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).

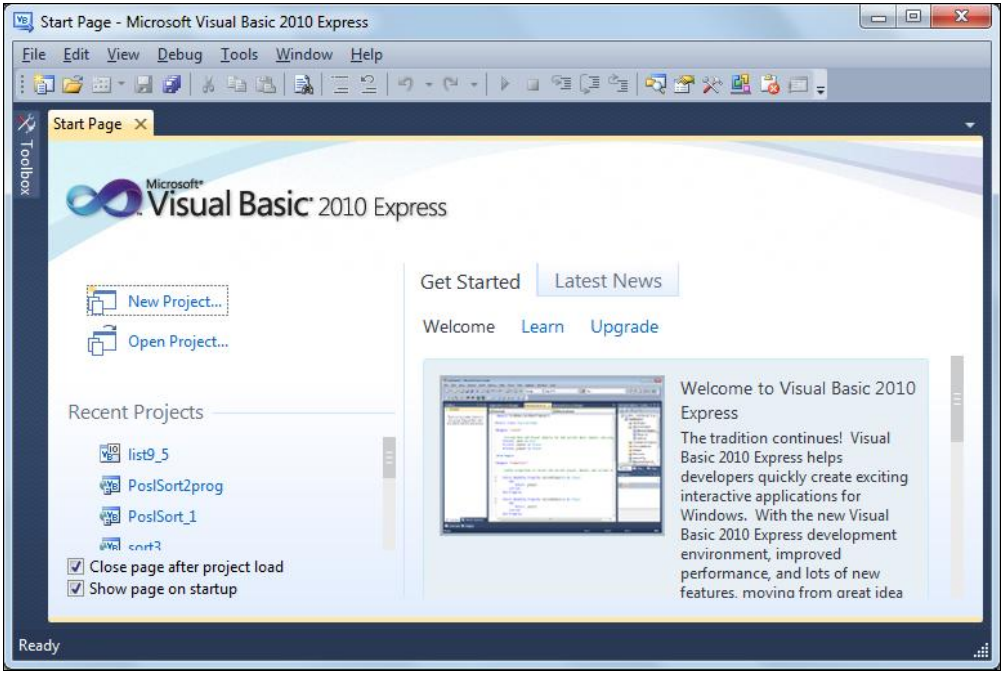

Рис. 1.1. Стартовое окно приложения Visual Basic 2010

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).

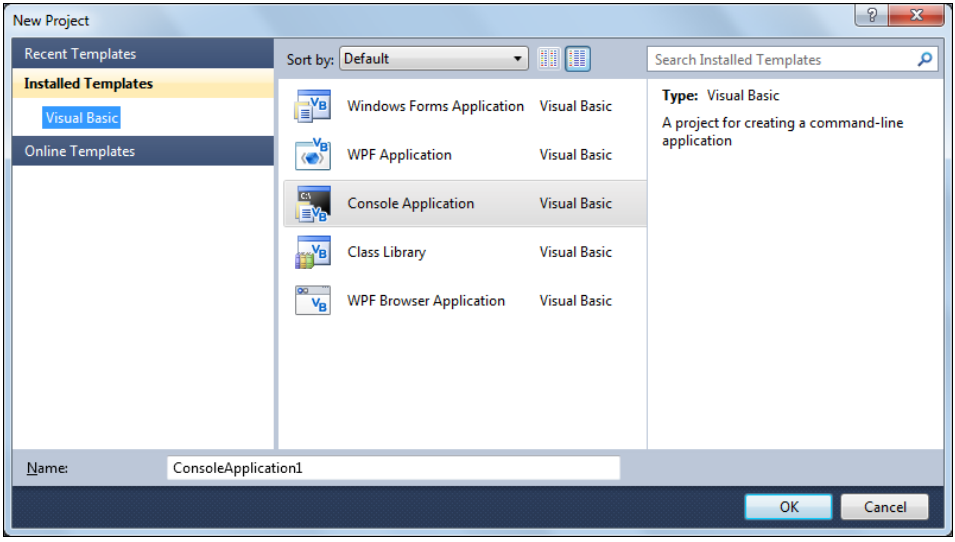

Рис. 1.2. Окно для выбора типа создаваемого проекта

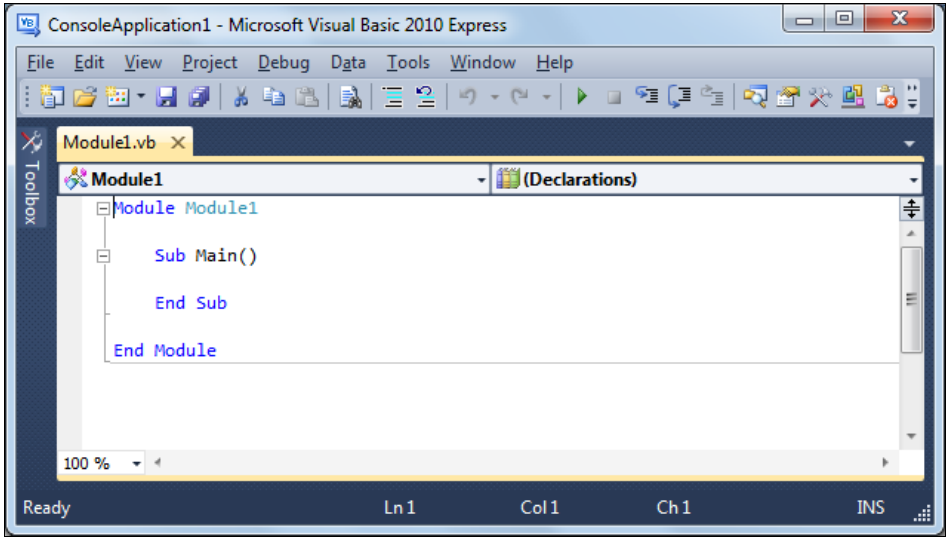

Рис. 1.3. Окно для написания кода программы

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")
```

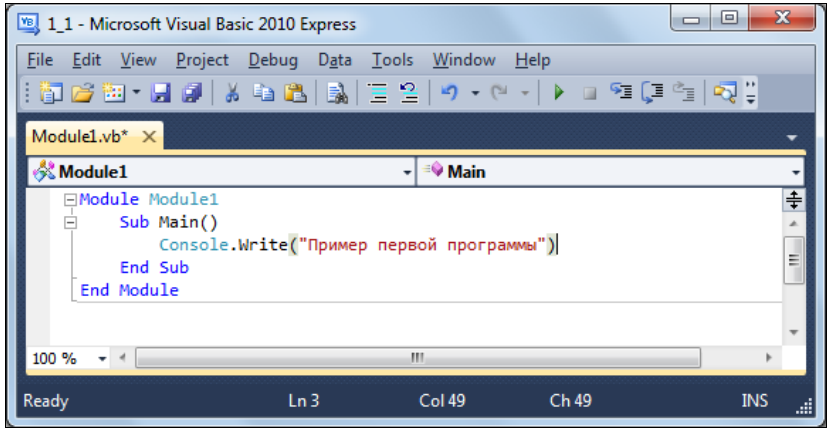

Рис. 1.4. Окно редактора кода с текстом программы вывода сообщения на экран

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).

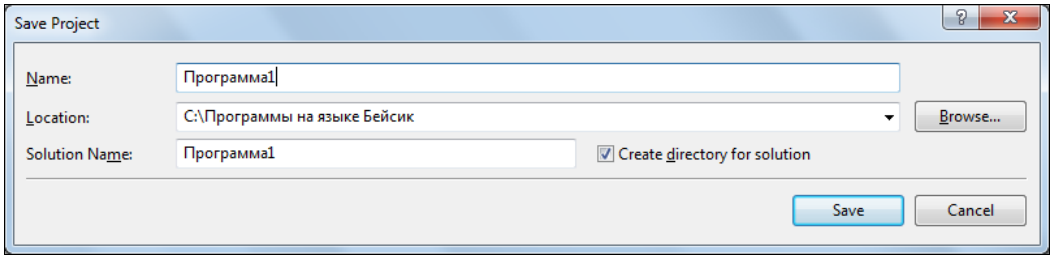

Рис. 1.5. Окно сохранения программного проекта

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.

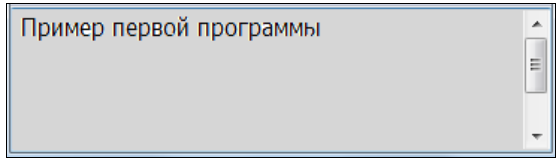

Рис. 1.6. Вывод информации программой, представленной в листинге 1.1

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
```

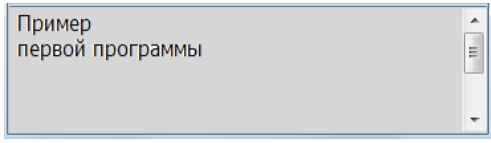

Рис. 1.7. Вывод информации программой, представленной в листинге 1.3

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.

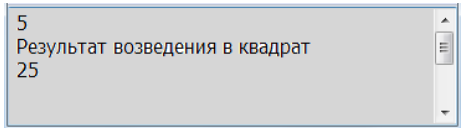

Рис. 1.8. Вывод информации в программе, представленной в листинге 1.4

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
```

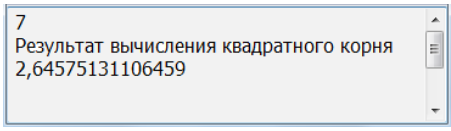

Рис. 1.9. Вывод информации в программе, приведенной в листинге 1.5

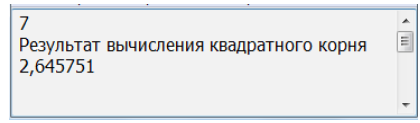

Рис. 1.10. Вывод информации в программе, приведенной в листинге 1.6

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.

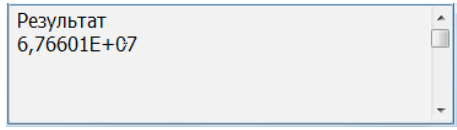

Рис. 1.11. Вывод информации в программе, приведенной в листинге 1.7

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
```

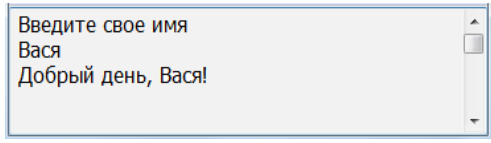

Рис. 1.12. Вывод информации в программе, приведенной в листинге 1.9

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.