Module1
Sub Main()
  Dim Mass(99) As Integer
  Dim N, I, J, Min, Num As Integer
  Dim C, Ms As String
  Dim Cs As Char
  For I = 1 To 99          ' Начальная инициализация массива
     Mass(I)=0
  Next
  N = Console.ReadLine()   ' Считывание числа учащихся
  For I = 1 To N           ' Цикл для считывания строк с информацией
    C = Console.ReadLine() ' Считывание строки с данными об очередном
                           '  учащемся
    Num = Len(C) ' Вычисление числа символов во введенной строке
    J = 1
    Do           ' Считывание из входной строки информации о фамилии
      Cs = Mid(C, J, 1)
      J = J + 1
    Loop While Cs <> " "
    Do           ' Считывание из входной строки информации об инициалах
       Cs = Mid(C, J, 1)
       J = J + 1
```

```
    Loop While Cs <> " "
                ' Считывание из входной строки информации о номере школы
    Ms = Mid(C, J, Num — J + 1)
    Mass(Val(Ms))= Mass(Val(Ms))+1
 Next
 Min = N+1       ' Поиск минимума в массиве Mass
 For I = 1 To 99
   If Mass(I) < Min And Mass(I) > 0 Then
       Min=Mass(I)
   End If
 Next
 For I=1 To 99
   If Mass(I) = Min Then
      Console.WriteLine(I)
   End If
 Next
 Console.Read()
End Sub
End Module
```

Можно предложить более эффективный алгоритм, идея которого основывается на том, что в каждой введенной строке нам необходимо только считать номер школы, который расположен в конце строки. При этом номер школы может составлять один или два символа (по условию номер школы однозначное либо двузначное число). В любом случае предыдущий символ перед номером школы является пробелом.

Таким образом, мы можем считать второй с конца строки символ, и если он является пробелом, то следует сформировать номер школы из последнего символа строки. В противном случае необходимо сформировать номер школы из двух последних символов строки.

В листинге 8.5 представлена программа, которая реализует рассмотренный алгоритм.

Листинг 8.5. Формирование отчета по олимпиаде (вариант 2)

```
Module Module1
Sub Main()
  Dim Mass(99) As Integer
  Dim N, I, Min, Num As Integer
  Dim C As String
  Dim Cs As Char
  For I= 1 To 99          ' Начальная инициализация массива
     Mass(I)=0
  Next
  N = Console.ReadLine()   ' Считывание числа учащихся
```

```
  For I = 1 To N           ' Цикл для считывания строк с информацией
    C = Console.ReadLine() ' Считывание строки с данными об очередном
                           ' учащемся
    Num = Len(C)          ' Вычисление числа символов во введенной строке
    Cs = Mid(C, Num — 1,1) ' Выделение предпоследнего символа в строке
    If Cs = " " Then
       Mass(Val(Mid(C, Num, 1)))= Mass(Val(Mid(C, Num, 1)))+ 1
    Else
      Mass(Val(Mid(C, Num — 1, 2)))= Mass(Val(Mid(C, Num — 1, 2)))+ 1
    End If
 Next
Min = N+1                  ' Поиск минимума в массиве Mass
For I = 1 To 99
   If Mass(I) < Min And Mass(I)>0 Then
       Min=Mass(I)
   End If
Next
For I= 1 To 99
   If Mass(I)=Min Then
     Console.WriteLine(I)
   End If
  Next
  Console.Read()
 End Sub
End Module
```

Сертификаты

А теперь рассмотрим такую задачу. Пусть некоторый вуз принимает сертификаты по трем предметам (математика, русский язык и информатика). Программа получает входную информацию об этих сертификатах. Так, в первой строке вводится количество абитуриентов — `N`. После этого вводится еще `N` строк, каждая из которых имеет следующий формат:

<фамилия с инициалами> <оценка1> <оценка1> <оценка1>,

где:

- *фамилия с инициалами* — строка, содержащая не более 100 символов (при этом в данной строке пробелов нет, а для разделения фамилии и инициалов используется символ нижнего подчеркивания);
- *оценка1* — целое число от 0 до 100;
- *оценка2* — целое число от 0 до 100;
- *оценка3* — целое число от 0 до 100.

Пример входной строки:

```
Петров_О.Л. 45 76 23
```

Необходимо составить список учеников, показавших три лучших результата. Если окажется несколько человек, которые показали такой же результат, то их также нужно включить в данный список.

Следует учитывать, что `N<300`.

В листинге 8.6 представлена программа, которая реализует необходимый алгоритм. В заключительной ее части выполняется поиск трех лучших результатов. Для этого предназначены три переменные целого типа:

- `Max1` — для лучшего результата;
- `Max2` — следующего за лучшим;
- `Max3` — следующего за двумя лучшими.

Листинг 8.6. Формирование списка лучших сертификатов

```
Module Module1
    Sub Main()
        Dim Mass(100) As String
        Dim Points(100) As Integer
        Dim N, I, J, Num, Max1, Max2, Max3 As Integer
        Dim M1, M2, M3 As String
        Dim C As String
        Dim Cs As Char
        N = Console.ReadLine()  ' Считывание числа учащихся
        For I = 1 To N          ' Цикл для считывания строк с информацией
           Mass(I) = ""
           C = Console.ReadLine() ' Считывание строки с данными об
                                  ' очередном учащемся
            Num = Len(C) ' Вычисление числа символов во введенной строке
            J = 1        ' Переменная J используется в качестве счетчика
                         ' символов во введенной строке
            Do  ' Извлечение информации о фамилии с инициалами и
                ' внесение их в элемент массива Mass
                Cs = Mid(C, J, 1)
                J = J + 1
                Mass(I) = Mass(I) + Cs
            Loop While Cs <> " "
            M1 = ""
            Do ' Извлечение информации о баллах, набранных по первому
               ' предмету, и внесение их в переменную M1
                Cs = Mid(C, J, 1)
                J = J + 1
                M1 = M1 + Cs
            Loop While Cs <> " "
            M2 = ""
```

```
            Do ' Извлечение информации о баллах, набранных по второму
               ' предмету, и внесение их в переменную M2
                Cs = Mid(C, J, 1)
                J = J + 1
                M2 = M2 + Cs
            Loop While Cs <> " "
            M3 = Mid(C, J, Num — J + 1) ' Извлечение информации о баллах,
               ' набранных по третьему предмету, и внесение их в переменную M3
            Points(I) = Val(M1) + Val(M2) + Val(M3)
        Next
        Max1 = -1
        Max2 = -1
        Max3 = -1
        For I = 1 To N
            If Points(I) > Max1 Then
                Max3 = Max2
                Max2 = Max1
                Max1 = Points(I)
            ElseIf Points(I) > Max2 Then
                Max3 = Max2
                Max2 = Points(I)
            ElseIf Points(I) > Max3 Then
                Max3 = Points(I)
            End If
        Next
        For I = 1 To N
            If Points(I) >= Max3 Then
                Console.Write(Mass(I))
                Console.Write(" ")
                Console.WriteLine(Points(I))
            End If
        Next
        Console.Read()
    End Sub
End Module
```

В предложенном варианте разработки мы выполнили условие задания, однако фамилии учащихся при таком алгоритме отражаются не в порядке уменьшения набранных баллов. Поэтому исправим эту недоработку. Второй вариант программы поиска лучших сертификатов приведен в листинге 8.7. Здесь мы сначала выводим информацию по набравшим максимальное количество баллов (`Max1`). При этом в переменной `Cols` подсчитываем количество таких учащихся. Если их уже не менее трех, то далее вывод учащихся с меньшими баллами уже не осуществляется. В противном случае выводятся фамилии учащихся, которые показали следующий результат. Если же общее количество выведенных на экран фамилий не менее трех, то далее вывод учащихся с меньшими баллами уже не проводится. В противном случае выводятся фамилии учащихся, которые показали третий результат.

Листинг 8.7. Список наилучших сертификатов (вариант 2)

```
Module Module1
    Sub Main()
        Dim Mass(100) As String
        Dim Points(100) As Integer
        Dim N, I, J, Num, Max1, Max2, Max3, Cols As Integer
        Dim M1, M2, M3 As String
        Dim C As String
        Dim Cs As Char
        N = Console.ReadLine() ' Считывание числа учащихся
        For I = 1 To N         ' Цикл для считывания строк с информацией
            Mass(I) = ""
            C = Console.ReadLine() ' Считывание строки с данными об
                                   ' очередном учащемся
            Num = Len(C) ' Вычисление числа символов во введенной строке
            J = 1        ' Переменная J используется в качестве счетчика
                         ' символов во введенной строке
            Do ' Извлечение информации о фамилии и внесение ее в элемент
               ' массива Mass
                Cs = Mid(C, J, 1)
                J = J + 1
                Mass(I) = Mass(I) + Cs
            Loop While Cs <> " "
            M1 = ""
            Do 'Извлечение информации о баллах, набранных по первому
               ' предмету, и внесение их в переменную M1
                Cs = Mid(C, J, 1)
                J = J + 1
                M1 = M1 + Cs
            Loop While Cs <> " "
            M2 = ""
            Do ' Извлечение информации о баллах, набранных по второму
               ' предмету, и внесение их в переменную M2
                Cs = Mid(C, J, 1)
                J = J + 1
                M2 = M2 + Cs
            Loop While Cs <> " "
            M3 = Mid(C, J, Num — J + 1) ' Извлечение информации о баллах,
               ' набранных по третьему предмету, и внесение их в переменную M3
            Points(I) = Val(M1) + Val(M2) + Val(M3)
        Next
        Max1 = -1
        Max2 = -1
        Max3 = -1
        For I = 1 To N
            If Points(I) > Max1 Then
                Max3 = Max2
```

```
                Max2 = Max1
                Max1 = Points(I)
            ElseIf Points(I) > Max2 Then
                Max3 = Max2
                Max2 = Points(I)
            ElseIf Points(I) > Max3 Then
                Max3 = Points(I)
            End If
        Next
        Cols = 0
        For I = 1 To N
            If Points(I) = Max1 Then
                Console.Write(Mass(I))
                Console.Write(" ")
                Console.WriteLine(Points(I))
                Cols = Cols + 1
            End If
        Next
        If Cols < 3 And Max1 <> Max2 Then
            For I = 1 To N
                If Points(I) = Max2 Then
                    Console.Write(Mass(I))
                    Console.Write(" ")
                    Console.WriteLine(Points(I))
                    Cols = Cols + 1
                End If
            Next
        End If
        If Cols < 3 And Max1 <> Max3 And Max2 <> Max3 Then
            For I = 1 To N
                If Points(I) = Max3 Then
                    Console.Write(Mass(I))
                    Console.Write(" ")
                    Console.WriteLine(Points(I))
                End If
            Next
        End If
        Console.Read()
    End Sub
End Module
```

Результаты экзамена

Теперь рассмотрим задачу, где на вход программы подаются сведения о результатах экзаменов. Общее количество учеников не превосходит 100. Так, в первой строке вводится количество учеников `N`.

После этого вводится еще N строк, каждая из которых имеет следующий формат:

<фамилия> <имя> <баллы>,

где:

- *фамилия* — строка, содержащая не более 20 символов;
- *имя* — строка, содержащая не более 15 символов;
- *баллы* — четыре числа, разделенные пробелами.

При этом фамилия и имя, а также имя и баллы разделены одним пробелом. Необходимо составить программу, которая будет выводить список учеников, набравших максимальную сумму баллов, а также количество таких учеников.

В листинге 8.8 представлена необходимая программа, которая реализует рассмотренный алгоритм.

Листинг 8.8. Формирование списка лучших учеников

```
Module Module1
    Sub Main()
        Dim Mass(100) As String
        Dim Points(100) As Integer
        Dim N, I, J, Max, K, Num As Integer
        Dim M1, M2, M3, M4 As String
        Dim C As String
        Dim Cs As Char
        N = Console.ReadLine() ' Считывание числа учащихся
        For I = 1 To N         ' Цикл для считывания строк с информацией
            Mass(I) = ""
            C = Console.ReadLine() ' Считывание строки с данными об
                                   ' очередном  учащемся
            Num = Len(C) ' Вычисление числа символов во введенной строке
            J = 1        ' Переменная J используется в качестве счетчика
                         ' символов во введенной строке
            Do ' Извлечение информации о фамилии и внесение ее в элемент
               ' массива Mass
                Cs = Mid(C, J, 1)
                J = J + 1
                Mass(I) = Mass(I) + Cs
            Loop While Cs <> " "
            Do ' Извлечение информации об имени и добавление ее в элемент
               ' массива Mass
                Cs = Mid(C, J, 1)
                J = J + 1
                Mass(I) = Mass(I) + Cs
            Loop While Cs <> " "
            M1 = ""
```

```
            Do 'Извлечение информации о баллах, набранных по первому
               ' предмету, и внесение их в переменную M1
                Cs = Mid(C, J, 1)
                J = J + 1
                M1 = M1 + Cs
            Loop While Cs <> " "
            M2 = ""
            Do ' Извлечение информации о баллах, набранных по второму
               ' предмету, и внесение их в переменную M2
                Cs = Mid(C, J, 1)
                J = J + 1
                M2 = M2 + Cs
            Loop While Cs <> " "
            M3 = ""
            Do ' Извлечение информации о баллах, набранных по третьему
               ' предмету, и внесение их в переменную M3
                Cs = Mid(C, J, 1)
                J = J + 1
                M3 = M3 + Cs
            Loop While Cs <> " "
            M4 = Mid(C, J, Num — J + 1) ' Извлечение информации о баллах,
               ' набранных по четвертому предмету, и внесение их в переменную M4
            Points(I) = Val(M1) + Val(M2) + Val(M3) + Val(M4)
        Next
        Max = Points(1)
        For J = 2 To N
            If Points(J) > Max Then Max = Points(J)
        Next
        K = 0
        For J = 1 To N
            If Points(J) = Max Then
                Console.WriteLine(Mass(J))
                K = K + 1
            End If
        Next
        Console.Write(" Число учеников, получивших максимальный балл ")
        Console.WriteLine(K)
        Console.Read()
    End Sub
End Module
```

Полупроходной балл

Допустим, что на вход программы подаются сведения о сертификатах абитуриентов. Общее количество абитуриентов не превосходит 500. В первой строке программы вводится количество абитуриентов N, а во второй вводится количество мест

K, на которые они претендуют. После этого вводится еще N строк, каждая из которых имеет следующий формат:

<*фамилия и инициалы*> <*баллы 1*> <*баллы 2*> <*баллы 3*>,

где:

- *фамилия и инициалы* — строка, содержащая фамилию и инициалы;
- *баллы 1* — строка, содержащая баллы по первому предмету;
- *баллы 2* — строка, содержащая баллы по второму предмету;
- *баллы 3* — строка, содержащая баллы по третьему предмету.

Баллы представляют собой три числа, разделенные пробелами (каждое число является целым в интервале от 0 до 100).

При этом фамилия с инициалами и баллы разделены одним пробелом (считается, что между фамилией и инициалами пробела нет — просто инициалы пишутся большими буквами). Необходимо составить программу, которая будет выводить список абитуриентов, набравших полупроходной балл, а также количество таких учеников. Или же необходимо указать, что полупроходного балла нет.

> ***ПРИМЕЧАНИЕ***
>
> *Полупроходным баллом* называется такой балл, когда лишь часть абитуриентов, набравших его, попадает в число поступивших.

В листинге 8.9 представлена программа, которая реализует необходимый алгоритм. Массив Points используется для подсчета числа абитуриентов, набравших конкретное количество баллов. Например, элемент Points(300) содержит число учащихся, набравших 300 баллов, соответственно Points(299) — число учащихся, набравших 299 баллов.

В массивах Mass и Balls фиксируются соответственно фамилия (с инициалами) и число баллов, набранных абитуриентом. Например, Mass(1) содержит фамилию первого учащегося, а в Balls(1) фиксируется число баллов, которое данный учащийся набрал.

После ввода всех данных осуществляется поиск полупроходного балла. Для этого мы использовали цикл с предусловием. В результате выполнения данного цикла в переменной J фиксируется значение полупроходного балла, а в переменной S — число учащихся, которые претендуют на K мест. В число S входят те, кто набрал или проходной балл, или полупроходной. Далее в случае наличия полупроходного балла (когда S > K) осуществляется вывод фамилий учащихся, которые его набрали.

Листинг 8.9. Формирование списка учеников с полупроходным баллом

```
Module Module1
    Sub Main()
        Dim Mass(500) As String
        Dim Balls(500) As Integer
```

```
Dim Points(300) As Integer
Dim N, I, J, Sum, S, M, K, Num As Integer
Dim M1, M2, M3 As String
Dim C As String
Dim Cs As Char
N = Console.ReadLine()      ' Считывание числа абитуриентов
K = Console.ReadLine()      ' Считывание числа мест
                  ' Подготовка массива для подсчета результатов
For J = 0 To 300
    Points(J) = 0
Next
For I = 1 To N   ' Цикл для считывания строк с информацией
    Mass(I) = ""
    C = Console.ReadLine() ' Считывание строки с данными об
                           ' очередном учащемся
    Num = Len(C) ' Вычисление числа символов во введенной строке
    J = 1        ' Переменная J используется в качестве счетчика
                 ' символов во введенной строке
    Do           ' Извлечение информации о фамилии с инициалами
        Cs = Mid(C, J, 1)
        J = J + 1
        Mass(I) = Mass(I) + Cs
    Loop While Cs <> " "
    M1 = ""
    Do          ' Извлечение информации о баллах, набранных по
                ' первому предмету, и внесение их в переменную M1
        Cs = Mid(C, J, 1)
        J = J + 1
        M1 = M1 + Cs
    Loop While Cs <> " "
    M2 = ""
    Do          ' Извлечение информации о баллах, набранных по
                ' второму предмету, и внесение их в переменную M2
        Cs = Mid(C, J, 1)
        J = J + 1
        M2 = M2 + Cs
    Loop While Cs <> " "
    M3 = Mid(C, J, Num — J + 1) ' Извлечение информации о баллах,
        ' набранных по третьему предмету, и внесение их в переменную M3
    Sum = Val(M1) + Val(M2) + Val(M3)
    Balls(I) = Sum
    Points(Sum) = Points(Sum) + 1
Next
J = 300
S = Points(J)
While S < K
    J = J — 1
```

```
            S = S + Points(J)
        End While
        If K = S Then
            Console.WriteLine(" Полупроходного балла нет ")
        Else
            For I = 1 To N
                If Balls(I) = J Then Console.WriteLine(Mass(I))
            Next
        End If
        Console.Read()
    End Sub
End Module
```

Сортировка

Сортировкой называется процесс размещения заданного множества объектов в определенном порядке (в частности, если речь идет о числах, то рассматривается сортировка в порядке убывания либо в порядке возрастания).

Существует много различных методов сортировки [5]. При этом простые методы предполагают элементарную программную реализацию, а сложные отличаются более изощренными программными действиями. Далее мы рассмотрим ряд алгоритмов сортировки применительно к массивам чисел.

Сортировка выбором

Сортировка выбором заключается в том, что сначала в неупорядоченном массиве выбирается минимальный элемент. Этот элемент исключается из дальнейшей обработки, а оставшаяся последовательность элементов принимается за исходную. Этот процесс повторяется до тех пор, пока все элементы не будут выбраны. В результате выбранные элементы образуют упорядоченную последовательность.

Для реализации данной идеи удобнее всего использовать дополнительный массив. После первого просмотра найденный минимальный элемент размещается на первом месте в этом дополнительном массиве (рис. 8.1). При втором просмотре мы должны найти следующий элемент в порядке возрастания. Для этого необходимо исключить уже найденный минимальный элемент. Лучший вариант состоит в том, чтобы вместо минимального элемента записать число, заведомо превосходящее все имеющиеся элементы массива. Тогда найденный при втором просмотре элемент

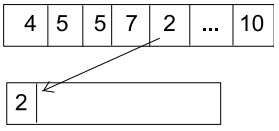

Рис. 8.1. Размещение минимального элемента в дополнительном массиве

размещается на втором месте в дополнительном массиве. Этот процесс аналогичным образом продолжается для третьего, четвертого и последующих элементов.

В листинге 8.10 приведена программа, реализующая рассматриваемый алгоритм сортировки. В начале программы осуществляется заполнение элементов массива с помощью датчика случайных чисел. Далее мы отыскиваем максимальный элемент для замены очередных найденных минимальных значений массива. После этого происходит поиск минимального элемента, который переносится в дополнительный массив. Затем выполняются уже описанные действия по последовательному переносу элементов массива.

Листинг 8.10. Сортировка массива выбором

```
Module Module1
    Sub Main()
        Dim A(10), B(10) As Integer
        Dim N, I, J, Jmin, Min, Max As Integer
        N = 10
        For J = 1 To N
            A(J) = Int(100 * Rnd())
        Next
        Console.WriteLine("A=")
        For i = 1 To N
            Console.WriteLine(A(I))
        Next
        Max = A(1)
        For J = 2 To N
            If A(J) > Max Then
                Max = A(J)
            End If
        Next
        For I = 1 To N
            Min = A(1)
            Jmin = 1
            For J = 2 To N
                If A(J) < Min Then
                    Min = A(J)
                    Jmin = J
                End If
            Next
            B(I) = Min
            A(Jmin) = Max
        Next
        Console.WriteLine("B=")
        For I = 1 To N
            Console.WriteLine(B(I))
        Next
```

```
        Console.Read()
    End Sub
End Module
```

Рассмотренный вариант сортировки массива методом выбора обладает двумя недостатками:

- для его реализации требуется дополнительный массив;
- для нахождения минимального элемента и его индекса на каждом проходе приходится просматривать все элементы массива.

Указанные недостатки устраняются, если все изменения проводить в исходном массиве. Отсортировать его можно следующим образом:

- найти минимальный элемент среди всех элементов массива и поменять его местами с первым элементом;
- найти минимальный элемент среди группы, начинающейся со второго элемента, и поменять его местами со вторым элементом;
- найти минимальный элемент среди группы, начинающейся с третьего элемента, и поменять его местами с третьим элементом и т. д.

На последнем этапе определяется минимальный элемент среди двух последних (по номеру индекса), после чего найденный элемент меняется местами с предпоследним элементом исходного массива. В листинге 8.11 приведена реализация описанного алгоритма.

Листинг 8.11. Сортировка выбором без использования дополнительного массива

```
Module Module1
    Sub Main()
        Dim A(10) As Integer
        Dim N, I, J, Jmin, Min, C As Integer
        N = 10
        For J = 1 To N
            A(J) = 100 * Rnd()
        Next
        Console.WriteLine("A=")
        For I = 1 To N
            Console.WriteLine(A(I))
        Next
        For I = 1 To N — 1
            Min = A(I)
            Jmin = I
            For J = I + 1 To N
                If A(J) < Min Then
                    Min = A(J)
                    Jmin = J
                End If
            Next
```

```
            C = A(I)
            A(I) = Min
            A(Jmin) = C
        Next
        Console.WriteLine("A=")
        For I = 1 To N
            Console.WriteLine(A(I))
        Next
        Console.Read()
    End Sub
End Module
```

Сортировка обменом значений

Сортировка обменом значений заключается в том, что все соседние элементы попарно сравниваются друг с другом и меняются местами, если предшествующий элемент больше последующего. В результате максимальный элемент постепенно смещается вправо. После первого такого просмотра массива максимальный элемент займет крайнее правое положение. Затем процесс просмотра повторяется, и свое место занимает второй по величине элемент, который также исключается из дальнейшего рассмотрения.

Однако описанная процедура оказывается неэффективной, если массив изначально практически отсортирован. В этом случае будут лишние просмотры, что приводит к увеличению времени выполнения программы. Возможен более эффективный вариант, связанный с фиксированием факта обмена. Так, если факта обмена при очередном просмотре массива не фиксируется, то сортировка завершается. В листинге 8.12 приведена программная реализация указанного способа, который называется *пузырьковой сортировкой* (или *сортировкой методом пузырька*). Здесь для фиксирования факта обмена мы создали переменную `Flag`, которая перед началом очередного просмотра сбрасывается в 0. Если в процессе просмотра происходит обмен значений, то переменная `Flag` принимает значение, равное 1. Это является поводом просмотреть массив еще раз.

Листинг 8.12. Сортировка с помощью метода пузырька

```
Module Module1
    Sub Main()
        Dim A(10) As Integer
        Dim N, J, Flag, M, Vr As Integer
        N = 5
        Randomize()
        For J = 1 To N
            A(J) = Int(100 * Rnd())
        Next
```

```
        For I = 1 To N
            Console.Write("A = ")
            Console.WriteLine(A(I))
        Next
        M = N
        Flag = 1
        While Flag = 1
            Flag = 0
            For J = 1 To M — 1
                If A(J) > A(J + 1) Then
                    Vr = A(J)
                    A(J) = A(J + 1)
                    A(J + 1) = Vr
                    Flag = 1
                End If
            Next
            M = M — 1
            For I = 1 To N ' Вывод массива после очередного просмотра
                Console.Write(A(I))
                Console.Write(" ")
            Next
            Console.WriteLine()
        End While
        Console.Read()
    End Sub
End Module
```

Анализ числа учащихся в классах

Рассмотрим задачу, где на вход программе подаются сведения об учениках некоторой средней школы. В первой строке сообщается количество учеников `N`, а каждая из следующих вводимых строк имеет такой формат:

<фамилия> <имя> <класс>,

где:

- *фамилия* — строка, содержащая не более 20 символов;
- *имя* — строка, содержащая не более 15 символов;
- *класс* — год обучения (от 1 до 11) и заглавная буква (от `A` до `Я`) без пробела.

Фамилия и имя, а также имя и класс разделены одним пробелом.

Пример входной строки выглядит так:

```
Иванов Петр 10Б
```

Требуется написать как можно более эффективную программу, которая будет выводить на экран информацию о параллелях (годе обучения) с наименьшим числом

учеников. Программа должна выводить на экран в первой строке количество учеников в искомых параллелях, а во второй строке — в порядке возрастания номера этих параллелей через пробел. Например,

```
30
1 7 11
```

В рассматриваемой задаче информация о фамилиях и именах учащихся нам не требуется, и следовательно, выделять память для хранения этих сведений мы не будем. Основная трудность связана с тем, что из названия класса необходимо выделить только числовую часть (отбросить букву). Начальные шаги аналогичны тем, которые встречались в наших предыдущих примерах. Так, мы считываем символы, составляющие фамилию и имя. После этого наша задача заключается в том, чтобы считать оставшуюся часть строки, которая содержит номер параллели. В результате в соответствующий элемент массива добавляется единица. В листинге 8.13 приведена программная реализация данной задачи.

Листинг 8.13. Извлечение информации о параллелях

```
Module Module1
    Sub Main()
        Dim Mass(11) As Integer
        Dim N, I, J, Num, Min As Integer
        Dim S As String
        Dim C As String
        Dim Cs As Char
        N = Console.ReadLine() ' Считывание числа учащихся
        For J = 1 To 11
            Mass(J) = 0
        Next
        For I = 1 To N         ' Цикл для считывания строк с информацией
            C = Console.ReadLine()
            Num = Len(C) ' Вычисление числа символов во введенной строке
            J = 1        ' J используется в качестве счетчика символов
            Do           ' Извлечение информации о фамилии
                Cs = Mid(C, J, 1)
                J = J + 1
            Loop While Cs <> " "
            Do           ' Извлечение информации об инициалах
                Cs = Mid(C, J, 1)
                J = J + 1
            Loop While Cs <> " "
            S = Mid(C, J, Num — J) ' Извлечение информации о параллели
                                   ' без буквы
            Mass(Val(S)) = Mass(Val(S)) + 1
        Next
        Min = Mass(1)
```

```
        For J = 2 To 11
            If Mass(J) < Min Then Min = Mass(J)
        Next
        Console.WriteLine(Min)
        For J = 1 To 11
            If Mass(J) = Min Then
                Console.Write(J)
                Console.Write(" ")
            End If
        Next
        Console.Read()
    End Sub
End Module
```

Учитывая, что в каждой введенной строке нас интересует только информация о параллели, то можно не считывать начальный фрагмент. Анализ данных можно выполнять с конца строки. А именно возможны две ситуации:

- третий символ с конца строки оказывается пробелом, и тогда номер параллели состоит из одной цифры (это вторая с конца строки);
- в противной ситуации номер параллели состоит из двух цифр, и они занимают третье и второе место начиная с конца строки.

В листинге 8.14 приведена реализация программы извлечения информации о параллелях для такого варианта.

Листинг 8.14. Извлечение информации о параллелях (вариант 2)

```
Module Module1
    Sub Main()
        Dim Mass(11) As Integer
        Dim N, I, J, Num, Min As Integer
        Dim S As String
        Dim C As String
        N = Console.ReadLine()      ' Считывание числа учащихся
        For J = 1 To 11
            Mass(J) = 0
        Next
        For I = 1 To N   ' Цикл для считывания строк с информацией
            C = Console.ReadLine() ' Считывание строки с данными
            Num = Len(C) ' Вычисление числа символов во введенной строке
            S = Mid(C, Num - 2, 1)
            If S = " " Then
                S = Mid(C, Num - 1, 1)
            Else
                S = Mid(C, Num - 2, 2)
            End If
```

```
            Mass(Val(S)) = Mass(Val(S)) + 1
        Next
        Min = Mass(1)
        For J = 2 To 11
            If Mass(J) < Min Then Min = Mass(J)
        Next
        Console.WriteLine(Min)
        For J = 1 To 11
            If Mass(J) = Min Then
                Console.Write(J)
                Console.Write(" ")
            End If
        Next
        Console.Read()
    End Sub
End Module
```

Статистика температуры

В этой задаче на вход программе подаются 365 строк, которые содержат информацию о среднесуточной температуре всех дней года. Формат каждой из строк следующий:

- дата в виде `dd.mm` (на запись номера дня и номера месяца в числовом формате отводится строго два символа, день от месяца отделен точкой);
- значение температуры — число со знаком плюс или минус с точностью до одной цифры после разделителя дробной и целой части (записано через пробел после даты).

Данная информация отсортирована по значению температуры, т. е. хронологический порядок нарушен. Требуется написать как можно более эффективную программу, которая будет выводить на экран информацию о месяцах с минимальной среднемесячной температурой. Найденные минимальные значения следует выводить в отдельной строке для каждого месяца в виде:

- номера месяца;
- значения среднемесячной температуры, округленного до одной цифры после десятичной точки.

В листинге 8.15 приведена программная реализация задачи.

В программе мы задействовали два массива:

- `Mass` — одномерный массив из 12 элементов для накопления суммы температур в течение каждого месяца;
- `Mass2` — одномерный массив из 12 элементов для подсчета количества измерений температуры в течение каждого месяца.

В цикле по строкам данных (365 строк) сначала считывается номер месяца (в соответствии с условием это пятый и шестой символы в строке):

```
S1 = Mid(C, 4, 2)
```

Символьное представление месяца далее мы преобразуем в числовое — `Val(S1)`. После этого из строки считывается значение температуры, что фиксируется в соответствующих элементах массивов `Mass` и `Mass2`.

Таким образом, на данном этапе мы получили суммарное значение температуры в каждом месяце и количество измерений температуры в каждом месяце. Далее необходимо подсчитать среднюю температуру в течение каждого месяца. Для этого организуется цикл от 1 до 12, где последовательно выполняется:

```
Mass(J)=Mass(J)/Mass2(J)
```

На заключительном этапе вычисляется минимальная температура по месяцам и на экран выводятся номер (номера) месяца, когда она была зафиксирована.

Листинг 8.15. Извлечение информации о средней температуре

```
Module Module1
    Sub Main()
        Const N = 365
        Dim Mass(12) As Single
        Dim Mass2(12) As Integer
        Dim J, I, Num As Integer
        Dim Min As Single
        Dim S1, S2 As String
        Dim C As String
        For J = 1 To 12
            Mass(J) = 0
            Mass2(J) = 0
        Next
        For I = 1 To N   ' Цикл для считывания строк с информацией
            C = Console.ReadLine()
            Num = Len(C) ' Вычисление числа символов во введенной строке
            S1 = Mid(C, 4, 2)           ' Считывание номера месяца
            S2 = Mid(C, 7, Num - 7 + 1) ' Считывание температуры
            Mass(Val(S1)) = Mass(Val(S1)) + Val(S2)
            Mass2(Val(S1)) = Mass2(Val(S1)) + 1
        Next
        For J = 1 To 12
            Mass(J) = Mass(J) / Mass2(J)
        Next
        Min = Mass(1)
        For J = 2 To 12
            If Mass(J) < Min Then
                Min = Mass(J)
            End If
        Next
```

```
        For J = 1 To 12
            If Mass(J) = Min Then
                Console.Write("Номер месяца ")
                Console.Write(J)
                Console.Write("Средняя температура ")
                Console.WriteLine(Format(Mass(J),"0.0"))
            End If
        Next
        Console.Read()
    End Sub
End Module
```

Отчет по школам

В программу должна вводиться информация об учащихся из различных школ, которые участвовали в олимпиаде.

Сначала в первой строке вводится количество учащихся, а затем в каждой очередной строке информация по одному из учащихся.

Формат представления данных сведений в очередной строке такой:

- фамилия учащегося (с инициалами, которые располагаются сразу же после фамилии без пробела);
- два целых числа, одно из которых является номером школы, а второе — номером класса (без буквы).

Между фамилией и номером школы, а также между номером школы и номером класса располагается строго по одному пробелу.

Далее следует строка с аналогичной информацией о другом учащемся и т. д. Для определенности: общее количество участников олимпиады не превосходит 500, а номера школ расположены в интервале от 1 до 99.

Программа должна обеспечить вывод на экран информацию о школе/школах, имеющих наибольшее число участников (таких школ может быть несколько).

Формат вывода выглядит так:

- целое число с номером школы;
- целое число, соответствующее числу учащихся этой школы на олимпиаде.

Организация подсчета количества учащихся из различных школ связана с использованием целочисленного массива `Mass` из 99 элементов (по условию номера школ расположены в интервале от 1 до 99). Считывание информации об очередном учащемся приводит к изменению в этом массиве одного из элементов. Например, если очередной учащийся из школы с номером 15, то значение элемента `Mass(15)` увеличивается на 1. В результате после внесения всех строк с учащимися мы получим в массиве `Mass` количество представителей от различных школ.

После этого в программе (листинг 8.16) осуществляется нахождение максимума среди элементов массива `Mass`.

Листинг 8.16. Формирование отчета о школах

```
Module Module1
    Sub Main()
        Dim Mass(99) As Integer
        Dim N, I, J, Max, Num As Integer
        Dim C, M As String
        Dim Cs As Char
        For I = 1 To 99            ' Начальная инициализация массива
            Mass(i) = 0
        Next
        N = Console.ReadLine()     ' Считывание числа учащихся
        For i = 1 To N  ' Цикл для считывания строк с информацией
            C = Console.ReadLine() ' Считывание строки с данными
            Num = Len(C) ' Вычисление числа символов во введенной строке
            J = 1
            Do ' Считывание из входной строки информации о фамилии
                Cs = Mid(C, J, 1)
                J = J + 1
            Loop While Cs <> " "
            M = ""
            Do ' Извлечение информации о номере школы
                Cs = Mid(C, J, 1)
                J = J + 1
                M = M + Cs
            Loop While Cs <> " "
            Mass(Val(M)) = Mass(Val(M)) + 1
        Next
        Max = Mass(1)    ' Поиск максимума в массиве Mass
        For I = 2 To 99
            If Mass(I) > Max Then
                Max = Mass(i)
            End If
        Next
        For I = 1 To 99
            If Mass(I) = Max Then
                Console.Write(I)
                Console.Write(" ")
                Console.WriteLine(Mass(I))
            End If
        Next
        Console.Read()
    End Sub
End Module
```

Отчет о результатах экзамена

Необходимо разработать программу, которая будет после ввода с клавиатуры результатов экзамена формировать статистику по экзамену (какой процент тех или иных оценок). Общее количество учащихся в группе предварительно (перед внесением оценок) вводится с клавиатуры.

Информация о сдаче экзамена вносится с клавиатуры построчно: одна строка содержит фамилию с инициалами (без пробела между ними) и через пробел оценку. Далее в следующих строках с клавиатуры вводятся сведения по следующим учащимся.

Будем считать, что в качестве вариантов оценок рассматриваются такие: 5, 4, 3, 2, 1. Последняя оценка (1) условно означает отсутствие учащегося на экзамене.

Формат вывода статистики такой:

- оценка (целое число);
- процент таких оценок в группе (дробное число).

В листинге 8.17 представлена программа, которая реализует ввод оценок и формирование необходимой статистики.

Организация подсчета оценок связана с использованием целочисленного массива `X` из пяти элементов. Считывание очередной оценки приводит к изменению в этом массиве значения соответствующего элемента. Например, если оценка очередного учащегося 5, то значение элемента `X(5)` увеличивается на 1. В результате после ввода всех строк с успеваемостью мы получим в массиве `X` количество различных оценок по результатам экзамена. Далее в программе выполняется перевод результата в проценты.

Листинг 8.17. Формирование статистики по проведенному экзамену

```
Module Module1
    Sub Main()
        Dim X(5) As Integer
        Dim N, I, Num As Integer
        Dim C, M As String
        For I = 1 To 5             ' Начальная инициализация массива
            X(I) = 0
        Next
        N = Console.ReadLine()     ' Считывание числа учащихся
        For i = 1 To N   ' Цикл для считывания строк с информацией
            C = Console.ReadLine() ' Считывание строки с данными
            Num = Len(C) ' Вычисление числа символов во введенной строке
            M = Mid(C, Num, 1)     ' Выделение из строки оценки
            X(Val(M)) = X(Val(M)) + 1
        Next
```

```
        For I = 1 To 5
            Console.Write(I)
            Console.Write(" ")
            Console.WriteLine(Format(100 * X(I) / N, "0.00"))
        Next
        Console.Read()
    End Sub
End Module
```

Формирование числа из символов

На вход программе подается последовательность латинских букв и цифр. Ввод символов завершается вопросительным знаком. В программе каждый символ вводится в отдельной строке. Программа должна из всех введенных нечетных цифр (полный набор нечетных цифр — 1, 3, 5, 7, 9) сформировать наименьшее число (без повторения цифр). Если нечетных цифр при вводе нет, то программа ничего не выводит.

Для фиксации той или иной нечетной цифры во введенном наборе символов нам потребуется массив из пяти элементов (по числу нечетных цифр) (листинг 8.18). При этом соответствие цифр и индексов элементов массива следующее:

- цифра 1 соответствует элементу с индексом 1;
- 3 — 2;
- 5 — 3;
- 7 — 4;
- 9 — 5.

Соответствие нечетной цифры (`K`) и индекса элемента (`I`) в виде соотношения выглядит так — `K=2*I-1`.

Листинг 8.18. Анализ символов и формирование числа из нечетных цифр

```
Module Module1
    Sub Main()
        Dim A(5) As Integer
        Dim I, K As Integer
        Dim Cs As Char
        For I = 1 To 5 ' Начальная инициализация массива
            A(I) = 0
        Next
        Cs = Console.ReadLine()
        While Cs <> "?"
            K = Asc(Cs) - Asc("0")
            If K > 0 And K < 10 And K Mod 2 = 1 Then
                A(K \ 2 + 1) = 1
            End If
```

```
            Cs = Console.ReadLine()
        End While
        For I = 1 To 5
            If A(I) > 0 Then
                Console.Write(2 * I — 1)
            End If
        Next
        Console.Read()
    End Sub
End Module
```

В формулировке одного из вариантов задания присутствует чуть усложненная формулировка предыдущей разработки. Так, требуется вывести на экран число, в котором каждая цифра из найденного наименьшего числа поделена на 2 и округлена в большую сторону. Если же нечетных цифр нет, то программа должна вывести ноль. Программа для такой постановки задачи приведена в листинге 8.19.

Листинг 8.19. Анализ символов и формирование числа из нечетных цифр

```
Module Module1
    Sub Main()
        Dim A(5) As Integer
        Dim I, K, Flag As Integer
        Dim Cs As Char
        For I = 1 To 5 ' Начальная инициализация массива
            A(I) = 0
        Next
        Cs = Console.ReadLine()
        While Cs <> "?"
            K = Asc(Cs) — Asc("0")
            If K > 0 And K < 10 And K Mod 2 = 1 Then
                A(K \ 2 + 1) = 1
            End If
            Cs = Console.ReadLine()
        End While
        Flag = 0
        For I = 1 To 5
            If A(I) > 0 Then
                Console.Write((2 * I — 1) \ 2 + 1)
                Flag = 1
            End If
        Next
        If Flag = 0 Then Console.Write("Нечетных цифр нет")
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще один вариант экзаменационного билета на данную тему.

На вход программе подается последовательность латинских букв и цифр. Ввод символов завершается вопросительным знаком. В программе каждый символ следует вводить в отдельной строке. Программа должна из всех введенных цифр (не равных нулю) сформировать наибольшее число (без повторения цифр). Если цифр среди введенных символов нет, то программа выводит сообщение об этом.

В данном случае (листинг 8.20) для фиксации той или иной цифры во введенном наборе символов нам потребуется массив из девяти элементов (по числу цифр). При этом соответствие цифр и индексов элементов массива очевидное: цифра соответствует элементу с аналогичным индексом.

Листинг 8.20. Анализ символов и формирование числа

```
Module Module1
    Sub Main()
        Dim A(9) As Integer
        Dim I, K, Flag As Integer
        Dim Cs As Char
        For I = 1 To 9 ' Начальная инициализация массива
            A(I) = 0
        Next
        Cs = Console.ReadLine()
        While Cs <> "?"
            K = Asc(Cs) — Asc("0")
            If K > 0 And K < 10 Then
                A(K) = 1
            End If
            Cs = Console.ReadLine()
        End While
        Flag = 0
        For I = 9 To 1 Step -1
            If A(I) > 0 Then
                Console.Write(I)
                Flag = 1
            End If
        Next
        If Flag = 0 Then Console.Write("Цифр нет")
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще один вариант экзаменационного билета на данную тему. На вход программе подается последовательность латинских букв и цифр. Ввод символов завершается вопросительным знаком. В программе каждый символ следует вводить в отдельной строке. Программа должна из всех введенных цифр (не равных нулю)

сформировать наибольшее число (без повторения цифр). При этом на экран должно выводиться число, равное синусу полученного числа. Если цифр больших нуля в исходной последовательности нет, то программа выводит ноль (листинг 8.21).

Листинг 8.21. Формирование синуса числа из последовательности символов

```
Module Module1
    Sub Main()
        Dim A(9) As Integer
        Dim I, Ves, K As Integer
        Dim Cs As Char
        Dim Sum As Integer
        For I = 1 To 9 ' Начальная инициализация массива
            A(I) = 0
        Next
        Cs = Console.ReadLine()
        While Cs <> "?"
            K = Asc(Cs) — Asc("0")
            If K > 0 And K < 10 Then
                A(K) = 1
            End If
            Cs = Console.ReadLine()
        End While
        Sum = 0
        Ves=1
        For I = 1 To 9
            If A(I) > 0 Then
                Sum = Sum + I * Ves
                Ves = Ves * 10
          End If
        Next
        If Ves > 1 Then
            Console.Write(Math.Sin(Sum))
        Else
            Console.Write(0)
        End If
        Console.Read()
    End Sub
End Module
```

Еще один вариант экзаменационного билета на рассматриваемую тему. На вход программе подается последовательность латинских букв и цифр. Ввод символов завершается восклицательным знаком. В программе каждый символ следует вводить в отдельной строке. Программа должна из всех введенных цифр (не равных нулю) сформировать наименьшее число (без повторения цифр).

При этом на экран выводится число, равное факториалу полученного числа (листинг 8.22). Если цифр больших нуля в исходной последовательности нет, то программа выводит ноль.

Примечание

Учитывая вычисление факториала числа, само число для определенности должно быть не более 19.

Листинг 8.22. Формирование факториала числа из последовательности символов

```
Module Module1
    Sub Main()
        Dim A(9) As Integer
        Dim I, Ves, K As Integer
        Dim Fact As Long
        Dim Cs As Char
        Dim Sum As Integer
        For I = 1 To 9 ' Начальная инициализация массива
            A(I) = 0
        Next
        Cs = Console.ReadLine()
        While Cs <> "!"
            K = Asc(Cs) — Asc("0")
            If K > 0 And K < 10 Then
                A(K) = 1
            End If
            Cs = Console.ReadLine()
        End While
        Sum = 0
        Ves = 1
        For I = 9 To 1 Step -1
            If A(I) > 0 Then
                Sum = Sum + I * Ves
                Ves = Ves * 10
            End If
        Next
        If Ves > 1 Then
            Fact = 1
            For I = 1 To Sum
                Fact = Fact * I
            Next
            Console.Write(Fact)
        Else
            Console.Write(0)
        End If
        Console.Read()
    End Sub
End Module
```

Еще один вариант экзаменационного билета на рассматриваемую тему. На вход программе подается последовательность латинских букв и цифр. Ввод символов завершается восклицательным знаком. В программе каждый символ следует вводить в отдельной строке. Программа должна из всех введенных четных цифр (не равных нулю) сформировать наименьшее число (без повторения цифр).

При этом на экран выводится число, равное факториалу полученного числа, предварительно деленному на 50 и округленному в меньшую сторону. Если цифр больших нуля в исходной последовательности нет, то программа выводит ноль.

Для фиксации той или иной четной цифры во введенном наборе символов нам потребуется массив из четырех элементов (по числу четных цифр). При этом соответствие цифр и индексов элементов массива следующее:

- цифра 2 соответствует элементу с индексом 1;
- 4 — 2;
- 6 — 3;
- 8 — 4.

Соответствие четной цифры (`K`) и индекса элемента (`I`) в виде соотношения выглядит так — `K=2*I`.

Разработка приведена в листинге 8.23.

Листинг 8.23. Формирование факториала числа из последовательности символов

```
Module Module1
    Sub Main()
        Dim A(4) As Integer
        Dim I, Ves, K As Integer
        Dim Fact As Long
        Dim Cs As Char
        Dim Sum As Integer
        For I = 1 To 4 ' Начальная инициализация массива
            A(I) = 0
        Next
        Cs = Console.ReadLine()
        While Cs <> "?"
            K = Asc(Cs) — Asc("0")
            If K > 0 And K < 10 And K Mod 2 = 0 Then
                A(K \ 2) = 1
            End If
            Cs = Console.ReadLine()
        End While
        Ves = 1
        For I = 8 To 2 Step -2
            If A(I \ 2) > 0 Then
                Sum = Sum + I * Ves
```

```
                Ves = Ves * 10
            End If
        Next
        If Ves > 1 Then
            Fact = 1
            For I = 1 To CInt(Sum \ 50)
                Fact = Fact * I
            Next
            Console.Write(Fact)
        Else
            Console.Write(0)
        End If
        Console.Read()
    End Sub
End Module
```

Рассмотрим еще один вариант экзаменационного билета на данную тему.

На вход программе подается последовательность латинских букв и цифр. Ввод символов завершается вопросительным знаком. В программе каждый символ следует вводить в отдельной строке. Программа должна из всех введенных цифр (не равных нулю и встретившихся только один или два раза) сформировать наибольшее число (без повторения цифр). Если цифр среди введенных символов нет, то программа выводит сообщение об этом.

В данном случае для фиксации той или иной цифры во введенном наборе символов нам потребуется массив из девяти элементов (по числу цифр) (листинг 8.24). При этом соответствие цифр и индексов элементов массива очевидное: цифра соответствует элементу с аналогичным индексом 1.

Листинг 8.24. Анализ символов и формирование числа

```
Module Module1
    Sub Main()
        Dim A(9) As Integer
        Dim I, K, Flag As Integer
        Dim Cs As Char
        For I = 1 To 9 ' Начальная инициализация массива
            A(I) = 0
        Next
        Cs = Console.ReadLine()
        While Cs <> "?"
            K = Asc(Cs) - Asc("0")
            If K > 0 And K < 10 Then
                A(K) = A(K) + 1
            End If
            Cs = Console.ReadLine()
        End While
```

```
        Flag = 0
        For I = 9 To 1 Step -1
            If A(I) = 1 Or A(I) = 2 Then
                Console.Write(I)
                Flag = 1
            End If
        Next
        If Flag = 0 Then Console.Write("Цифр нет")
        Console.Read()
    End Sub
End Module
```

Итак, разработки этой главы продемонстрировали уровень наиболее сложных заданий части *C* билетов Единого государственного экзамена. Примерно половина приведенных примеров взята из билетов ЕГЭ прошлых лет. И если вы разобрались с рассмотренными примерами программ, то можно с уверенностью сказать, что к ЕГЭ вы готовы!

В заключение хотим порекомендовать читателям познакомиться с рядом очень хороших книг по разработке алгоритмов и программированию, список которых приводится далее в этой книге.

ПРИЛОЖЕНИЕ

Описание электронного архива

На сайте издательства вы найдете все программы, которые представлены в книге. В исходном архиве имеется два каталога. В каталоге "Примеры" приведены проекты, которые вы можете выполнить в среде Visual Basic 2010. В другом каталоге "Тексты примеров" приведены тексты программ, которые вы можете открыть в текстовом редакторе (например, в Блокноте). Названия проектов (из каталога "Примеры") и названия файлов (из каталога "Тексты примеров") соответствуют названиям листингов в книге. В табл. П1 приводятся пояснения относительно содержания папок, имеющихся в каталогах "Примеры" и "Тексты примеров".

Таблица П1. *Содержание каталогов "Примеры" и "Тексты примеров"*

Название папки	**Описание**
Glava 1	Программы, описанные в *главе 1*
Glava 2	Программы, описанные в *главе 2*
Glava 3	Программы, описанные в *главе 3*
Glava 4	Программы, описанные в *главе 4*
Glava 5	Программы, описанные в *главе 5*
Glava 6	Программы, описанные в *главе 6*
Glava 7	Программы, описанные в *главе 7*
Glava 8	Программы, описанные в *главе 8*

Список используемой литературы

1. Дукин А. Н., Пожидаев А. А. Самоучитель Visual Basic 2010. — СПб.: БХВ-Петербург, 2010. — 560 с.: ил.
2. Зиборов В. Visual Visual Basic 2010 на примерах. — СПб.: БХВ-Петербург, 2010. — 336 с.: ил.
3. Кашаев С. М., Шерстнева Л. В. Самостоятельная подготовка к ЕГЭ по информатике. Необходимая теория и достаточная практика. — СПб.: БХВ-Петербург, 2009. — 464 с.: ил.
4. Златопольский Д. М. Сборник задач по программированию. — СПб.: БХВ-Петербург, 2007. — 240 с.: ил.
5. Златопольский Д. М. Программирование: типовые задачи, алгоритмы, методы. — М.: Бином, 2007. — 233 с.: ил.

Предметный указатель

П

Р

С

Т

Ф

Ц

Ч

Я