Введение

Это методическое пособие предназначено для учителей, которые планируют проводить занятия по курсу «Основы программирования на примере Visual Basic .NET» (.NET читается как «дот нет»). Курс разработан всемирно известной корпорацией Майкрософт в рамках инициативы «Партнерство в образовании»[1] и ориентирован на учащихся 10–11 классов. По соглашению между Майкрософт и Министерством образования и науки Российской Федерации в ряде учебных заведений планируется апробация русской версии Microsoft Visual Basic .NET. Данный курс позволяет научиться разрабатывать проекты с помощью этой системы.

Цель курса состоит в том, чтобы познакомить с основными концепциями программирования учеников старших классов, которые либо обладают начальной подготовкой в области компьютеров, либо вообще не знакомы с ними. На протяжении курса ученики изучают такие базовые приемы программирования, как написание псевдокода, создание форм, объявление переменных, вычисление выражений, использование ветвлений и циклических конструкций и многое, многое другое. При этом они осваивают приемы создания интересных и привлекательных программ (приложений).

Все приведенные в курсе задания по разработке приложений ориентированы на использование Visual Basic. NET. Однако в тексте упоминаются и другие языки программирования .NET (C#, J#) и приводятся примеры кодирования базовых алгоритмических конструкций на этих языках. Это помогает ученикам получить представление о многообразии имеющихся в их распоряжении средств создания программ.

В данном пособии описаны учебные материалы, подготовленные корпорацией Майкрософт к курсу, и даются рекомендации по подготовке учебного класса к занятиям, тематическому планированию курса и проведению занятий. Кроме того, здесь приведены базовые сведения об архитектуре .NET Framework и методологии объектно-ориентированного программирования и даны ссылки на источники дополнительной информации.

Электронная версия методического пособия доступна по адресу www.microsoft.com/rus/education

---

1 Более подробную информацию об этой программе вы можете получить на веб-сайте Майкрософт по адресу: http://www.microsoft.com/Rus/Education/PiL

1. История развития языков программирования

Машинный язык. На заре компьютерной эры, в 40–50-е годы XX века, программы писались на языке машинных кодов (computer language) и представляли собой бинарные (двоичные) инструкции для процессора, т. е. фактически очень длинные последовательности нулей и единиц. Составление и отладка таких программ были чрезвычайно трудоемким делом. Программы на машинных языках были машинно зависимыми, т. е. для каждой ЭВМ необходимо было создавать свою собственную программу, так как в программе в явной форме учитывались аппаратные ресурсы ЭВМ.

Ассемблеры. В начале 50-х годов XX века были созданы языки программирования, получившие общее название ассемблеров (assembly languages), в которых процессору задавались текстовые инструкции. Теперь вместо нулей и единиц программисты могли пользоваться операторами, которые были похожи на слова английского языка (например, команда MOV пересылала данные между регистрами). Для преобразования текста программы на ассемблере в понятный компьютеру машинный код использовался компилятор. Программы на ассемблере были, так же как и машинные коды, машинно зависимыми.

Языки высокого уровня. С середины 50-х годов XX века начали создаваться первые языки программирования высокого уровня (high-level languages). Это были машинно независимые языки программирования, так как они использовали универсальную компьютерную логику и не были привязаны к типу ЭВМ.

Ниже приведены примеры языков программирования высокого уровня, создававшихся и использовавшихся для решения разных задач:

- FORTRAN (расшифровывается как FORmula TRANslator — транслятор формул) — язык, предназначенный для научных и технических расчетов.

- COBOL (Common Business-Oriented Language — стандартный язык для делового применения) — язык, в основном предназначавшийся для коммерческих приложений, обрабатывавших большие объемы нечисловых данных.

- LISP (List Processing — обработка списков) и PROLOG — языки, созданные для исследований в области искусственного интеллекта.

- BASIC (Beginner's All-Purpose Symbolic Instruction Code — универсальный язык символьных инструкций для начинающих) — язык, отличающийся простотой создания программ.

- Pascal (назван его создателем Виртом в честь великого французского математика, физика, литератора и философа Блеза Паскаля) — создан в 1970 году как язык для обучения программированию. По мнению Вирта, Pascal должен был способствовать соблюдению правильной дисциплины программирования, поэтому, наряду со строгой типизацией, в нем сведены к минимуму возможные синтаксические неоднозначности, а сам синтаксис интуитивно понятен даже при первом знакомстве с языком.

- C (произносится «Си») — язык, позволяющий создавать быстро и эффективно выполняющийся программный код.

Языки объектно-ориентированного программирования. В 90-х годах XX века начали создаваться объектно-ориентированные языки программирования (следующий этап развития языков программирования высокого уровня), позволяющие визуально конструировать графический интерфейс приложений:

- Visual Basic .NET — среда разработки, созданная корпорацией Microsoft для создания приложений с графическим интерфейсом в среде операционной системы Windows на основе платформы .NET.

- C++ и C# — языки объектно-ориентированного программирования, созданные на базе языка C и использующие все его преимущества.

- Object Pascal — язык, созданный компанией Borland для разработки приложений с графическим интерфейсом в среде Delphi для операционной системы Windows.

На рис. 1 представлена упрощенная схема развития языков программирования.

История развития языков программирования[1]

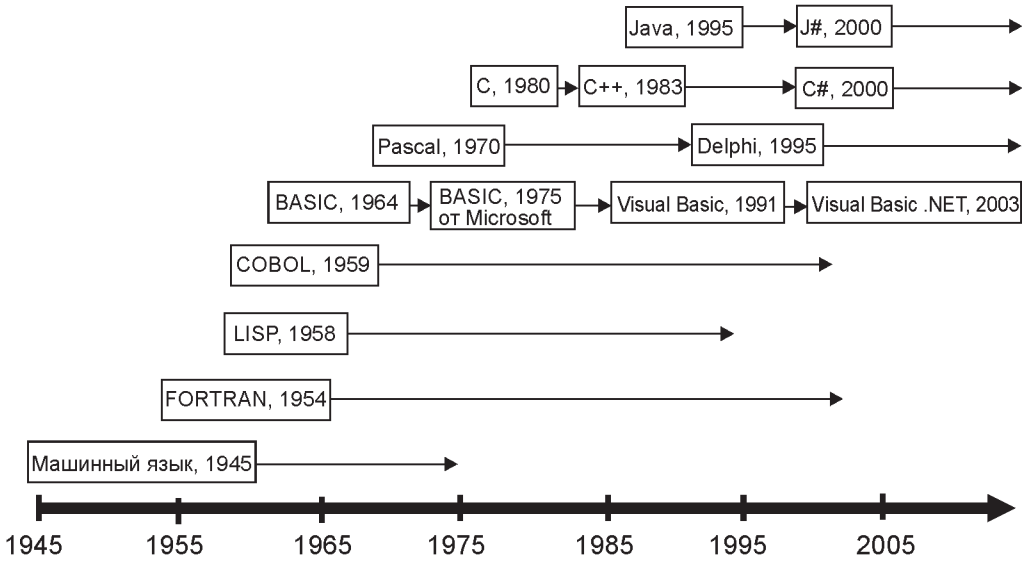


Рис. 1

2. Введение в объектно-ориентированное программирование

В данном пособии рассматривается учебный курс с использованием системы программирования Visual Basic .NET, которая базируется на методологии под названием «объектно-ориентированное программирование» (ООП).

В основе ООП лежит понятие *объекта*, объединяющего в себе *свойства* объекта и *методы* объекта (действия объектов и над объектами). Такое объединение свойств и методов в объекте называется *инкапсуляцией*. Она позволяет при создании программы скрыть особенности реализации того

---

1 Составлена на базе источника: http://www.oreilly.com/pub/a/oreilly/news/languageposter_0504.html

или иного способа обработки информации внутри объекта и облегчает повторное использование ранее написанного кода.

Для описания разных видов программных объектов служат *классы*. Класс определяет набор свойств и методов, являясь своего рода шаблоном, на основе которого создаются объекты. Соответственно, любой объект является *экземпляром* некоторого класса. Разработка объектно-ориентированной программы сводится, в первую очередь, к созданию набора классов, обладающих необходимыми программисту свойствами и методами.

Новые классы могут создаваться на основе уже существующих (предков), при этом они (потомки) *наследуют* свойства и методы последних. Создавая классы-потомки, разработчик может добавить им новые свойства и методы, а может переопределить методы, унаследованные от класса-предка. Возможность менять поведение программных объектов при вызове одноименного метода называется *полиморфизмом*. Инкапсуляция, наследование и полиморфизм — три базовых принципа, лежащих в основе ООП.

После того как необходимые классы определены, разработчику программы остается создать на их основе нужные объекты и организовать взаимодействие между ними путем обмена сообщениями и реакции на *события*. В этом взаимодействии объектов, собственно, и заключается работа объектно-ориентированной программы.

В приложении 1 даны определения базовых понятий объектно-ориентированного программирования.

Если вы не знакомы с языком Visual Basic .NET, рекомендуем также изучить справочник по нему, приведенный в приложении 2.

3. Краткий обзор .NET Framework и Visual Studio .NET

Система объектно-ориентированного программирования Visual Basic .NET является составной частью единой среды разработки приложений Visual Studio .NET. Последняя, в свою очередь, базируется на разработанной корпорацией Майкрософт платформе .NET Framework и представляет собой универсальный инструмент, с помощью которого можно создавать са-

мые разнообразные приложения, начиная от программ командной строки и заканчивая веб-службами XML. Такие приложения могут создаваться с использованием разных языков программирования, из которых Visual Basic .NET является, по-видимому, наиболее простым для восприятия и освоения.

Чтобы немного разобраться в сущности разработанных Майкрософт технологий, кратко рассмотрим архитектуру .NET Framework, а также место Visual Studio .NET и Visual Basic .NET в этой архитектуре.

Основным элементом .NET Framework является общеязыковая среда выполнения приложений. Эту среду (по-английски она называется Common Language Runtime, или CLR) можно считать неким агентом, который выполняет следующие функции:

- обеспечивает компиляцию кода по мере вызова тех или иных компонентов программы;
- распределяет память для кэширования откомпилированного кода и размещения данных;
- управляет потоками вычислений и удаленным взаимодействием программ;
- обеспечивает строгую проверку типов данных и другие виды проверки точности кода, что гарантирует безопасность и надежность выполнения программ.

Таким образом, основным принципом работы CLR является управление программным кодом. Именно поэтому код, который выполняется в .NET Framework, называют *управляемым кодом* (managed code), а код, который выполняется на компьютере, минуя CLR, — *неуправляемым* (unmanaged). Более того, данные, с которыми работает управляемый код, также находятся под полным контролем CLR и поэтому называются *управляемыми данными* (managed data).

Другой основной компонент .NET Framework — общая для всех языков программирования библиотека классов. Ее наличие позволяет разработчикам использовать единую систему типов данных и вызываемых функций (точнее, программных объектов, их свойств и методов[1]). Соответственно, большая часть функциональности программы, которая ранее реализовывалась за счет функций и процедур конкретного языка программирования, теперь обеспечивается использованием библиотеки

---

[1] Классы, объекты, свойства и методы более подробно обсуждаются далее в разделе, посвященном основам объектно-ориентированного программирования.

классов. Например, чтобы вычислить квадратный корень в предыдущих версиях Visual Basic, программисту нужно было воспользоваться конструкцией вида:

```
X=SQR(Y)
```

В Visual Basic .NET аналогичный оператор будет выглядеть иначе:

```
X=System.Math.Sqrt(Y)
```

В этой конструкции уже будет задействован один из стандартных классов библиотеки .NET Framework, относящийся к так называемому *пространству имен* [1] (namespace) System.Math.

Существенным преимуществом данного подхода является возможность использовать одни и те же классы в программах, написанных на разных языках программирования: и на тех, которые разработаны корпорацией Майкрософт, и на языках сторонних производителей. Более того, разработчики могут создавать свои собственные библиотеки классов и использовать их в дальнейшей работе. А межъязыковое взаимодействие и общая среда разработки позволяют, например, в Visual Basic .NET-приложениях использовать компоненты, написанные на других языках .NET Framework.

Если изобразить основные компоненты, обеспечивающие разработку и выполнение программ на компьютере с операционной системой Windows, то получится схема, приведенная на рис. 2.

Как видно из схемы, Visual Studio .NET — это единая среда разработки приложений, как традиционных, так и работающих в среде выполнения .NET Framework. И если первые весьма жестко привязаны к особенностям интерфейса прикладного программирования в операционной системе Windows (Win32 API), то для вторых CLR «экранирует» эти особенности. Такой подход делает написанные для общеязыковой среды программы легко переносимыми на компьютеры, работающие не под управлением Windows [2].

---

1 В .NET Framework пространством имен называется некоторая область, в которой определены названия, свойства и методы используемых классов (как системных, так и создаваемых разработчиком программы).

2 Примеры такого переноса уже известны, например, проект Portable .NET для Linux и других Unix-подобных систем или проект Rotor для FreeBSD и Mac OS X.

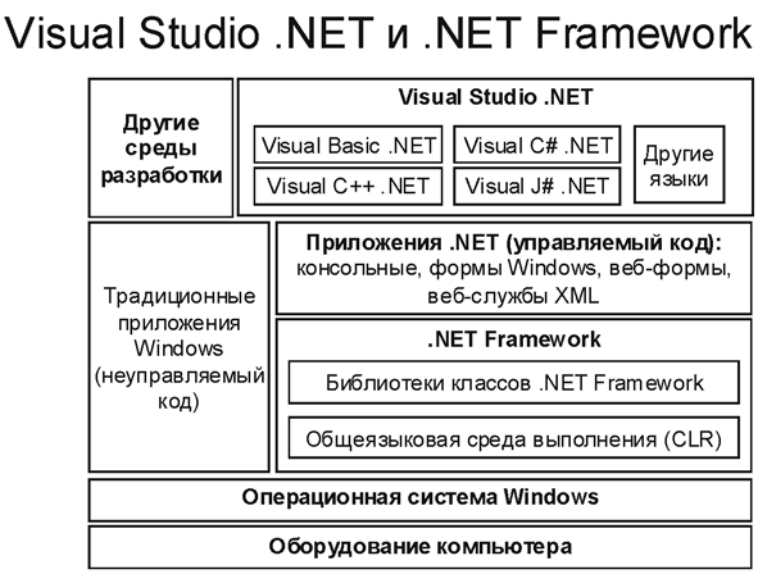


Рис. 2

Совместное использование Visual Studio .NET и .NET Framework предоставляет в распоряжение разработчиков один из самых мощных на сегодняшний день инструментов для создания приложений. В то же время этот инструмент является весьма простым в освоении, что дает возможность применять его в курсах школьной учебной программы.

В настоящее время корпорация Майкрософт выпускает два разных варианта Visual Studio для платформы .NET [1]:

- Visual Studio .NET 2003 (в различных редакциях) и отдельные компоненты этой среды разработки;
- Visual Studio 2005 (опять же в различных редакциях) и ее отдельные компоненты.

Некоторые компоненты Visual Studio 2005 в варианте Express Edition (например, Visual Basic 2005 Express Edition) сейчас распространяются корпорацией Майкрософт бесплатно; их дистрибутивы доступны для «скачивания» с сайта корпорации [2].

---

[1] Более подробные сведения приведены на русскоязычном веб-сайте Майкрософт: http://www.microsoft.com/Rus/Msdn/vs

[2] См. http://msdn.microsoft.com/vstudio/express

Следует отметить, что в этих вариантах (2005 — наиболее современный) различаются как среда разработки, так и среда выполнения приложений (версия 2003 базируется на .NET Framework 1.1, а версия 2005 — на .NET Framework 2.0). Различия необходимо учитывать и в процессе разработки программ с помощью данных систем, и при использовании программ в дальнейшем.

В состав Visual Studio .NET входят следующие языки программирования:

- Visual Basic .NET;
- C# (произносится Си-шарп);
- J# (произносится Джей-шарп);
- C++ (произносится Си плюс плюс).

Кроме того, в эту систему включена электронная справочная система (так называемая библиотека MSDN). Русскоязычная справка размещена в Интернете по адресу http://msdn.microsoft.com/library/rus/

4. Учебные материалы

В комплект подготовленных к курсу «Основы программирования на примере Visual Basic .NET» учебных материалов входят:

- методическое пособие для учителей, которое вы сейчас читаете;
- учебное пособие по языку объектно-ориентированного программирования Visual Basic .NET для учащихся;
- прилагаемый к данному пособию компакт-диск (в тексте он обозначается как **Microsoft-CD**).

Для того чтобы подготовиться к проведению занятий по данному курсу, необходимо заранее и тщательно ознакомиться с содержимым этих материалов. Рассмотрим подробнее, как устроено учебное пособие и что находится на компакт-диске.

Учебное пособие

Учебное пособие включает 12 глав, каждая из которых предваряется вступлением, напрямую не относящимся к обсуждаемому материалу. В этих вступлениях кратко раскрывается история развития Microsoft через создание различных версий операционных систем и языка программирования Basic, что позволяет параллельно с изучением языка программирования совершить «путешествие во времени».

В тексте пособия используются следующие соглашения, касающиеся шрифтового оформления и выделения важной информации.

- *Курсивом* выделены важные понятия и термины, а также названия диалоговых окон, пунктов меню и управляющих элементов (текстовых полей, кнопок и т. д.) графического интерфейса.
- Шрифтом `Courier` выделены тексты программ на языке программирования Visual Basic.
- Важная информация и формулы выделены в тексте восклицательным знаком:

Важная информация

- Материалы, содержащие дополнительную интересную информацию, выделены значком .

В конце каждой главы приведен тест, позволяющий оценить степень усвоения материала учениками. На каждый вопрос правильный ответ единственный, его нужно выбрать из четырех предложенных вариантов.

Приведем краткий обзор содержания учебного пособия.

Глава 1 «Программы в повседневной жизни» посвящена обсуждению понятий *«программа»*, *«программист»*, *«программирование»*. Здесь же даются базовые сведения о разных языках программирования и о синтаксисе языка программирования.

В **главе 2 «Система программирования Visual Basic .NET»** рассматриваются компоненты интегрированной среды разработки Visual Studio .NET и описываются основные приемы создания приложений в этой среде. В ходе выполнения практических заданий в данной главе учащиеся

разрабатывают свою первую программу с графическим интерфейсом и проверяют ее работу.

В **главе 3 «Алгоритмы и программы»** обсуждаются элементы, из которых строятся программы вне зависимости от используемого для их написания языка программирования. Здесь же вводится понятие *«алгоритм»* и даются рекомендации по описанию алгоритма работы программы в виде так называемого псевдокода.

В **главе 4 «Формы и элементы управления»** рассматриваются *формы* — основа графического интерфейса приложения Windows (óкна приложения) и *элементы управления*, которые помещаются на форму и обеспечивают взаимодействие между приложением и пользователем. В этой главе также говорится о *событиях*, с помощью которых программа «узнает» о действиях работающего с ней пользователя.

Глава 5 «Свойства и методы» посвящена обсуждению понятий *«свойство»* и *«метод»*. С помощью свойств разработчик может определять те или иные характеристики программных объектов (например, форм, элементов управления и т. п.), а методы позволяют ему управлять действиями этих объектов при работе программы. Здесь же описывается, как в коде программы использовать свойства и *вызывать* методы.

В **главе 6 «Присваивание и переменные»** описывается оператор *присваивания*, с помощью которого в программах, написанных на самых разных языках, чаще всего задаются значения свойств объектов и значения *переменных*. Последние используются для хранения данных практически в любой программе, поэтому в главе также описываются основные правила объявления переменных и работы с ними в языке Visual Basic .NET.

Глава 7 «Операции» посвящена обсуждению способов обработки информации в ходе работы программы с помощью арифметических, строковых и логических *операций*. Здесь также вводится понятие *отладки* программы и описываются основные приемы отладки в среде Visual Studio .NET.

В **главе 8 «Ветвление: неполная форма»** и в **главе 9 «Ветвление: полная форма»** даются базовые понятия *булевой* (двоичной) *логики* и описываются *операции сравнения*, позволяющие в ходе работы программы осуществить проверку истинности условий. Далее в этих главах рассматриваются сначала простые, а затем более сложные алгоритмические конструкции, с помощью которых порядок работы программы можно изменять в зависимости от истинности одного или нескольких условий.

Глава 10 «Циклы со счетчиком» начинает обсуждение вопросов, связанных с организацией *циклов*, т. е. алгоритмических конструкций,

обеспечивающих при работе программы многократное выполнение определенной последовательности операторов. В этой главе рассматриваются простые и вложенные циклы со счетчиком, которые используются, когда число повторений заранее известно.

Глава 11 «Циклы с условием» посвящена более сложным циклическим конструкциям, в которых число повторений заранее неизвестно, а в качестве критерия прекращения работы цикла выступает истинность или ложность некоторого условия. Рассматриваются циклы, в которых проверка истинности этого условия осуществляется до выполнения операторов внутри них, и циклы, в которых эта проверка проводится после выполнения операторов.

В **главе 12 «Подпрограммы и функции»** обсуждаются приемы разбиения общего алгоритма работы программы на части и реализации отдельных частей в виде подпрограмм и функций. Описываются и способы создания подпрограмм и функций, и способы работы с ними.

Компакт-диск

На компакт-диске, который прилагается к данному методическому пособию, находится следующая информация.

- *Учебное пособие в формате PDF*. Раскрыв папку Учебное пособие, вы увидите вложенные папки с материалами учебного пособия и сможете ознакомиться с ними. Каждая глава учебного пособия заканчивается интерактивным тестом. Ученики могут использовать этот тест для быстрой оценки своих знаний.
- *Файлы с готовыми проектами в системе Visual Basic .NET 2003 и с заготовками таких проектов*. Все эти файлы находятся в папке PRACTICUM. Для использования в ходе занятий их необходимо переписать на жесткий диск. Эта операция будет описана ниже в разделе, посвященном подготовке компьютеров в классе.
- *Дистрибутивы программного обеспечения, необходимого для установки системы объектно-ориентированного программирования Visual Basic 2005 Express Edition и электронной документации к ней, а также дополнительных программ, которые могут оказаться полезными учителю*. Файлы дистрибутивов находятся в папке PROGRAM. Там же содержится *описание используемого в курсе программного обеспечения и краткие инструкции по его установке на компьютеры в классе*.

- *Справочные таблицы по языку Visual Basic .NET*. расположены в папке PLACAT.

5. Подготовка учебного класса к занятиям

Для успешного проведения занятий по данному курсу особенно важно уделить особое внимание подготовке учебного класса. В первую очередь, это касается технического оснащения класса компьютерами и установки на эти компьютеры всего необходимого программного обеспечения. Кроме того, все установленное на компьютеры программное обеспечение необходимо должным образом настроить и проверить в работе, чтобы избежать возникновения проблем в дальнейшем. Эти процедуры подготовки учебных компьютеров к занятиям рассмотрены далее.

Поскольку все примеры в учебном пособии[1] базируются на использовании локализованной русской версии системы Visual Basic .NET 2003, основное внимание ниже будет уделено именно ей. Эту версию корпорация Майкрософт предлагает для средних учебных заведений по специальным низким ценам и с расширенными условиями применения[2]. Кроме того, облегченная версия Visual Basic 2005 Express Edition по разрешению корпорации размещена на компакт-диске **Microsoft-CD**. Поэтому установка и использование этой версии также будут кратко рассмотрены.

Системные требования

Для проведения занятий необходимо, чтобы компьютеры в классе удовлетворяли определенным требованиям с точки зрения оборудования и установленной на них операционной системы. Минимальные требования приведены в следующей таблице (в скобках указаны рекомендованные значения):

---

1 И соответственные файлы на компакт-диске **Microsoft-CD**.

2 См. http://www.microsoft.com/Rus/Licensing/Volume/Academic/PilSa.mspx

Ресурс	Visual Basic .NET 2003	Visual Basic 2005 Express Edition
Процессор	Pentium II, 450 МГц (Pentium III, 600 МГц)	Pentium, 600 МГц (Pentium, 1 ГГц)
Оперативная память, Мб	Windows Server 2003 — 160, Windows XP Professional — 160, Windows XP Home Edition — 96, Windows 2000 Professional — 96, Windows 2000 Server — 192	192 (256)
Свободное пространство на жестком диске	500 Мб свободного пространства на системном диске, 1,5 Гб свободного пространства на диске, куда выполняется установка. Дополнительные 1,9 Гб свободного пространства для необязательной документации библиотеки MSDN	До 1,3 Гб доступного дискового пространства
Операционная система (ОС)	Microsoft Windows Server 2003, Windows XP Professional, Windows XP Home Edition, Windows 2000 Professional (требуется пакет обновления SP3 или более поздний), Windows 2000 Server (требуется пакет обновления SP3 или более поздний)	Microsoft Windows 2003 Server (требуется пакет обновления SP1), Windows XP (требуется пакет обновления SP2), Windows 2000 (требуется пакет обновления SP4)
Дисковод	Устройство для чтения компакт-дисков или DVD-дисков	Устройство для чтения компакт-дисков или DVD-дисков не требуется
Монитор	1024 × 768, 256 цветов	800 × 600, 256 цветов (1024 × 768, 65 536 цветов)
Мышь	Мышь Microsoft или совместимое указывающее устройство	Мышь Microsoft или совместимое указывающее устройство

Подчеркнем, что в таблице приведены минимальные требования к оборудованию. Использование более мощных процессоров и/или оперативной памяти большего объема позволит работать на компьютерах в классе более комфортно и продуктивно.

Установка операционной системы Windows XP Professional

Как видно из предыдущей таблицы, Visual Basic .NET 2003 или Visual Basic 2005 Express Edition можно установить на компьютеры с разными операционными системами, выпускаемыми корпорацией Майкрософт. Для определенности мы будем полагать, что в учебном классе будет использоваться локализованная русская версия операционной системы Windows XP Professional. Если эта операционная система еще не установлена в вашем компьютерном классе (или по каким-то причинам требуется выполнить ее переустановку), вы можете обратиться к описанию процедуры ее установки с загружаемого дистрибутивного компакт-диска[1], приведенной в приложении 3.

Дополнительные настройки компьютера

Для того чтобы избежать проблем при проведении занятий в классе, рекомендуется правильно настроить защиту файлов и папок, находящихся на жестком диске, и настроить квотирование дискового пространства. Последнее позволит избежать переполнения диска файлами учеников. Кроме того, необходимо переписать с компакт-диска **Microsoft-CD** файлы примеров, которые могут быть использованы в ходе занятий.

Установка разрешений на корневую папку диска C:

1. Откройте окно командной строки, для чего щелкните по кнопке *Пуск*, затем выберите *Выполнить*, в поле *Открыть* наберите *cmd* и нажмите клавишу *Enter*.
2. После этого выполните команду

```
cacls c:\ /e /r Все СОЗДАТЕЛЬ-ВЛАДЕЛЕЦ /p Пользователи:R
```

[1] Для облегчения развертывания системы мы рекомендуем воспользоваться дистрибутивом для многократной установки (Volume License Media), в который уже интегрирован пакет обновлений SP2.

С ее помощью из списка разрешений на доступ к корневой папке диска C: будут удалены записи для специальных групп *Все* и *СОЗДАТЕЛЬ-ВЛАДЕЛЕЦ*, а разрешения для обычных пользователей модифицированы так, чтобы они могли только читать данные из этой папки. Вывод сообщения «обработан каталог c:\» свидетельствует об успешном выполнении команды.

Примечание. Работая в режиме командной строки, набирайте команды в одну строку и следите за тем, чтобы ключи и параметры команд были отделены друг от друга хотя бы одним пробелом.

Настройка квотирования дискового пространства

1. Запустите программу Проводник (*Пуск-Все программы-Стандартные-Проводник*.).
2. В левой части окна программы щелкните по значку *Мой компьютер*, затем щелкните правой кнопкой по значку *Локальный диск (C:)* и выберите *Свойства*.
3. В диалоговом окне *Свойства: Локальный диск (C:)* перейдите на вкладку *Квота*.
4. Установите флажки *Включить управление квотами* и *Не выделять место на диске при превышении квоты*, после этого установите переключатель в положение *Выделять на диске не более* и задайте в поле справа от него объем дискового пространства, который будет доступен каждому ученику, например 20 Мб [1] .
5. В поле *Порог выдачи предупреждений* введите немного меньшее значение, например 15 Мб.
6. Щелкните по кнопке *Применить* и подтвердите включение системы квот, щелкнув по кнопке *OK*.
7. Чтобы квотирование дискового пространства не затрагивало пользователя *Администратор*, выберите *Записи квот*, затем в окне записей квот дважды щелкните по пункту *Администратор*.

   В окне *Параметры квоты для...* установите переключатель в положение *Не ограничивать выделение места на диске* и щелкните по кнопке *OK*.

---

[1] Выберите это значение с учетом емкости установленных на компьютерах в классе жестких дисков и предполагаемого числа учеников, которые будут заниматься на этих компьютерах.

8. Закройте окно с записями квот, после чего щелкните по кнопке *OK* в окне свойств диска C:.
9. Не закрывайте окно программы *Проводник*.

Копирование файлов с примерами

1. Вставьте компакт-диск **Microsoft-CD** в дисковод.
2. Перетащите папку PRACTICUM в корневую папку диска C:.
3. Дождитесь завершения операции копирования, после чего с помощью программы ATTRIB снимите установленные для скопированных файлов и папок атрибуты *Только для чтения* и *Скрытый*. Для этого в ранее открытом окне командной строки подайте команду:

   ```
   attrib -r -h c:\practicum\* /s
   ```

   Примечание. Даже после этого скопированные администратором файлы примеров будут доступны обычным пользователям компьютера только для чтения. Поэтому покажите ученикам, как копировать эти файлы из папки PRACTICUM в папку, на которую у них есть разрешение на запись.

4. Закройте все открытые на экране окна программ.

Установка Visual Basic .NET 2003

Дистрибутив локализованной русской версии Visual Basic .NET 2003 [1] поставляется на пяти компакт-дисках:

- собственно Visual Basic .NET 2003 Standard,
- диск с необходимым для установки дополнительным программным обеспечением (Visual Basic .NET 2003 Prerequisites),
- три диска с переведенной на русский язык справочной документацией (библиотека MSDN).

---

[1] Эту версию компания Майкрософт предлагает для учебных заведений по специальным низким ценам и с расширенными условиями по использованию — специальная программа лицензирования для средних учебных заведений: http://www.microsoft.com/Rus/Licensing/Volume/Academic/PilSa.mspx

Чтобы установить Visual Basic .NET 2003 на компьютере с операционной системой Windows XP Professional, проделайте следующее:

1. Вставьте первый из перечисленных выше компакт-дисков в дисковод. При этом должна автоматически запуститься программа установки. Если этого по каким-то причинам не произойдет, запустите программу SETUP.EXE из корневой папки компакт-диска.
2. В окне *Visual Basic .NET — Установка* выберите *Подготовка системы для Visual Studio .NET* (шаг 1).
3. Поскольку школьный курс не предполагает создания веб-приложений, щелкните по кнопке *Пропустить* в окне *Требования для веб-проектов*. На экране появится диалоговое окно *Вставка диска*.
4. Вставьте в дисковод следующий из упомянутых выше компакт-дисков и щелкните по кнопке *OK*.
5. На странице *Лицензионное соглашение* установите переключатель в положение *Принимаю*, затем щелкните по кнопке *Продолжить*.
6. На следующей странице со списком необходимых для Visual Studio .NET компонентов выберите *Установить сейчас* и дождитесь, когда эти компоненты будут полностью установлены. Затем щелкните по кнопке *Готово*.
7. Во вновь появившемся окне *Visual Basic .NET — Установка* выберите *Visual Basic .NET* (шаг 2). Затем вставьте в дисковод первый компакт-диск и щелкните по кнопке *OK* в окне *Вставка диска*.
8. На странице *Начало* установите переключатель в положение *Принимаю*, введите в поле *Ключ продукта* 25-символьный ключ, после чего щелкните по кнопке *Продолжить*.
9. На странице *Параметры* оставьте все установленные по умолчанию флажки и щелкните по кнопке *Установить*. Начнется процесс установки всех компонентов Visual Basic .NET 2003, который может занять довольно продолжительное время. После завершения установки щелкните по кнопке *Готово*.
10. Во вновь появившемся окне *Visual Basic .NET — Установка* выберите *Документация продукта* (шаг 3).
11. Вставьте в дисковод первый компакт-диск библиотеки MSDN и щелкните по кнопке *OK* в окне *Вставка диска*. Запустится мастер установки библиотеки MSDN на русском языке.

12. На странице *Добро пожаловать* щелкните по кнопке *Далее*.
13. На странице *Лицензионное соглашение* установите переключатель в положение *Я принимаю условия лицензионного соглашения*, затем щелкните по кнопке *Далее*.
14. На странице *Сведения о пользователе* щелкните по кнопке *Далее*.
15. На странице *Выберите вариант установки* установите переключатель в положение *Обычный*, после чего щелкните по кнопке *Далее*.
16. На странице *Конечная папка* оставьте выбранную по умолчанию папку для установки и щелкните по кнопке *Далее*.
17. На странице *Все готово для установки программы* щелкните по кнопке *Установить*, чтобы начать установку библиотеки MSDN. Заменяйте компакт-диски в дисководе по мере необходимости.
18. Когда установка будет завершена, щелкните по кнопке *Готово*.
19. На экране вновь появится окно *Visual Basic .NET — Установка*. Если в вашем дистрибутиве есть дополнительный компакт-диск с исправлениями для Visual Studio .NET 2003, щелкните по кнопке *Наборы исправлений* (шаг 4), а затем — по пункту *Установка набора исправлений с диска* и следуйте указаниям программы.
20. Если диска с исправлениями нет, завершите процедуру установки, щелкнув по кнопке *Выход*.

Проверка работы Visual Basic .NET 2003

Чтобы проверить работу Visual Basic .NET 2003, попробуйте создать в этой системе программу и запустить ее. Поскольку разработка Windows-приложений подробно обсуждается в учебном пособии, попробуйте для разнообразия создать консольное приложение (или программу командной строки). Для этого:

1. Запустите Visual Studio .NET 2003 (*Пуск-Все программы-Microsoft Visual Studio .NET 2003-Microsoft Visual Studio .NET 2003*), на начальной странице переключитесь на вкладку *Проекты* и выберите *Создать проект*.
2. В диалоговом окне *Создать проект* в поле *Шаблоны* выделите *Консольное приложение*, после чего введите текст «HelloWorld» в поле *Имя* (рис. 3). Обратите внимание, в какой папке по умолчанию создаются проекты в системе Visual Studio .NET 2003 (поле *Расположение*).

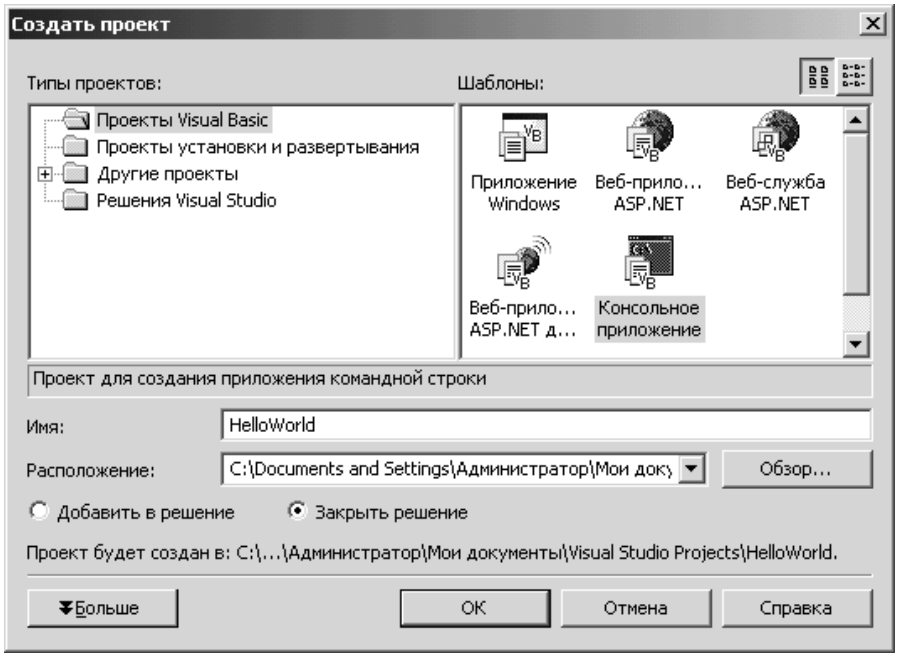


Рис. 3

3. Щелкните по кнопке *OK*. После этого будет создан новый программный проект, и на экране откроется окно редактирования текста программы (`Module1.vb`).

4. Наберите в этом окне приведенные ниже строки (удалив или изменив уже имеющиеся):

```
Class ClassHW
  ReadOnly Property Message() As String
    Get
      Return "Привет, Мир!"
    End Get
  End Property
End Class
```

```
Module Module1
  Sub Main()
    Dim ObjHW As New ClassHW
    System.Console.WriteLine(ObjHW.Message())
  End Sub
End Module
```

5. Нажмите комбинацию клавиш *Ctrl+F5*, чтобы откомпилировать программу, а затем запустить ее без отладки.

6. Если текст программы был набран правильно, то через некоторое время на экране появится окно консольного приложения, в котором вы увидите строку «Привет, Мир!» (рис. 4).

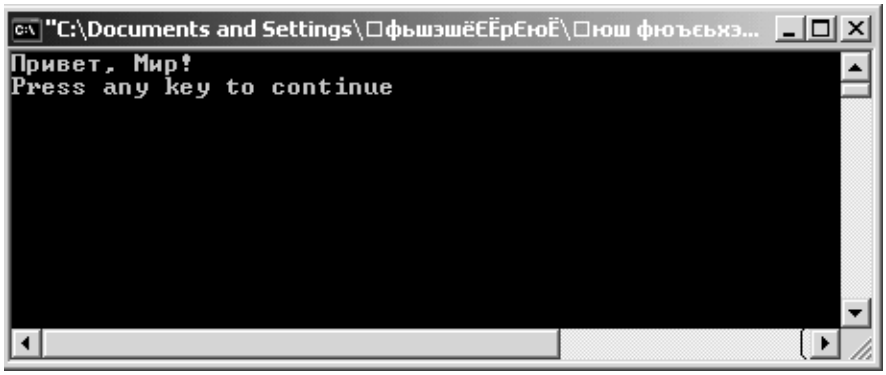


Рис. 4

7. Закройте Visual Studio .NET.

С учетом того, что вы ранее прочитали про объектно-ориентированное программирование, прокомментируем приведенный выше текст программы. В ее начале (от оператора `Class` до оператора `End Class` включительно) определяется класс с названием `ClassHW`. Для него задается одно-единственное свойство, названное `Message`. Это свойство — строка символов «Привет, Мир!» (слово `ReadOnly` указывает, что это свойство неизменяемое). Никакие методы, определяющие «поведение» класса, в программе не задаются.

В оставшейся части текста программы (начиная со строки `Module Module1`) указывается, что будет происходить при ее выполнении. Сначала (оператор Dim) будет создан объект с именем `ObjHW`, который получит все свойства класса `ClassHW`. Затем с помощью конструкции

```
System.Console.WriteLine(ObjHW.Message())
```

будет вызван метод `WriteLine` класса `Console`, который в .NET Framework определен в пространстве имен `System`. Этому методу в качестве параметра будет передано текущее значение свойства `Message` объекта `ObjHW` (обозначается как `ObjHW.Message()`). Вызов метода `WriteLine` используется для вывода информации в консольных приложениях. Поэтому результатом работы программы будет вывод строки «Привет, Мир!» в окне командной строки. Это окно будет находиться на экране до тех пор, пока вы не нажмете любую клавишу на клавиатуре.

Создание учетных записей для учеников

При проведении занятий по курсу «Основы программирования на примере Visual Basic .NET» учащиеся, работая на компьютерах в классе, должны выполнять практические задания. Разрешать им работать от имени пользователя *Администратор*, обладающего в системе Windows XP Professional [1] неограниченными полномочиями, слишком рискованно. Поэтому мы рассмотрим процедуру создания для учеников учетных записей пользователей (user accounts), не являющихся администраторами компьютера, но имеющих возможность работать в среде Visual Basic .NET. Вы можете создать такие учетные записи для каждого из них либо обойтись одной-единственной для всех.

Чтобы создать учетную запись пользователя в системе Windows XP Professional, можно воспользоваться оснасткой *Локальные пользователи и группы* в программе Управление компьютером. Но если вы хотите максимально автоматизировать этот процесс, лучше задействовать программы командной строки и создать командный файл, который можно будет повторно использовать в дальнейшем. Чтобы создать такой файл, а затем с его помощью пробную учетную запись пользователя, выполните следующее:

---

[1] Как, впрочем, и в других необходимых для установки Visual Basic .NET 2003 или Visual Basic 2005 Express Edition операционных системах (см. раздел «Системные требования»).

1. Работая на компьютере от имени *Администратора*, откройте окно командной строки (*Пуск-Все программы-Стандартные-Командная строка*).
2. Последовательно выполните следующие команды [1]:

```
echo NET USER %1 %2 /ADD /FULLNAME:%3 >newuser.cmd
echo NET LOCALGROUP "Пользователи отладчика"
%1 /ADD >>newuser.cmd
```

3. Затем подайте команду

```
newuser student Extybr "Ученик в классе"
```

   и убедитесь, что в обоих случаях команда NET была выполнена успешно. С ее помощью на компьютере будет создана новая учетная запись пользователя с именем `student` и паролем для входа в систему `Extybr` (лучше задать более сложный пароль). Кроме того, эта учетная запись будет добавлена в группу *Пользователи отладчика*. Последнее необходимо для того, чтобы пользователь имел возможность выполнять отладку создаваемых в системе Visual Studio .NET 2003 приложений.

4. Выйдите из системы, а затем снова войдите в нее как ученик. Запустите Visual Studio .NET 2003 и убедитесь, что у вас есть возможность создавать и запускать программы в этой системе.

Командный файл newuser.cmd облегчит процедуру создания индивидуальных учетных записей для учащихся, если вы решите остановиться на этом варианте.

Описанным выше способом подготовьте и все остальные компьютеры в классе [2].

Установка и использование Visual Basic 2005 Express Edition

Если при подготовке компьютеров класса к занятиям вы не можете использовать версию Visual Basic .NET 2003, воспользуйтесь дистрибути-

---

[1] Подробные сведения об этих и других командах приведены в имеющемся в Windows XP *Справочнике по командной строке*. Открыть его можно с помощью команды `hh ntcmds.chm`

[2] Для упрощения подготовки можно воспользоваться различными способами клонирования образа жесткого диска уже подготовленного компьютера — например, использовать специальные продукты (см. Altiris Deployment Solution (http://www.altiris.com), Symantec Ghost (http://www.symantec.com и др.).

вом Visual Basic 2005 Express Edition, который находится в папке PROGRAM на компакт-диске **Microsoft-CD**. При этом следует учесть, что:

- данный вариант Visual Basic не локализован для России, поэтому, работая с ним, ученики должны уметь ориентироваться в англоязычных названиях пунктов меню, диалоговых окон и т. д. Кроме того, если использовать стандартные для локализованной русской версии Windows XP Professional региональные установки, в процессе работы с Visual Basic 2005 Express Edition могут возникнуть проблемы [1];
- поставляемая с этой версией электронная документация не переведена на русский язык, что затрудняет ее использование в ходе занятий;
- все находящиеся на компакт-диске файлы готовых проектов подготовлены в системе Visual Basic .NET 2003. Чтобы использовать эти файлы при работе с версией 2005, надо сначала преобразовать их в новый формат [2].

Чтобы установить Visual Basic 2005 Express Edition (в предположении, что вы уже установили и настроили Windows XP Professional так, как описано выше), проделайте следующее:

1. Работая на компьютере от имени *Администратора*, вставьте компакт-диск **Microsoft-CD** в дисковод.
2. Откройте на компакт-диске папку PROGRAM\VisualBasic, а затем двойным щелчком запустите на выполнение программу DOTNETFX.EXE. Откроется диалоговое окно программы установки .NET Framework 2.0.
3. В этом окне щелкните по кнопке *Далее*, на следующей странице установите флажок *Я принимаю условия лицензионного соглашения* и щелкните по кнопке *Установить*.
4. Когда установка будет завершена, щелкните по кнопке *Готово*.

---

1 В частности, в качестве разделителя целой и дробной части чисел вместо запятой нужно использовать точку.

2 *Мастер обновления* автоматически запустится в системе Visual Basic 2005 Express Edition при попытке открыть файл, созданный в одной из предыдущих версий.
Для справки. Этот же мастер используется для преобразования проектов более ранних версий языка (VB 5.0 CCE, Visual Basic 6.0) в проекты на Visual Basic .NET. Чтобы преобразовать проект, необходимо ввести команду [*File-Open-Project...*], найти в папке проекта на языке Visual Basic 5.0 или 6.0 файл с расширением vbp, а в папке проекта на языке Visual Basic .NET 2003 — файл с расширением vbproj, и следовать указаниям в диалоговых окнах мастеров.

5. Дважды щелкните по названию файла IXPVB.EXE (папка PROGRAM\VisualBasic). Запустится программа установки Visual Basic 2005 Express Edition.

6. Если в ходе ее работы на экране будут появляться диалоговые окна с сообщением об ошибке («Source file not found»), просто щелкайте по кнопке *Ignore*, чтобы продолжить процесс установки.

7. После того как установка завершится, запустите программу MSDNIXP.EXE (папка PROGRAM\VisualBasic). На экране появится окно программы установки усеченного варианта библиотеки MSDN.

8. В этом окне щелкните по кнопке *Next*, на следующей странице установите флажок *I accept the terms of the License Agreement* и щелкните по кнопке *Install*.

9. Щелкните по кнопке *Finish*, когда установка завершится.

Аналогично тому, как было описано в разделе «Проверка работы Visual Basic .NET 2003», проверьте возможность создания приложений в системе Visual Basic 2005 Express Edition. Попробуйте в этой системе открыть файл проекта или решения из папки PRACTICUM и ознакомьтесь с процедурой преобразования этих файлов в новый формат.

Дополнительные материалы для подготовки класса

На компакт-диске **Microsoft-CD** имеются файлы с таблицами, блок-схемами и другой справочной информацией, которая может оказаться полезной учащимся во время занятий по курсу «Основы программирования на примере Visual Basic .NET». Вы найдете файлы таблиц в папке PLACAT.

Определитесь, будете ли вы использовать таблицы в учебном процессе. Если да, то их имеет смысл распечатать на принтере (лучше в формате A3) и вывесить в том классе, где будут проходить занятия по курсу.

6. Рекомендации по преподаванию курса «Основы программирования на примере Visual Basic .NET»

Пояснительная записка

В новом образовательном стандарте на третьей ступени общего образования, т. е. в старшей школе (10–11 классы), предусмотрено изучение элективных курсов. Данный курс можно преподавать в качестве элективного.

Классы: 10 или 11.

Количество часов: 68 (46 часов в урочной форме, 22 — во внеурочной).

Профили: естественно-математический и информационно-технологический.

Цель курса: научить учащихся основам объектно-ориентированного программирования с использованием системы программирования Visual Basic .NET.

Компьютерный практикум. Разработка каждого проекта реализуется в форме выполнения практической работы на компьютере (*компьютерный практикум*). В учебном пособии содержатся подробные указания по построению компьютерных моделей и их реализации в форме проектов на языках программирования и в электронных таблицах.

Кроме разработки проектов под руководством учителя учащимся предлагаются *практические задания для самостоятельного выполнения*. В учебном пособии содержатся указания по их выполнению, а на Microsoft-CD хранятся готовые проекты на языке объектно-ориентированного программирования Visual Basic.

Метод обучения. Основным методом обучения в данном курсе является *метод проектов*. Проектная деятельность позволяет развить исследовательские и творческие способности учащихся. В процессе обучения учитель кратко объясняет новый материал и ставит задачи, а затем консультирует учащихся в процессе решения этих задач. Учащиеся решают задачи, в основном практические, реализуя проекты по созданию приложений на компьютере (*компьютерный практикум*). Подробные указания по построению компьютерных моделей и их реализации в форме проектов на языке программирования Visual Basic .NET содержатся в учебном пособии к курсу.

Индивидуализация обучения. В учебном пособии имеются задания (всего их 63) разного уровня сложности. Это позволяет учителю построить для каждого учащегося индивидуальную образовательную траекторию.

Контроль знаний и умений. Текущий контроль уровня усвоения материала должен осуществляться, в основном, по результатам выполнения учащимися практических заданий на компьютере. Для грубой оценки можно воспользоваться результатами ответов на вопросы тестов, приведенных в конце каждой главы учебного пособия (ответить на эти же вопросы можно с помощью интерактивных веб-страниц, доступных на **Microsoft-CD**).

Итоговый контроль осуществляется по результатам *защиты итоговых проектов*. В начале курса каждому учащемуся нужно предложить в течение всего времени изучения курса разработать проект для решения некоторой задачи. В процессе защиты учащийся должен представить работающую компьютерную программу, которая решает поставленную перед ним задачу, и обосновать способ ее решения.

Организация учебного процесса. Курс предусматривает организацию учебного процесса в двух взаимосвязанных и взаимодополняющих формах:

- *урочная форма*, в которой учитель на уроке объясняет новый материал и консультирует учащихся в процессе выполнения ими практических заданий на компьютере;
- *внеурочная форма*, в которой учащиеся после уроков (дома или в школьном компьютерном классе) выполняют на компьютере практические задания самостоятельно.

Тематическое планирование курса

Ниже приведены рекомендации, касающиеся тематического планирования курса. Материал в учебном пособии разбит на 12 глав, поэтому тематический план разделен на 12 модулей. Каждый модуль предусматривает как изучение теории, так и выполнение практических заданий. Поэтому в плане приведены главы и разделы учебного пособия и методического пособия, которые следует проработать учителю при подготовке к ведению того или иного модуля курса, а также список практических заданий, которые ученики должны выполнить в ходе занятий (на уроках и самостоятельно).

В разделе «Компьютерный практикум» предусмотрено выполнение 63 практических заданий.

Теория	Компьютерный практикум
Учебный модуль 1 «Программы в повседневной жизни» — 2 часа	
1.1. Программы в повседневной жизни. 1.2. Чем занимаются программисты. 1.3. Что такое программа. 1.4. Возможности языков программирования. 1.5. Синтаксис языков программирования	
Контроль знаний и умений:	Тест 1
Учебное пособие: Глава 1. Программы в повседневной жизни. **Методическое пособие:** 1. История развития языков программирования	**Microsoft-CD:** Глава 1. Программы вокруг нас
Учебный модуль 2 «Система программирования Visual Basic .NET» — 6 часов	
2.1. Visual Basic .NET и IDE. 2.2. Запуск и настройка Visual Studio .NET. 2.3. Создание первого проекта. 2.4. Конструирование графического интерфейса проекта. 2.5. Создание программного кода проекта. 2.6. Построение решения. 2.7. Запуск проекта. 2.8. Сохранение проекта. 2.9. Вывод сообщений на форму	1. Проект «Привет, Мир». 2. Проект «Мое имя»
Контроль знаний и умений:	Тест 2
Учебное пособие: Глава 2. Система программирования Visual Basic .NET. **Методическое пособие:** 5. Подготовка учебного класса к занятиям	**Microsoft-CD:** Глава 2. Система программирования Visual Basic .NET
Учебный модуль 3 «Алгоритмы и программы» — 6 часов	
3.1. Основные элементы кода. 3.2. Алгоритм в форме псевдокода. 3.3. Комментарии в коде	1. Записать в форме псевдокода и построить блок-схему алгоритма включения компьютера, запуска операционной системы Windows и системы программирования Visual Basic .NET. 2. Проект «Цена бензина». 3. Проект «Цвет рыбок»

Теория	Компьютерный практикум
Контроль знаний и умений:	Тест 3
Учебное пособие: Глава 3. Алгоритмы и программы. **Методическое пособие:** Приложение 2, параграф 3. Основные типы алгоритмических структур и их кодирование на языке Visual Basic	**Microsoft-CD:** Глава 3. Алгоритмы и программы
Учебный модуль 4 «Формы и элементы управления» — 6 часов	
4.1. Форма — основа графического интерфейса. 4.2. Свойства форм. 4.3. Элементы управления и их свойства. 4.4. Генерация событий	1. Проект «Замена шины». 2. Проект «Цена пиццы»
Контроль знаний и умений:	Тест 4
Учебное пособие: Глава 4. Формы и элементы управления. **Методическое пособие:** 2. Введение в объектно-ориентированное визуальное программирование. 3. Краткий обзор .NET Framework и Visual Studio .NET. Приложение 1. Реализация объектно-ориентированного программирования в Visual Basic .NET	**Microsoft-CD:** Глава 4. Формы и элементы управления. Таблица 1. Основы объектно-ориентированного программирования. Таблица 2. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (начало). Таблица 3. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (продолжение 1). Таблица 4. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (продолжение 2). Таблица 5. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (продолжение 3)
Учебный модуль 5 «Свойства и методы» — 6 часов	

Теория	Компьютерный практикум
5.1. С чего начинается код. 5.2. Чтение значений свойств в коде. 5.3. Присваивание значений свойствам в коде. 5.4. IntelliSense и точечная нотация. 5.5. Методы	1. Проект «Чтение свойств». 2. Проект «Установка свойств». 3. Проект «Intellisence». 4. Проект «Методы»
Контроль знаний и умений:	Тест 5
Учебное пособие: Глава 5. Свойства и методы. **Методическое пособие:** Приложение 2, параграф 5. Возможности работы с графикой в Visual Basic .NET 2003 и Visual Basic 2005 Express Edition	**Microsoft-CD:** Глава 5. Свойства и методы
Учебный модуль 6 «Присваивание и переменные» — 6 часов	
6.1. Присваивание. 6.2. Переменные. 6.3. Объявление переменных. 6.4. Переменные в программах	1. Проект «Прыжок лягушки». 2. Проект «Цвет». 3. Проект «Число Pi». 4. Проект «Количество щелчков». 5. Проект «Найди ошибки»
Контроль знаний и умений:	Тест 6
Учебное пособие: Глава 6. Присваивание и переменные. **Методическое пособие:** Приложение 2, параграф 1. Переменные в языке программирования Visual Basic .NET	**Microsoft-CD:** Глава 6. Присваивание и переменные. Таблица 6. Типы переменных
Учебный модуль 7 «Операции» — 6 часов	
7.1. Арифметические операции. 7.2. Строковые операции. 7.3. Логические операции. 7.4. Отладка кода	1. Проект «Цена бензина-1». 2. Проект «Диаметр окружности». 3. Проект «Путь». 4. Проект «Цена бензина-2». 5. Проект «Имя, фамилия». 6. Проект «Логика». 7. Проект «Отладка». 8. Проект «Объем»
Контроль знаний и умений:	Тест 7
Учебное пособие: Глава 7. Операции. **Методическое пособие:** Приложение 2, параграф 4. Функции в языке программирования Visual Basic .NET	**Microsoft-CD:** Глава 7. Операции
Учебный модуль 8 «Ветвление: неполная форма» — 6 часов	

Теория	Компьютерный практикум
8.1. Булева логика. 8.2. Операции сравнения. 8.3. Оператор If...Then. 8.4. Множественные условия. 8.5. Булевы операции в коде	1. Проект «Логические операции». 2. Проект «If_Then». 3. Проект «Пожарная тревога». 4. Проект «Выбор цвета». 5. Проект «Выбор цвета-2». 6. Проект «Выбор подарка». 7. Проект «Проверка»
Контроль знаний и умений:	Тест 8
Учебное пособие: Глава 8. Ветвление: неполная форма. **Методическое пособие:** Приложение 2, пункт 3.1. Алгоритмическая структура «ветвление»	**Microsoft-CD:** Глава 8. Ветвление: неполная форма. Таблица 7. Алгоритмическая структура «ветвление» и ее кодирование на языке программирования
Учебный модуль 9 «Ветвление: полная форма» — 6 часов	
9.1. Вложенные операторы If...Then. 9.2. Противоположные условия. 9.3. Оператор If...Then...Else. 9.4. Пошаговое выполнение If. 9.5. Операторы If в языках C# и J#. 9.6. Булевы операции и операции сравнения в C# и J#	1. Проект «Флажки». 2. Проект «IfThenOtherwise». 3. Проект «If-Then-Else». 4. Проект «If-Then-Else-2». 5. Проект «Гольф». 6. Проект «Step-In-If». 7. Проект «Магический квадрат». 8. Проект «Бросок монеты»
Контроль знаний и умений:	Тест 9
Учебное пособие: Глава 9. Ветвление: полная форма. **Методическое пособие:** Приложение 2, пункт 3.1. Алгоритмическая структура «ветвление»	**Microsoft-CD:** Глава 9. Ветвление: полная форма. Таблица 7. Алгоритмическая структура «ветвление» и ее кодирование на языке программирования
Учебный модуль 10 «Циклы со счетчиком» — 6 часов	
10.1. Циклы For...Next. 10.2. Пошаговое выполнение цикла For...Next. 10.3. Проекты с использованием For...Next. 10.4. Вложенные циклы. 10.5. Выход из циклов. 10.6. Циклы в C# и J#	1. Проект «For-Next». 2. Проект «Сложение». 3. Проект «Сложение строк». 4. Проект «Цвет формы». 5. Проект «Подсчет кроликов». 6. Проект «Цикл в цикле». 7. Проект «Выход из циклов». 8. Проект «Простые числа». 9. Проект «Пенсия»
Контроль знаний и умений:	Тест 10

Теория	Компьютерный практикум
Учебное пособие: Глава 10. Циклы со счетчиком. **Методическое пособие:** Приложение 2, пункт 3.3. Алгоритмическая структура «цикл»	**Microsoft-CD:** Глава 10. Циклы со счетчиком. Таблица 9. Алгоритмическая структура «цикл со счетчиком» и ее кодирование на языке программирования
Учебный модуль 11 «Циклы с условием» — 6 часов	
11.1. Циклы Do While...Loop. 11.2. Пошаговое выполнение цикла Do While...Loop. 11.3. Циклы Do Until...Loop. 11.4. Проекты с использованием Do...Loop. 11.5. Циклы с постусловием. 11.6. Циклы в C# и J#. 11.7. Выход из циклов	1. Проект «Do-While-Loop». 2. Проект «Do-Until-Loop». 3. Проект «Do-While-Loop-2». 4. Проект «Do-Until-Loop-2». 5. Проект «Циклы с постусловием». 6. Проект «Exit-Do». 7. Проект «Мишень». 8. Проект «Монета». 9. Проект «Средняя оценка»
Контроль знаний и умений:	Тест 11
Учебное пособие: Глава 11. Циклы с условием. **Методическое пособие:** Приложение 2, пункт 3.3. Алгоритмическая структура «цикл»	**Microsoft-CD:** Глава 11. Циклы с условием. Таблица 10. Алгоритмическая структура «цикл с условием» и ее кодирование на языке программирования
Учебный модуль 12 «Подпрограммы и функции» — 6 часов	
12.1. Что такое подпрограммы? 12.2. Создание и вызов подпрограмм. 12.3. Подпрограммы с аргументами. 12.4. Создание и вызов собственных функций. 12.5. Встроенные функции. 12.6. Функции в J# и C#	1. Проект «Подпрограмма». 2. Проект «Функция». 3. Проект «Единицы измерения». 4. Проект «Встроенные функции». 5. Проект «Предсказатель». 6. Проект «Кот и мышь»
Контроль знаний и умений:	Тест 12
Учебное пособие: Глава 12. Подпрограммы и функции. **Методическое пособие:** Приложение 1. Реализация объектно-ориентированного программирования в Visual Basic .NET	**Microsoft-CD:** Глава 12. Подпрограммы и функции

Ниже даны советы по проведению занятий и рекомендации учителю по способам представления материала по каждому модулю.

Модуль 1. Программы вокруг нас

Примерное время изложения теоретического материала: 1,5 ч.
Примерное время проведения практических занятий: 0,5 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 1 учебного пособия;
- постарайтесь сами ответить на приведенные в пособии вопросы для размышления;
- ознакомьтесь с содержимым 1-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 1.

Как учить в данном модуле

Начните первый урок модуля с опроса: у кого из учащихся есть дома персональный компьютер и для чего он в основном используется? На основе ответов покажите, что компьютер превращается в полезный инструмент исключительно благодаря работающим на нем программам. Приведите примеры устройств, которые не считаются компьютерами, но в которых работают программы (музыкальный центр, мобильный телефон и т. п.).

Затем переходите к разделу, посвященному рассказу о программистах, т. е. людях, создающих программы. Поинтересуйтесь, кто из учеников считает себя программистом. Если таковые найдутся, спросите, какие программы написал кто-нибудь из них, для чего эти программы были написаны. После этого переходите к обсуждению того, что такое компьютерные программы и как они разрабатываются.

Обратите внимание учеников, что программы могут создаваться для самых разных применений. Соответственно существует множество разных языков программирования. Приведите примеры известных языков: FORTRAN, BASIC, Pascal.

Начало следующего урока можно посвятить обсуждению того, что такое синтаксис языков программирования, и постепенно перейти к рассмотрению Visual Basic .NET как языка, который будет изучаться в ходе курса.

В завершение объясните ученикам, как начать работать на компьютерах в классе, и проконтролируйте выполнение ими входа в систему. Ознакомьте учащихся с содержимым компакт-диска с материалами к курсу.

Оцените степень усвоения пройденного материала с помощью теста 1.

Модуль 2. Система программирования Visual Basic .NET

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 2 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 2-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 2.

Как учить в данном модуле

Начните модуль с повторения того, что такое компьютерные программы и языки программирования. Затем расскажите, что раньше программы создавались, в основном, с помощью нескольких разных инструментов. Для ввода и редактирования текста программы использовался текстовый редактор, для преобразования текста в машинные команды вызывался компилятор и т. п.

После этого перейдите к рассмотрению интегрированной среды разработки Visual Studio .NET. Подчеркните, что слово «интегрированная» означает, что в составе среды имеются все инструменты, необходимые для написания, редактирования, компиляции, отладки и запуска программ. Также обратите внимание учеников, что в этой среде программы можно создавать с помощью самых разных языков программирования (см. список в учебном пособии).

После этого ученики могут приступать к освоению основных приемов работы в Visual Studio .NET. Проконтролируйте, все ли смогли запустить

эту систему на компьютере и настроить профиль. После этого дайте ученикам задание создать проект «Привет, Мир», руководствуясь приведенными в учебном пособии инструкциями. При необходимости оказывайте им поддержку и консультируйте по вопросам выполнения тех или иных действий. Если кто-то из них будет интересоваться подробностями записи тех или иных языковых конструкций, скажите, что это будет обсуждаться в курсе позднее.

После создания и успешного запуска первого приложения проверьте усвоение учащимися пройденного материала с помощью теста 2.

Затем раздайте учащимся задания для самостоятельного выполнения, используя материалы учебного пособия. Расскажите, где на жестком диске компьютера (или компакт-диске **Microsoft-CD**) находятся файлы с заготовками проектов и как этими файлами воспользоваться.

Продумайте, как вы будете осуществлять контроль выполнения заданий.

Модуль 3. Алгоритмы и программы

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 3 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 3-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 3.

Как учить в данном модуле

В начале модуля обсудите, что такое *код* (другими словами, текст программы) и кодирование (т. е. написание этого текста). Расскажите об основных элементах кода, которые имеются практически во всех языках программирования. Не углубляйтесь при этом в детали записи тех или иных элементов на языке Visual Basic .NET, поскольку этот материал будет рассматриваться в курсе позднее.

Сформулируйте понятие *алгоритма* как набора инструкций, описывающих последовательность действий, выполнение которых ведет к реше-

нию задачи. Затем приведите пример какого-либо алгоритма. Объясните, что *псевдокод* — это описание алгоритма работы программы на естественном языке. Дайте учащимся задание написать с помощью псевдокода алгоритм решения какой-либо несложной задачи из обыденной жизни (используйте примеры, приведенные в учебном пособии, либо придумайте свои). Обсудите, что получилось в результате выполнения задания.

После этого перейдите к обсуждению необходимости использования комментариев в программном коде. Отметьте, что использование комментариев допускается практически во всех языках программирования и что в разных языках комментарии обозначаются по-разному. Также обратите внимание учеников на то, что комментирование фрагментов кода — один из весьма распространенных и эффективных приемов выявления ошибок в программах.

В заключение расскажите об использовании отступов и пробелов для улучшения читаемости кода. Отметьте, что при программировании в системе Visual Basic .NET отступы и пробелы могут вставляться в текст программы автоматически при его наборе.

Проверьте усвоение пройденного материала с помощью теста 3.

После этого учащиеся могут переходить к выполнению практических заданий, а ваша задача — контролировать этот процесс и при необходимости помогать им.

Модуль 4. «Формы и элементы управления»

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 4 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 4-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- распечатайте таблицы, находящиеся в папке Плакаты по языку Visual Basic .NET;
- ответьте на вопросы теста 4.

Как учить в данном модуле

В начале модуля сформулируйте определение *пользовательского интерфейса* как способа взаимодействия пользователя и компьютерной программы. Объясните, чем отличается графический интерфейс пользователя от интерфейса командной строки. Поинтересуйтесь, какой способ взаимодействия с программой предпочитают ваши ученики, и попросите их прокомментировать свои предпочтения.

Затем расскажите, что в системе Windows чаще используются приложения с графическим интерфейсом пользователя. Если создавать такое приложение в Visual Basic .NET, то в файл проекта автоматически добавляется форма — шаблон окна приложения. Спросите, что произойдет, если создать новый проект для Windows-приложения, а затем сразу запустить его. Будет ли такое приложение взаимодействовать с пользователем? Ответ: будет, но только через стандартные элементы управления окном (кнопки *Свернуть*, *Развернуть*, *Закрыть* и так называемое *оконное меню* программы).

Затем приведите примеры различных элементов управления, которые можно в процессе разработки размещать на форме. Объясните, что именно эти элементы после запуска готового приложения обеспечивают взаимодействие между ним и пользователем. Обсудите основные правила размещения элементов управления на форме. Кратко опишите основные свойства, которые можно задать и для формы, и для различных элементов управления.

После этого учащиеся могут начать работу над проектом «Замена шины».

Когда бóльшая часть учеников выполнит задание по созданию в данном проекте формы и размещению на ней элементов управления, расскажите о том, как в программе задается реакция на события, связанные с действиями пользователя. Объясните, что такое *обработчик события* (процедура, которая выполняется в программе, когда это событие происходит).

Не забудьте проверить усвоение учащимися пройденного материала с помощью теста 4.

Используя материалы учебного пособия, раздайте учащимся задания для самостоятельного выполнения.

Модуль 5. «Свойства и методы»

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 5 учебного пособия;
- перечитайте материал раздела «2. Введение в объектно-ориентированное программирование» данного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 5-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 5.

Как учить в данном модуле

Начните модуль с обсуждения следующего вопроса: как узнать или изменить значение какого-либо свойства формы или элемента управления после того, как программа запущена? Ответ: с помощью инструкций (или кода) в самой программе. Расскажите, что формы и элементы управления суть программные *объекты*, и для обращения к их свойствам в программах на языке Visual Basic .NET нужно использовать *точечную нотацию* вида `Объект.Свойство`.

После этого учащиеся могут начать работу над проектами «Чтение свойств» и «Установка свойств», а вы — контролировать ход ее выполнения.

Затем перейдите к объяснению сути технологии IntelliSence. Расскажите, как с ее помощью в системе программирования Visual Studio .NET упрощается процесс написания операторов, в которых осуществляется обращение к свойствам тех или иных программных объектов.

Далее обратите внимание учеников, что в списке IntelliSence кроме свойств присутствуют также и методы. Напомните, что с помощью свойств разработчик может задавать характеристики объектов, а методы позволяют ему управлять действиями этих объектов при работе программы. Сделайте акцент на том, что совершенно не случайно названия свойств — это существительные (например, `TextBox1.Text`), а названия методов — глаголы (например, `TextBox1.Update`).

В ходе изучения материала модуля учащиеся могут выполнять задания по учебному пособию, ваша задача состоит в том, чтобы контролировать, как они выполняются.

Ближе к концу модуля с помощью теста 5 проверьте, как ученики усвоили пройденный материал.

Модуль 6. Присваивание и переменные

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 6 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 6-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 6.

Как учить в данном модуле

В начале занятий запишите на доске уже знакомый учащимся оператор вида

```
TextBox1.Text = "Текст"
```

и попросите кого-нибудь из них объяснить, что означает такая конструкция. Если ответ будет правильным, повторите его. Не забудьте дать название этого оператора и объясните, что присваивается (текст) и чему (свойству объекта).

Затем на примере свойства Text расскажите, что выражение

```
TextBox1.Text = TextBox1.ForeColor
```

является ошибочным из-за несоответствия *типов* величин слева и справа от знака равенства. Используйте какие-либо аналогии из обыденной жизни, например невозможность поставить автобус в гараж для легкового автомобиля. Объясните, что тип величины определяет множество значений, которые она может принимать, и набор операций, которые над

ней можно производить (конкретные диапазоны значений для данных различных типов приведены в табл. 6 приложения 3).

Далее переходите к рассмотрению понятия *переменной*, которую можно определить как именованную область памяти, где может храниться какая-то информация (данные). Расскажите о разных типах переменных, которые можно использовать в Visual Basic .NET, а также о правилах записи значений этих переменных.

Попросите учащихся создать проект «Цвет» и поэкспериментировать с типами данных класса System.

Расскажите о необходимости объявления переменных в программе с помощью оператора Dim. Рассмотрите разные способы записи этого оператора и сформулируйте основные правила именования переменных.

Рассказывая о *локальных* и *глобальных* переменных, можно упомянуть, что в Visual Basic .NET обработчик событий — это один из видов подпрограмм, которые будут рассмотрены в конце курса. Локальными являются любые переменные, которые объявлены в подпрограмме.

После этого учащиеся могут приступать к выполнению оставшихся заданий данного модуля.

Не забудьте оценить степень усвоения пройденного материала с помощью теста 6.

Модуль 7. Операции

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 7 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 7-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 7.

Как учить в данном модуле

Объясните, что такое *операция*, покажите общий синтаксис для разных операций. В качестве примера можно использовать арифметические

операции (+, –, *, /). Обратите внимание на приоритет операций, расскажите о возможности использования скобок для изменения порядка вычислений.

Затем напомните учащимся, что такое оператор присваивания. Не забудьте сказать, что в качестве операндов в выражении справа можно использовать любые переменные — даже стоящую в левой части оператора присваивания и что присваивание происходит только после завершения вычисления значения выражения. Вполне можно использовать конструкцию вида

```
a = a + a
```

Также напомните, что типы величин в левой и правой частях оператора присваивания должны совпадать. Объясните, как можно использовать функцию `Val()` для преобразования строк в числа. Дайте ученикам задание создать проект «Цена бензина-2».

Расскажите об операции конкатенации строк. Не забудьте напомнить ученикам о том, что Visual Basic .NET не вставляет между строками пробелы, так что это должен делать сам программист.

Затем расскажите о булевой логике, величинах `True` и `False`. Перейдите к логическим операциям `AND`, `NOT`, `OR`, упомянув о таблицах истинности соответствующих операций (эти таблицы приведены в главе 8 учебного пособия). Дайте учащимся задания на вычисление результатов выражений с логическими операциями (используйте примеры, приведенные в учебном пособии, либо придумайте свои). Расскажите об операциях сравнения (<, >, <=, >=, <>, =). Не забудьте упомянуть о том, что при помощи них можно сравнивать не только числа, но и строки или булевы выражения.

После этого объясните ученикам, что такое отладка. Обратите особое внимание на то, что если непонятно, почему тот или иной код не работает, следует использовать отладочные средства Visual Studio .NET, а не пытаться заставить код работать методом проб и ошибок. Покажите, как Visual Studio .NET помогает выявлять ошибки в коде еще при его написании (на примере необъявленной переменной), а затем объясните ученикам, как можно использовать точки останова и пошаговое выполнение для отладки во время выполнения программы.

Проверьте усвоение пройденного материала с помощью теста 7.

Модуль 8. Ветвление: неполная форма

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 8 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 8-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 8.

Как учить в данном модуле

Напомните ученикам о логических операциях и правилах их выполнения.

Расскажите о синтаксисе условного оператора «если...то» (`If … Then … End If`), после чего дайте ученикам задание создать проект «If_Then», руководствуясь приведенными в учебном пособии инструкциями. При необходимости оказывайте им поддержку и консультируйте по вопросам выполнения тех или иных действий. Напомните о выделении кода отступами при использовании данного оператора. Не забудьте рассказать о двух формах записи условного оператора: однострочной (без `End If`) и многострочной. Можно также рассказать о возможности записи нескольких операторов на одной строке (разделяя их символом «:»).

Упомяните о том, что вычисление результатов логических операций в приложениях .NET идет по краткой форме: если значения левого аргумента логической операции достаточно для получения результата (например, `A AND B` всегда ложно, если A ложно), значение правого аргумента не будет вычисляться.

После этого объясните, как можно использовать последовательности операторов `If … Then` для выбора различных вариантов выполнения кода и расскажите о возможности использования логических операций для проверки истинности нескольких условий (проект «Выбор цвета»).

С помощью теста 8 оцените степень усвоения пройденного материала.

Модуль 9. Ветвление: полная форма

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 9 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 9-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 9.

Как учить в данном модуле

Расскажите о возможности вложения операторов `If … Then` друг в друга для того, чтобы какой-то код выполнялся только при истинности нескольких условий. Упомяните, что нет никаких ограничений на уровень вложенности операторов.

Затем перейдите к необходимости выполнения разных действий при истинности и ложности какого-либо условия. Приведите пример с двумя последовательными операторами `If … Then` с противоположными условиями. Объясните, что такой подход отнимает лишнее время и легко может стать причиной ошибок, если условие нужно будет изменить. Расскажите об операторе `If … Then … Else`, после чего дайте ученикам задание создать проект «If-Then-Else», руководствуясь инструкциями из учебного пособия. Упомяните, что у этого оператора также есть однострочная форма записи.

Посоветуйте ученикам с помощью отладочных средств Visual Studio .NET (например, пошагового выполнения программы) проверить, как выполняется код в операторах `If … Then … Else`.

Кратко расскажите об основных отличиях оператора ветвления в Visual Basic .NET и C# или J#. Заострите внимание на отличающихся операциях сравнения (== вместо = и != вместо <>), отсутствии ключевого слова `Then`, необходимости заканчивать оператор точкой с запятой. Отметьте, что в C# и J# группы операторов заключаются в фигурные скобки.

Проверьте усвоение пройденного материала с помощью теста 9.

Модуль 10. Циклы со счетчиком

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 10 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 10-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 10.

Как учить в данном модуле

В начале модуля обсудите необходимость поддержки циклов в любом языке программирования, используя примеры из учебного пособия. Упомяните о разных видах циклов (со счетчиком, с условием). Затем расскажите о синтаксисе оператора `For … Next`, обратив внимание на то, что к значению переменной-счетчика можно обращаться внутри цикла. Расскажите о ключевом слове `Step`, позволяющем определять шаг цикла, после чего дайте ученикам задание создать проект «For-Next». Предложите им воспользоваться пошаговым выполнением программы, чтобы выяснить, как выполняется цикл `For … Next`.

Затем расскажите о возможности вкладывать друг в друга циклы и другие операторы. Объясните, как использовать оператор `Exit For` для преждевременного выхода из цикла на примере проекта «Выход из циклов».

После этого дайте ученикам задание создать проект «Простые числа», руководствуясь приведенными в учебном пособии инструкциями. При необходимости консультируйте их по вопросам выполнения тех или иных действий. Обратите внимание на необходимость использования в данном проекте оператора `Exit For` (если в цикле проверки числа на делимость получилось, что это число делится на что-то, кроме самого себя и единицы), так как без него большая часть вычислений при работе данной программы будет идти впустую.

Кратко расскажите об основных отличиях оператора цикла со счетчиком в Visual Basic .NET и C# или J#: инициализации переменной-счетчика в самом цикле, другой форме записи конечного условия, операторе **++**.

Проверьте усвоение пройденного материала с помощью теста 10.

Модуль 11. Циклы с условием

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 11 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 11-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 11.

Как учить в данном модуле

Объясните, что в некоторых случаях использовать циклы со счетчиком нельзя, так как неизвестно количество выполнений цикла. Расскажите о циклах с условием, после чего объясните синтаксис оператора `Do While…Loop`. Приведите примеры подобных циклов (их можно взять из учебного пособия), обратив особое внимание на то, что условие не обязательно должно содержать в себе какой-то счетчик.

При помощи отладочных средств Visual Studio .NET пошагово покажите, как проходит выполнение цикла `Do While…Loop`. Упомяните о том, что ошибки, допущенные при написании программы, могут привести к появлению бесконечного цикла, покажите, как при этом остановить программу.

Затем расскажите о циклах `Do Until…Loop` и о циклах с постусловием (обратите особое внимание на то, что они выполняются как минимум один раз). Дайте ученикам задание создать проект «Do-While-Loop-2», после этого попросите модифицировать его так, чтобы использовался цикл `Do Until…Loop`.

Кратко расскажите об основных отличиях оператора цикла со счетчиком в Visual Basic .NET и C# или J#: отсутствии ключевых слов `Until` и `Loop` и разнице в синтаксисе оператора цикла.

Проверьте усвоение пройденного материала с помощью теста 11.

Модуль 12. Подпрограммы и функции

Примерное время изложения теоретического материала: 2 ч.
Примерное время проведения практических занятий: 4 ч.

Для подготовки к проведению занятий в данном модуле

- внимательно прочитайте материалы главы 12 учебного пособия;
- выполните все приведенные в тексте учебного пособия практические задания по созданию проектов в системе Visual Studio .NET;
- ознакомьтесь с содержимым 12-й части в разделе «Практикум по программированию на Visual Basic .NET» на компакт-диске **Microsoft-CD**;
- ответьте на вопросы теста 12.

Как учить в данном модуле

Расскажите о том, что такое подпрограмма, зачем она нужна. Объясните, что написание и отладка набора подпрограмм, каждая из которых отвечает за некоторое действие, намного удобнее написания и отладки неструктурированной программы. Упомяните, что подпрограммы помогают повторно использовать уже написанный код, уменьшая размер программы, облегчая отладку и экономя время.

Затем расскажите о синтаксисе объявления подпрограммы (в вариантах без аргументов и с аргументами). Объясните, как нужно вызывать подпрограммы, упомянув о том, что это можно делать откуда угодно — даже из других подпрограмм. Напомните учащимся, что такое локальные переменные, не забыв упомянуть о том, что после завершения подпрограммы они становятся недоступными. После этого дайте ученикам задание создать проект «Подпрограмма», руководствуясь инструкциями из учебного пособия.

Затем расскажите о функциях, обязательно упомянув о необходимости возврата какого-нибудь значения. Покажите на примере проекта «Функция», что при вызове функции в качестве параметра можно использовать в том числе и результат вызова любой функции, даже той же самой. Расскажите о встроенных функциях платформы .NET, используя примеры из учебного пособия.

Кратко расскажите об отличиях синтаксиса функций и подпрограмм Visual Basic .NET и C# или J#.

Проверьте усвоение пройденного материала с помощью теста 12.

7. Дополнительная литература

Книги

К настоящему моменту издано весьма немало книг, которые могут оказаться полезными при подготовке и проведении занятий по обсуждаемому в данном пособии курсу. Это и книги, посвященные технологиям .NET Framework, и издания, в которых рассматриваются использование среды разработки Visual Studio .NET и программирование на языке Visual Basic .NET. Некоторые из них перечислены ниже:

1. *Хальворсон М.* Visual Basic.NET. версия 2003. Русская версия. (+CD). — М.: ЭКОМ.
2. *Рамел Д.* Visual Basic .NET. Справочник программиста. — М.: ЭКОМ, 2002.
3. *Гарнаев А.* Visual Basic .NET. Разработка приложений. — СПб.: БХВ-Петербург, 2002.
4. *Пономарев В.* Visual Basic .NET. — СПб.: БХВ-Петербург, 2003.
5. *Долженков В.*, *Мозговой М.* Visual Basic .NET. Учебный курс. СПб.: Питер, 2003.
6. *Климов А.* Занимательное программирование на Visual Basic .NET. — СПб.: БХВ-Петербург, 2005.
7. *Гарнаев А.* Самоучитель Visual Studio .NET 2003. — СПб.: БХВ-Петербург, 2003.
8. *Джонсон Б.*, *Скибо К.*, *Янг М.* Основы Microsoft Visual Studio .NET 2003. — М.: Русская Редакция, 2003.
9. *Туан Тай*, *Хонг К. Лэм.* Платформа .NET. Основы. — М.: Символ-Плюс, 2003.

Ссылки на информацию о .NET Framework, Visual Studio .NET и Visual Basic .NET в Интернете

1. Русскоязычный сайт корпорации Майкрософт, посвященный использованию среды разработки Visual Studio .NET (версий 2003 и 2005): http://www.microsoft.com/Rus/Msdn/vs
2. Раздел упомянутого выше сайта о Visual Basic .NET: http://www.microsoft.com/Rus/Msdn/vbasic

3. Библиотека MSDN на русском языке, где собрана большая подборка документации:
   http://msdn.microsoft.com/library/rus
4. Общие сведения о платформе .NET, информация о программировании для нее, примеры, дополнительные программы (на английском языке):
   http://msdn.microsoft.com/netframework
5. Аналогичный русскоязычный сайт со статьями, примерами, досками обсуждений и т. д.:
   http://www.gotdotnet.ru/
6. Азы программирования для .NET Framework (приведенные примеры написаны, в основном, на C# и C++):
   http://www.firststeps.ru/dotnet
7. Несколько русскоязычных сайтов, где имеются статьи, примеры кода, доски обсуждений по Visual Basic (разных версий):
   http://www.vbstreets.ru/
   http://www.codenet.ru/progr/vbasic
   http://www.vbnet.ru/
8. Подборка примеров кода на Visual Basic и других языках программирования:
   http://www.freevbcode.com/

Приложение 1

Реализация объектно-ориентированного программирования в Visual Basic .NET

Объекты (`Objects`). Основной единицей в объектно-ориентированном программировании является программный объект, который объединяет в себе как описывающие его данные (свойства), так и средства обработки этих данных (методы). Если говорить образно, то объекты — это существительные, свойства объекта — это прилагательные, а методы объекта — это глаголы.

Объекты являются частью приложения и представляют собой своеобразные «строительные блоки», из которых приложение создается. И даже более того, само приложение в целом является объектом.

Программные объекты обладают **свойствами**, могут использовать **методы** и реагируют на **события**.

Свойства объектов (`Properties`). Каждый объект обладает определенным набором свойств, первоначальные значения которых можно установить с использованием диалогового окна системы программирования.

Значения свойств объектов можно изменять в программном коде. Для присваивания свойству объекта нового значения в левой части строки программного кода необходимо указать имя объекта и затем — название свойства, которые в соответствии с правилами точечной нотации разделяются между собой точкой. В правой части строки необходимо записать конкретное значение свойства:

```
Объект.Свойство = ЗначениеСвойства
```

Методы объектов (`Methods`). Для того чтобы объект выполнил какую-либо операцию, необходимо применить метод, которым он обладает. Многие методы имеют аргументы, которые позволяют задать параметры выполняемых действий. Для присваивания аргументам конкретных зна-

чений используется двоеточие и знак равенства, а между собой аргументы разделяются запятой.

Обратиться к методу объекта можно также с использованием точечной нотации. Чтобы определить, для какого объекта вызывается метод, перед именем метода указывается имя объекта, отделенное точкой:

```
Объект.Метод аргумент1:=значение, аргумент2:=значение
```

События (Events). Событие представляет собой действие, распознаваемое объектом. Событие может создаваться пользователем (например, щелчок мышью или нажатие клавиши) или быть результатом воздействия других программных объектов. В качестве реакции на события вызывается определенная процедура, которая может изменять свойства объекта, вызывать его методы и т. д.

Классы объектов являются «шаблонами», определяющими наборы свойств, методов и событий, по которым создаются объекты. Основные классы объектов представляют объекты, реализующие графический интерфейс проектов. Класс может содержать переменные, свойства, процедуры и события.

Классы могут иметь взаимоотношения друг с другом, наиболее известные из них — это:

- наследование — когда класс, унаследованный от базового класса, получает все данные и код реализации наследуемого класса;
- включение — когда один класс содержит в себе переменную, указывающую на другой класс, и использует ее для вызова членов этого класса.

Для определения класса используется следующая конструкция:

```
Class ИмяКласса
    [Inherits ИмяБазовогоКласса]
    [Implements ИменаИнтерфейсов]
    [СписокЧленовКласса]
End Class
```

`ИмяБазовогоКласса` задает имя базового класса, от которого унаследован данный. Если это имя не указано, то используется класс `System.Object`. `ИменаИнтерфейсов` — это перечисление через запятую

имен интерфейсов[1], которые реализует данный класс. СписокЧленовКласса задает список переменных, свойств, процедур или событий, определяемых в классе.

Каждый из членов класса может иметь один из модификаторов доступа, задающий права доступа к соответствующему члену:

- Public — член класса доступен из любого места приложения;
- Protected — член класса доступен только в классе-потомке, который его унаследовал;
- Friend — член класса доступен только в программном модуле (сборке), в котором он определен;
- Private — член класса доступен только внутри определения класса и может быть использован только из других процедур того же самого класса;
- Protected Friend — член класса доступен либо только в программном модуле (сборке), в котором он определен, либо в классе-потомке, который его унаследовал.

Объект, созданный по «шаблону» класса объектов, является *экземпляром класса* и *наследует* весь набор свойств, методов и событий данного класса. Каждый экземпляр класса имеет уникальное для данного класса имя. Различные экземпляры класса обладают одинаковым набором свойств, однако значения свойств у них могут различаться.

В случае если экземпляр класса является *визуальным объектом* (widget) и предназначен для помещения на форму приложения, то он имеет дополнительный набор свойств, методов и событий, позволяющих визуальному объекту отображать себя на экране внутри формы и взаимодействовать с пользователем (например, реагировать на щелчок мышью).

Для получения программного объекта необходимо создать экземпляр класса, при этом количество создаваемых экземпляров не ограничено. Для создания объекта используется одна из следующих конструкций:

1. **Dim** ИмяПеременной **As** ИмяКласса

В данном случае мы объявляем переменную ИмяПеременной, имеющую тип, заданный как ИмяКласса. При этом экземпляр объекта не создается, поэтому в дальнейшем его необходимо создать при помощи ключевого слова New:

---

[1] То есть способов взаимодействия с другими программными объектами.

```
ИмяПеременной = New ИмяКласса
```

2. `Dim ИмяПеременной As New ИмяКласса`

В этом случае мы не только объявляем переменную `ИмяПеременной`, имеющую тип, заданный как `ИмяКласса`, но и сразу создаем на основе этого класса экземпляр объекта, содержащий все необходимые данные и готовый к использованию.

Для использования определенных в классе переменных, свойств, процедур или событий используется следующая конструкция:

```
ИмяПеременной.ИмяЧленаКласса
```

`ИмяЧленаКласса` — это название переменной, свойства, процедуры или события, определенного в классе. Если при этом вызывается процедура, то ей в списке параметров передаются необходимые для вызова данные.

Графический интерфейс. Системы объектно-ориентированного программирования позволяют визуализировать процесс создания графического интерфейса разрабатываемого проекта, т. е. позволяют создавать объекты и устанавливать значения свойств с помощью диалоговых окон системы программирования.

Графический интерфейс необходим для реализации интерактивного диалога пользователя с работающим проектом. Основой для создания графического интерфейса разрабатываемого проекта является *форма*, представляющая собой окно, на котором размещаются элементы управления. Необходимо отметить, что графический интерфейс проекта может включать в себя несколько форм.

Форма — это объект, представляющий собой окно на экране, в котором размещаются элементы управления.

Визуальное конструирование графического интерфейса проекта состоит в том, что на форму с помощью мыши помещаются и «рисуются» те или иные *элементы управления*.

Классы элементов управления имеют различное назначение в графическом интерфейсе проекта. Текстовые поля, надписи и списки обычно используются для ввода и вывода данных, графические окна — для вывода графики, командные кнопки, переключатели и флажки — для организации диалога и т. д.

На форму может быть помещено несколько экземпляров одного класса элементов управления. Например, несколько кнопок, каждая из которых обладает индивидуальными значениями свойств (надпись, размеры и др.).

Элементы управления — это объекты, являющиеся элементами графического интерфейса проекта и реагирующие на события, производимые пользователем или другими программными объектами.

Форма и элементы управления обладают определенными наборами свойств, методов и событий.

Методы обработки событий. Для каждого события можно запрограммировать отклик, т. е. реакцию объекта на произошедшее событие. Если пользователь производит какое-либо воздействие на элемент графического интерфейса (например, щелчок), в качестве отклика выполняется некоторая последовательность действий (*метод обработки событий*).

Каждый метод обработки события представляет собой отдельный программный модуль, который реализует определенный алгоритм. Создание программного кода метода производится с использованием алгоритмических структур различных типов (линейная, ветвление, выбор и цикл).

Метод обработки событий представляет собой код, который начинает выполняться после реализации определенного события.

Общие процедуры. При разработке сложного алгоритма целесообразно стараться выделить в нем последовательности действий, которые выполняют решение каких-либо подзадач и могут многократно вызываться из основного алгоритма. Такие алгоритмы называются *вспомогательными* и в алгоритмических языках программирования реализуются в форме *подпрограмм*, которые вызываются из основной программы.

В объектно-ориентированных языках программирования вспомогательные алгоритмы реализуются с помощью *общих процедур*. Общие процедуры создаются в тех случаях, когда в программном модуле можно выделить многократно повторяющиеся последовательности действий (алгоритмы).

Запуск общих процедур не связывается с какими-либо событиями, а реализуется путем вызова из других процедур.

Общая процедура представляет собой подпрограмму, которая начинает выполняться после ее вызова из другой процедуры.

Общая процедура в языке Visual Basic .NET может быть определена двумя способами:

1. При помощи ключевого слова `Shared` при объявлении процедуры в теле класса:

```
Shared Sub ИмяПроцедуры(СписокПараметров)
  Программный код
end Sub
```

2. Объявлением процедуры в теле так называемого модуля (`Module`):

```
Module ИмяМодуля
  Sub ИмяПроцедуры(СписокПараметров)
  Программный код
  end Sub
End Module
```

Во втором случае количество определяемых модулей не ограничено, каждая процедура, определенная в рамках модуля, является общей.

Оба описанных варианта могут быть использованы как для определения общих процедур, не возвращающих значения (для этого используется ключевое слово `Sub`), так и для общих процедур, возвращающих значение (для этого используется ключевое слово `Function`).

В рамках модуля или класса общей процедуре дается уникальное название — имя процедуры и устанавливается список входных и выходных параметров процедуры. При этом названия процедур в разных модулях или классах могут быть одинаковыми.

Список входных параметров представляет собой набор переменных, значение которых должно быть установлено до начала выполнения процедуры.

Список выходных параметров представляет собой набор переменных, значение которых должно быть установлено после окончания выполнения процедуры.

Общая процедура может быть вызвана на выполнение либо по имени, либо с помощью оператора `Call`. По умолчанию рекомендуется, как более простой, первый вариант, однако второй также является возможным, прежде всего для сохранения совместимости с предыдущими версиями языка Visual Basic. Использование оператора `Call` повышает наглядность исходного кода. Вызов процедуры по имени выглядит следующим образом:

```
ИмяПроцедуры(СписокПараметров)
```

А при использовании `Call` так:

```
Call ИмяПроцедуры(СписокПараметров)
```

Следует также помнить, что при использовании второго способа для вызова общих процедур, возвращающих значение (определение, использующее `Function`), возвращенное значение игнорируется и не может быть использовано как результат вызова общей процедуры.

Проект (Project). Проект является объединением файлов, содержащих описание визуального интерфейса, и программный код. Файлы, входящие в состав проекта, обрабатываются компилятором Visual Basic .NET и преобразовываются в приложения, которые затем могут выполняться в среде .NET операционной системой Windows.

Приложение 2

Справочник по языку Visual Basic .NET

1. Переменные в языке программирования Visual Basic .NET

Тип переменной. Тип переменной определяется набором допустимых значений (данных) переменной и допустимыми операциями под этими значениями. Значениями переменных числовых типов (`Byte`, `Short`, `Integer`, `Long`, `Single`, `Double`, `Decimal`) являются числа, логического типа `Boolean` — «истина» (`True`) или «ложь» (`False`), строкового типа `Char` — один символ и `String` — последовательность символов, типа даты (`Date`) — дата. Обозначения типов переменных являются ключевыми словами языка и поэтому выделяются.

Различные типы данных требуют для своего хранения в оперативной памяти компьютера различное количество ячеек (байтов):

Тип переменной	Возможные значения	Объем занимаемой памяти
Byte	Целые неотрицательные числа от 0 до 255	1 байт
Short	Целые числа от −32 768 до 32 767	2 байта
Integer	Целые числа от −2 147 483 648 до 2 147 483 647	4 байта
Long	Целые числа от −9 223 372 036 854 до 9 223 372 036 853	8 байтов
Decimal	Целые десятичные числа со знаком от 10^{-28} до 10^{28}	16 байтов
Single	Десятичные числа одинарной точности (7–8 значащих цифр) от $-1{,}4 \cdot 10^{-45}$ до $3{,}4 \cdot 10^{38}$	4 байта
Double	Десятичные числа двойной точности (15–16 значащих цифр) от $-5.0 \cdot 10^{-324}$ до $1{,}7 \cdot 10^{308}$	8 байтов
Boolean	Логическое значение **True** или **False**	2 байта
String	Строка символов в кодировке Unicode	2 байта на символ
Date	Даты от 1 января 0001 года до 31 декабря 9999 года и время от 0:00:00 до 23:59:59	8 байтов

Имя переменной. Имя каждой переменной (идентификатор) уникально и не может меняться в процессе выполнения программы. Имя переменной:

- должно начинаться с буквенного символа или с подчеркивания «_»;
- может содержать только буквенные символы, десятичные цифры и подчеркивания;
- должно содержать, по крайней мере, один буквенный или цифровой символ, если оно начинается с подчеркивания;
- не должно составлять более чем 1023 знака.

Объявление переменной. Важно, чтобы исполнитель программы (компьютер) «понимал», переменные какого типа используются в программе. Для объявления переменной используется оператор определения переменной. Синтаксис (правило записи) этого оператора следующий:

```
Dim ИмяПеременной As ТипПеременной = Значение
```

С помощью одного оператора можно объявить сразу несколько переменных, например:

```
Dim Число As Integer, Строка As String
```

Величины, значения которых не меняются в процессе выполнения программы, называются *константами*. Синтаксис объявления констант следующий:

```
Const ИмяКонстанты As Тип = ЗначениеКонстанты
```

Присваивание переменным значений. Переменная может получить или изменить значение с помощью *оператора присваивания*. Синтаксис этого оператора следующий:

```
ИмяПеременной = Выражение
```

При выполнении оператора присваивания переменная, имя которой указано слева от знака равенства, получает значение, равное значению

выражения (арифметического, строкового или логического), которое находится справа от знака равенства.

2. Массивы в языке программирования Visual Basic

Типы массивов и объявление массива. Массив является набором однотипных переменных, объединенных одним именем. Массивы бывают одномерные, которые можно представить в форме таблицы из одного столбца, двумерные, которые можно представить в форме обычной таблицы, и многомерные.

Массив состоит из пронумерованной последовательности элементов. Номера в этой последовательности определяются индексом, который может принимать целочисленные значения. Каждый из этих элементов является переменной, т. е. обладает именем и значением, и поэтому массив можно назвать последовательностью *переменных с индексом*.

Массивы могут быть различных типов: числовые, строковые и т. д.

Объявление массива производится аналогично объявлению переменных, необходимо только дополнительно указать диапазон изменения индекса. Например, объявление одномерного целочисленного массива, содержащего 10 элементов, производится следующим образом:

```
Dim A() As Byte
Dim A() As Integer = {0,1,2}
```

Обращение к элементу массива производится по его имени, состоящему из имени массива и значения индекса, например: `A(5)`.

Заполнение массива случайными числами. Для начала работы с массивом необходимо его предварительно заполнить, т. е. присвоить элементам массива определенные значения. Заполним числовой массив `A(I)` целыми случайными числами в интервале от 1 до 100.

Для генерации последовательности случайных чисел используем функцию `Rnd()`. При запуске программы функция `Rnd()` дает равномерно распределенную псевдослучайную (т. е. каждый раз повторяющуюся) последовательность чисел в интервале $0 \le X < 1$.

Для получения последовательности случайных чисел в заданном интервале $A \le X < B$ необходимо использовать следующую формулу:

```
(B-A)*Rnd() + A
```

Тогда получение целочисленной последовательности случайных чисел на интервале $1 \le X < 100$ достигается использованием функции выделения целой части числа:

```
CInt(Int(100*Rnd()) + 1)
```

В итоге получим:

```
For I = 0 To 9
    A(I) = CInt(Int(100*Rnd()) + 1)
Next I
```

Для генерации различающихся между собой последовательностей случайных чисел рекомендуется использовать оператор `Randomize()`.

3. Основные типы алгоритмических структур и их кодирование на языке Visual Basic

3.1. Алгоритмическая структура «ветвление»

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в алгоритмические структуры «*ветвление*» входит условие, в зависимости от истинности или ложности которого реализуется та или иная последовательность команд (серий).

В алгоритмической структуре **«ветвление»** та или иная серия команд выполняется в зависимости от истинности **условия**.

Условие. В простом условии два числа, две строки, две переменные, два арифметических, строковых или логических выражения сравниваются между собой с использованием операций сравнения (>, <, =, >=, <=). Например: `5 > 3`, `"A" = "B"` и т. д. В зависимости от результата сравнения условие принимает значение `True` (истина) или `False` (ложь).

Сложное условие — это последовательность простых условий, объединенных между собой знаками логических операций. Например, `X > 3` **`And`** `Y = 4×4`.

При определении условий как в качестве составной части сложного условия, так и при задании просто условия можно напрямую использовать переменную типа `Boolean`. В этом случае при вычислении результата сложного условия значение переменной проверяется на результат `True/False` так же, как если бы оно представляло собой простое условие.

Инструкция `If-Then-Else` (рис. 5). На языке программирования Visual Basic после первого ключевого слова `If` должно быть размещено условие. После второго ключевого слова `Then` — последовательность команд (Серия 1), которая должна выполняться, если условие принимает

значение «истина». После третьего ключевого слова `Else` размещается последовательность команд (Серия 2), которая должна выполняться, если условие принимает значение «ложь».

Оператор условного перехода может быть записан в многострочной форме или в однострочной форме.

Блок-схема	Язык программирования Visual Basic .NET

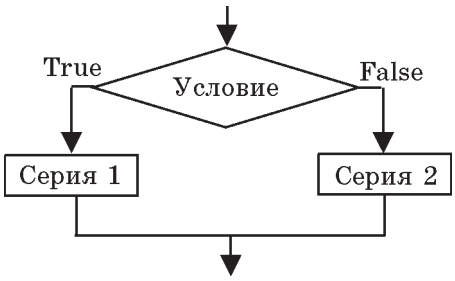
	**If** Условие **Then**
	  Серия 1
	[**Else**
	  Серия 2]
	End If
	
	If Условие _
	Then Серия 1 _
	[**Else** Серия 2]

Рис. 5

В многострочной форме он записывается с помощью инструкции `If ... Then ... Else ... End If` (Если ... То ... Иначе ... Конец Если). В этом случае второе ключевое слово `Then` размещается в той же строке, что и условие, а последовательность команд (Серия 1) — в следующей строке. Третье ключевое слово `Else` размещается в третьей строке, а последовательность команд (Серия 2) — в четвертой. Конец инструкции ветвления `End If` размещается в пятой строке.

В однострочной форме он записывается с помощью инструкции `If ... Then ... Else ...` (Если ... То ... Иначе ...). Если инструкция не помещается в одной строке, она может быть разбита на несколько строк. Такое представление инструкций более наглядно для человека. Компьютер же должен знать, что разбитая на строки инструкция представляет единое целое. Это обеспечивает знак «переноса», который задается символом подчеркивания после пробела « _».

Третье ключевое слово `Else` (Серия 2) может отсутствовать. (Необязательные части оператора мы здесь помечаем квадратными скобками.) Тогда, в случае если условие ложно, выполнение оператора условного перехода заканчивается и выполняется следующая строка программы.

Кроме того, для варианта записи в виде многострочной формы возможно многократное дублирование условия, указываемого после инструкции `If`, для этого используется следующая форма:

```
If Условие1 Then
  Серия 1
[ElseIf Условие2 Then
  Серия 2]
...
[ElseIf УсловиеN Then
  Серия N]
[Else
  Серия N + 1]
End If
```

В данном случае переход к следующему условию `ElseIf` происходит, только если предыдущее условие вернуло `False`. Если ни одно из условий `ElseIf` не вернуло `True`, управление, так же, как в простой многострочной форме, передается конструкции `Else`.

3.2. Алгоритмическая структура «выбор»

Алгоритмическая структура «*выбор*» применяется для реализации ветвлений со многими вариантами серий команд. В структуру выбора входят несколько условий, проверка которых осуществляется в последовательности их записи в структуре выбора. При истинности одного из условий (Условие 1, Условие 2 и т. д.) выполняется соответствующая последовательность команд (Серия 1, Серия 2 и т. д.). Если ни одно из условий не истинно, то будет выполнена последовательность команд Серия.

В алгоритмической структуре **«выбор»** выполняется одна из нескольких последовательностей команд при истинности соответствующего **условия**.

Инструкция `Select Case` (рис. 6). На языке программирования инструкция выбора начинается с ключевых слов `Select Case`, после которых записывается выражение (переменная или арифметическое выражение). После ключевых слов `Case` записываются значения, с которыми сравнивается заданное выражение (проверяется истинность условий). При истинности одного из условий начинает выполняться соответствующая серия

команд. Если ни одно из условий не истинно, то будет выполнена серия команд после ключевых слов `Case Else`. Заканчивается инструкция ключевыми словами `End Select`.

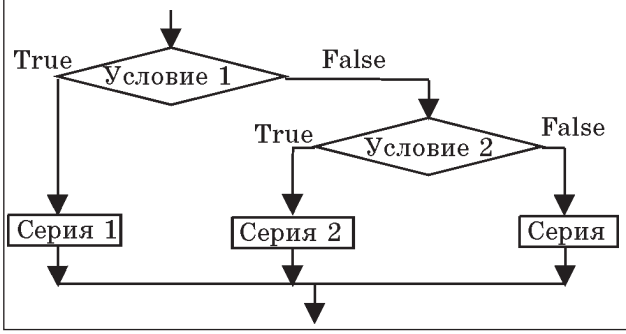
Блок-схема	Язык программирования Visual Basic .NET
	**Select Case** Выражение **Case** Условие1 Серия 1 **Case** Условие2 Серия 2 [**Case Else** Серия] **End Select**

Рис. 6

Следует также отметить, что количество блоков, использующих ключевое слово `Case`, не ограничено. Блок `Case Else` может отсутствовать. В качестве условия, заданного после ключевого слова `Case`, может быть использован один из двух вариантов:

1. Когда указывается диапазон значений, для этого используется следующая конструкция:

   `Значение1` **`To`** `Значение2`

   При этом `Значение1` обязательно должно быть меньше или равно `Значению2`.

2. Когда необходимо использовать вычисленное в конструкции `Select Case` выражение в условии, используя один из операторов сравнения (=, <>, <, <=, >, or >=):

   `Is` `ОператорСравнения УсловноеВыражение`

   В качестве `УсловногоВыражения` может быть использовано как просто значение, с которым мы производим сравнение, так и переменная, содержащее это значение. Также допускается конструировать сложные условные выражения. Ключевое слово `Is` при этом указывает на выражение, вычисленное в конструкции `Select Case`.

3.3. Алгоритмическая структура «цикл»

В алгоритмическую структуру «*цикл*» входит серия команд, выполняемая многократно. Такая последовательность команд называется *телом цикла*.

В алгоритмической структуре **«цикл»** серия команд (**тело цикла**) выполняется многократно.

Циклические алгоритмические структуры бывают двух типов:

- *циклы со счетчиком*, в которых тело цикла выполняется определенное количество раз;
- *циклы для обработки массивов или коллекций*, тело цикла при этом выполняется для каждого элемента массива или коллекции;
- *циклы по условию*, в которых тело цикла выполняется, пока условие истинно (или до тех пор, пока условие не станет истинным).

Цикл со счетчиком (рис. 7). Когда заранее известно, какое число повторений тела цикла необходимо выполнить, можно воспользоваться циклической инструкцией (оператором цикла со счетчиком) `For ... Next`.

Блок-схема	Язык программирования Visual Basic .NET

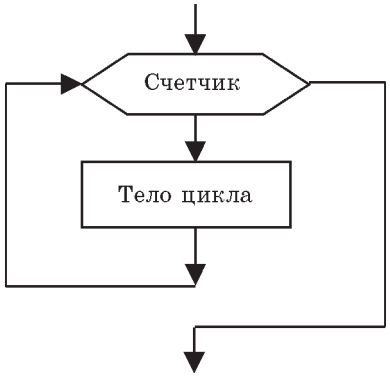
	`For Счетчик=НачЗнач`
	`To КонЗнач [Step шаг]`
	`  Тело цикла`
	`Next [Счетчик]`

Рис. 7

Синтаксис оператора `For ... Next` следующий: строка, начинающаяся с ключевого слова `For`, является заголовком цикла, а строка с ключевым словом `Next` — концом цикла, между ними располагаются операторы, являющиеся телом цикла.

В начале выполнения цикла значение переменной `Счетчик` устанавливается равным `НачЗнач`. При каждом «проходе» цикла (выполнении тела цикла) переменная `Счетчик` изменяется (увеличивается или уменьшается) на величину шага. Если она достигает величины `КонЗнач`, то цикл завершается и выполняются следующие за ним операторы.

Циклы для обработки массивов или коллекций (рис. 8). Этот вариант напоминает цикл со счетчиком; в данном случае известно, какое число повторений тела цикла необходимо выполнить, причем это число задается размером массива или коллекции, а в теле происходит обработка конкретного текущего элемента. При этом для определения используется инструкция `For Each ... Next`.

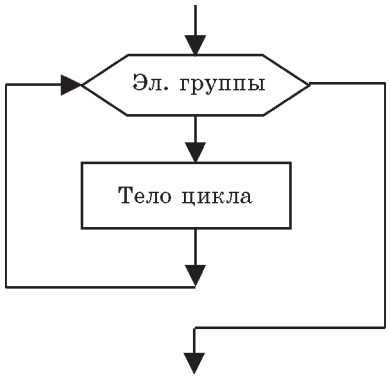
Блок-схема	Язык программирования Visual Basic .NET
	**For Each** Элемент **[As** ТипЭлемента**] In** Группа Тело цикла **Next** [Элемент]

Рис. 8

Основное различие между операторами `For ... Next` и `For Each ... Next` состоит в том, что после `For Each` находится описание переменной, в которую помещается элемент коллекции, указанной после ключевого слова `In`. При этом для данной переменной можно задать тип данных, к которому будет приведен текущий элемент коллекции.

Вообще говоря, использование оператора `For Each ... Next` не ограничивается только массивами и коллекциями, любой класс, реализую-

щий интерфейс `System.Collections.IEnumerable`, может быть обработан при помощи `For Each ... Next`.

Цикл с предусловием (рис. 9). Часто бывает так, что необходимо повторить тело цикла, но заранее неизвестно, какое количество раз это надо сделать. В таких случаях количество повторений зависит от некоторого условия. Цикл с предусловием никогда не выполняется в случае ложности условия.

Цикл называется **циклом с предусловием**, если условие выхода из цикла стоит в начале, перед телом цикла.

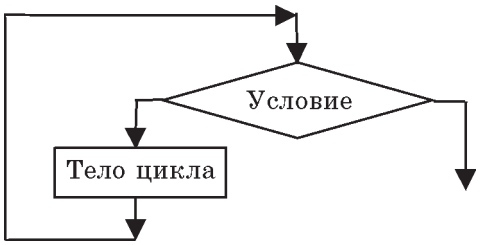
Блок-схема	Язык программирования Visual Basic
	**Do While** Условие Тело цикла **Loop** **Do Until** Условие Тело цикла **Loop**

Рис. 9

Цикл с предусловием реализуется с помощью инструкций `Do While ... Loop` и `Do Until ... Loop`.

Проверка истинности условия выхода из цикла проводится с помощью ключевых слов `While` или `Until`. Эти слова придают одному и тому же условию противоположный смысл. Ключевое слово `While` обеспечивает выполнение цикла, пока условие имеет значение «истина». Как только условие примет значение «ложь», выполнение цикла закончится. В этом случае условие является условием продолжения цикла.

Ключевое слово `Until` обеспечивает выполнение цикла, пока условие имеет значение «ложь». Как только условие примет значение «истина», выполнение цикла закончится. В этом случае условие является условием завершения цикла.

Цикл с постусловием (рис. 10). Цикл с постусловием выполняется обязательно как минимум один раз, независимо от того, выполняется условие или нет.

Цикл называется **циклом с постусловием**, если условие выхода из цикла стоит в конце, после тела цикла.

Цикл с постусловием реализуется с помощью инструкций `Do ... Loop While` и `Do ... Loop Until`. Проверка истинности условия выхода из цикла проводится с помощью ключевых слов `While` или `Until`.

Блок-схема	Язык программирования Visual Basic

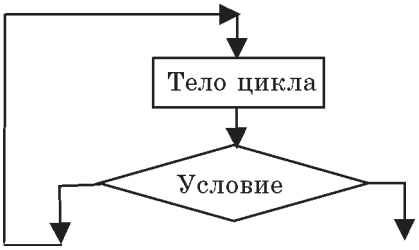
	**Do**
	  Тело цикла
	Loop While Условие
	
	Do
	  Тело цикла
	Loop Until Условие

Рис. 10

В любом из описанных типов циклов программист может организовать управление структурой исполнения цикла при помощи одной из следующих, соответствующих типу цикла, конструкций.

- `Exit For`
- `Exit While`
- `Exit Do`

При этом происходит автоматический выход из цикла и управление передается следующей за циклом команде.

- `Continue For`
- `Continue While`
- `Continue Do`

При этом происходит переход к следующей итерации, элементу массива или коллекции, обрабатываемой в теле цикла. Если при этом один цикл вложен в другой, то область действия `Exit` и `Continue` действует только на вложенный цикл, из которого данная команда вызывается.

С точки зрения оптимизации работы циклов рекомендуется использовать переменные типов `Integer` и `UInteger`, использование типов данных `Short`, `Long`, `UShort`, `ULong` не является эффективным, а использование типа `Decimal` ведет к существенному снижению скорости.

4. Функции в языке программирования Visual Basic .NET

4.1. Функции преобразования типов данных

Функции преобразования реализуют преобразование данных из одного типа в другой.

Функция `Val()`. Часто необходимо преобразовать строковое значение в числовое. Это можно сделать с помощью функции Val(), аргументом которой является строка, а значением — число:

```
Val(Строка)
```

Например, функция `Val()` применяется при вводе чисел в текстовые поля для преобразования строкового значения свойства `Text` в число, которое затем используется в вычислениях.

Строковое выражение, являющееся аргументом функции `Val()`, может быть задано не только в десятичной, но также в восьмеричной (приставка `"&O"`) и шестнадцатеричной (приставка `"&H"`) системах счисления. Например, значением функций `Val("&O3720")` и `Val("&H7D0")` является десятичное число 2000.

Таким образом, появляется возможность перевода чисел, выраженных в строковой форме, из восьмеричной и шестнадцатеричной систем счисления в десятичную систему счисления.

Функции `Str()`, `Oct()` и `Hex()`. Функции `Str()`, `Oct()` и `Hex()` позволяют производить преобразование десятичных чисел соответственно в десятичные, восьмеричные и шестнадцатеричные числа в строковой форме. Аргументом функции является десятичное число, а значением — строка:

```
Str(Число)
Oct(Число)
Hex(Число)
```

Например, значениями функций `Str(2000)`, `Oct(2000)`, `Hex(2000)` являются десятичное число `2000`, восьмеричное число `3720` и шестнадцатеричное число `7D0` в строковой форме.

4.2. Математические функции

В математических функциях значениями как аргументов, так и функций являются числа. В языке программирования Visual Basic 2005 математические функции реализуются с использованием математических методов:

Функция	Аргумент функции x	Возвращаемое функцией значение
`Math.Sin(x)`	Число (в радианах)	Синус числа
`Math.Cos(x)`	Число (в радианах)	Косинус числа
`Math.Tan(x)`	Число (в радианах)	Тангенс числа
`Math.Atan(x)`	Число	Арктангенс в радианах
`Math.Sqrt(x)`	Неотрицательное число	Квадратный корень из числа
`Math.Log(x)`	Число	Натуральный логарифм числа
`Math.Exp(x)`	Число	Экспонента числа
`Rnd()`	Нет аргумента	Псевдослучайное число N ($0 < N < 1$)
`Math.Round(x)`	Число	Число, ближайшее по величине к заданному числу
`Math.Abs(x)`	Число	Модуль числа
`Math.Sign(x)`	Число	Знак числа

4.3. Строковые функции

В строковых функциях либо аргументы, либо возвращаемые функциями значения являются строками.

Конкатенация строк. Над переменными и строками может производиться *операция конкатенации*. Эта операция состоит в объединении строк или значений строковых переменных в единую строку. Операция конкатенации обозначается знаком «+», который не следует путать со знаком сложения чисел в арифметических выражениях.

Функция определения длины строки. В функции определения длины строки `Len(Строка)` аргументом является строка `Строка`, а возвращает функция числовое значение длины строки (количество символов в строке). Синтаксис функции:

```
Len(Строка)
```

Пусть аргументом функции `Len()` будет строка `"информатика"`, тогда в результате выполнения оператора присваивания

```
ДлинаСтроки = Len("информатика")
```

значением целочисленной переменной `ДлинаСтроки` будет число 11.

Функции вырезания подстроки. В функциях вырезания подстроки (части строки) `Left(Строка, Длина)`, `Right(Строка, Длина)` и `Mid(Строка, Позиция, Длина)` аргументами являются строка `Строка` и числа или целочисленные переменные `Длина` и `Позиция`. Функции возвращают строковое значение, равное вырезанной подстроке. Синтаксис функций:

```
Left(Строка, Длина)
Right(Строка, Длина)
Mid(Строка, Позиция, Длина)
```

Значением функции `Left()` является левая подстрока, которая начинается с крайнего левого символа строки и имеет количество символов, равное значению числового аргумента `Длина`.

Пусть аргументом функции `Left()` будет строка `"информатика"`, тогда в результате выполнения оператора присваивания

```
ЛеваяПодстрока = Left("информатика",2)
```

значением строковой переменной `ЛеваяПодстрока` будет строка `"ин"`.

Значением функции `Right` является правая подстрока, которая начинается с крайнего правого символа строки и имеет количество символов, равное значению числового аргумента `Длина`.

Пусть аргументом функции `Right` будет строка `"информатика"`, тогда в результате выполнения оператора присваивания

```
ПраваяПодстрока = Right("информатика",4)
```

значением строковой переменной `ПраваяПодстрока` будет строка `"тика"`.

Значением функции Mid является подстрока, которая начинается с символа, позиция которого в строке задается числовым аргументом Позиция, и имеет длину, равную значению числового аргумента Длина.

Пусть аргументом функции Mid будет строка "информатика", тогда в результате выполнения оператора присваивания

```
Подстрока = Mid("информатика",3,5)
```

значением строковой переменной Подстрока будет строка "форма".

Функция Asc(). Функция Asc() осуществляет преобразование строки в числовой код первого символа. Аргументом функции является строка, а значением — число:

```
Asc(Строка)
```

Для обработки данных в формате Unicode рекомендуется использовать специализированную функцию AscW.

Функция Chr(). Функция Chr() осуществляет преобразование числового кода в символ. Аргументом функции является число, а значением — символ:

```
Chr(Число)
```

Для преобразования данных в формате Unicode рекомендуется использовать специализированную функцию ChrW.

Функция InStr(). Функция InStr() осуществляет поиск подстроки в строке. Аргументами функции являются строка и подстрока, а значением — номер первого символа подстроки в строке:

```
InStr(Строка,Подстрока)
InStr(Старт,Строка,Подстрока)
```

Второй вариант имеет дополнительный параметр, позволяющий указать смещение в строке, с которого начинается поиск.

Строковые функции и их значения:

Функция	Значение функции
Len("информатика")	11
Left("Килобайт",4)	"Кило"
Right("Килобайт",4)	"байт"
Mid("информатика",3,5)	"форма"
Asc("и")	232
Chr(255)	"я"
InStr("информатика", "форма")	3

4.4. Функции ввода и вывода данных

Функция InputBox(). Функция InputBox() позволяет вводить данные с помощью диалогового окна ввода. Аргументами этой функции являются три строки, а значением функции — строка по умолчанию или строка, введенная пользователем. Синтаксис функции:

```
Переменная = InputBox("Подсказка", "Заголовок",
["ЗначениеПоУмолчанию"], [КоординатаX], [КоординатаY])
```

В процессе выполнения этой функции появляется диалоговое окно с текстовым полем (рис. 11), в котором:

- в строку заголовка окна выводится значение второго аргумента "Заголовок";
- в само окно выводится значение аргумента "Подсказка";
- в текстовое поле выводится значение аргумента "ЗначениеПоУмолчанию" (если это значение отсутствует, содержимое текстового окна также отсутствует);
- КоординатаX и КоординатаY — смещение от левого верхнего угла экрана.

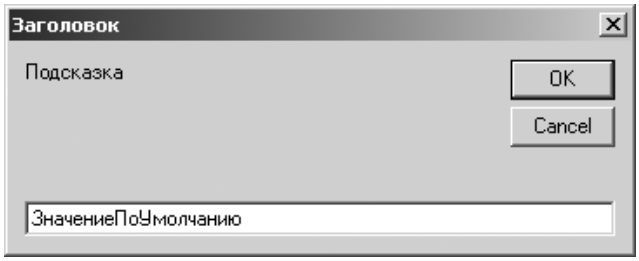


Рис. 11

Если пользователь щелкнет по кнопке *ОК*, то значением функции станет строка, введенная пользователем в текстовое поле. Если пользователь щелкнет по кнопке *Cancel*, то значением функции станет строка `"Значение ПоУмолчанию"`.

Функция `MsgBox()`. Функция `MsgBox()` позволяет выводить сообщения не на форму, а в специальное окно сообщений, в котором можно разместить определенный набор кнопок и информационный значок о типе сообщения. Кроме того, функция `MsgBox()` получает определенное значение, которое может быть присвоено числовой переменной. Синтаксис функции:

```
Переменная = MsgBox("Сообщение"[,ЧисКод1+ЧисКод2]
[,"Заголовок"])
```

В процессе выполнения этой функции появляется диалоговое окно с кнопками, в котором:

- аргумент `"Сообщение"` выводится в окно сообщений;
- аргумент `ЧисКод1+ЧисКод2` определяет внешний вид окна, значение `ЧисКод1` определяет вид пиктограммы, которая помещается на окно сообщений, а значение `ЧисКод2` — набор кнопок, размещаемых в окне;
- аргумент `"Заголовок"` выводится в строку заголовка окна.

Значения ЧисКод1 и ЧисКод2, определяющие вид окна сообщений:

ЧисКод1	Тип сообщения	Пиктограмма
16	Ошибка	
32	Вопрос	
48	Внимание	
64	Информация	

ЧисКод2	Набор кнопок
0	*ОК*
1	*ОК, Отмена*
2	*Стоп, Повторить, Пропустить*
3	*Да, Нет, Отмена*
4	*Да, Нет*
5	*Повторить, Отмена*

С помощью одного числа, являющегося суммой чисел ЧисКод1 и ЧисКод2, можно одновременно установить определенную пиктограмму и определенную комбинацию кнопок, размещаемых в окне сообщений. Например, число 36 можно рассматривать как сумму чисел 32 (код пиктограммы «Вопрос») и 4 (код комбинации кнопок *Да*, *Нет*). В этом случае функция MsgBox() будет выводить окно сообщений с текстом, пиктограммой-знаком вопроса и кнопками *Да*, *Нет*.

Например, если задать функции с аргументами MsgBox("Сообщение", 48 + 3, "Заголовок"), то будет выведено окно сообщений, показанное на рис. 12.

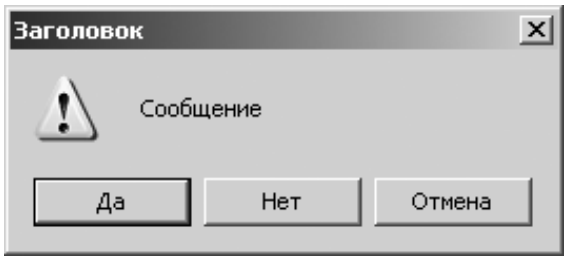


Рис. 12

Нажатие кнопки приводит к вычислению значения функции, которое зависит от нажатой кнопки. То есть значение, возвращаемое функцией MsgBox(), позволяет определить, какая из кнопок была нажата:

Нажатая кнопка	Значения функции `MsgBox()`
ОК	1
Отмена	2
Стоп	3
Повтор	4
Пропустить	5
Да	6
Нет	7

5. Возможности работы с графикой в Visual Basic .NET 2003 и Visual Basic 2005 Express Edition

Графические методы в языках программирования Visual Basic .NET 2003 и Visual Basic 2005 Express Edition существенно отличаются от графических операторов предшествующих версий Visual Basic 5.0 и 6.0.

На форме и управляющих элементах можно рисовать линии, прямоугольники, окружности и другие графические фигуры. Для рисования необходимо определить объекты `Graphics` (Область рисования), `Pen` (Перо) и `Brush` (Кисть).

Область рисования. Объект `Graphics` (Область рисования) позволяет выбрать в качестве области рисования определенный элемент управления и обладает методами рисования графических фигур. Сначала необходимо в разделе объявления переменных определить имя объекта, например:

```
Dim Graph1 As Graphics
```

Затем в программном коде событийной процедуры необходимо указать определенный элемент управления в качестве области рисования. Обычно в качестве области рисования выбирается размещенное на форме графическое поле (например, `PictureBox1`):

```
Graph1 = Me.PictureBox1.CreateGraphics()
```

Перо. Объект `Pen` (Перо) определяет цвет и ширину линии рисования. Сначала необходимо в разделе объявления переменных определить имя объекта (например, `Pen1`), установить цвет (например, красный `Color.Red`) и ширину линии в пикселях (например, 3):

```
Dim Pen1 As New Pen(Color.Red, 3)
```

Затем в программном коде событийной процедуры можно установить новые значения цвета и ширины линии, например:

```
Pen1.Color = Color.Green
Pen1.Width = 15
```

Кисть. Объект `Brush` (Кисть) определяет цвет и стиль закрашивания прямоугольников, окружностей и других замкнутых фигур. Сначала необходимо в разделе объявления переменных определить имя объекта (например, `Brush1`) и установить тип закраски и цвет (например, сплошная закраска синего цвета `SolidBrush(Color.Blue)`):

```
Dim Brush1 As New SolidBrush(Color.Blue)
```

Затем в программном коде событийной процедуры можно установить новый цвет закраски (например, пурпурный):

```
Brush1.Color = Color.Magenta
```

Графические методы. Графические фигуры рисуются с использованием графических методов. Замкнутые фигуры, такие как прямоугольники или эллипсы, состоят из двух частей — из контура и из внутренней области. Контур рисуется с использованием заданного пера, а внутренняя область закрашивается с использованием заданной кисти.

`DrawLine()` — метод рисования линии, аргументами которого являются перо определенного цвета и толщины (например, `Pen1`), а также координаты концов линии `X1`, `Y1` и `X2`, `Y2`:

```
Graph1.DrawLine(Pen1, X1, Y1, X2, Y2)
```

`DrawRectangle()` — метод рисования прямоугольника, аргументами которого являются перо определенного цвета и толщины (например, `Pen1`), а также координаты левого верхнего угла `X1`, `Y1`, ширина `Width` и высота `Height`:

```
Graph1.DrawRectangle(Pen1, X1, Y1, Width, Height)
```

`FillRectangle()` — метод закраски прямоугольника с использованием кисти определенного цвета. Например:

```
Graph1.FillRectangle(Brush1, X1, Y1, Width, Height)
```

`DrawEllipse()` — метод рисования окружности или эллипса, аргументами которого являются перо определенного цвета и толщины (например, `Pen1`), а также координаты левого верхнего угла описанного прямоугольника `X1`, `Y1`, ширина `Width` и высота `Height`:

```
Graph1.DrawEllipse(Pen1, X1, Y1, Width, Height)
```

`Graph1.FillEllipse()` — метод закраски окружности или эллипса с использованием кисти определенного цвета. Например:

```
Graph1.FillEllipse(Brush1, X1, Y1, Width, Height)
```

Для рисования точки с заданными координатами `X1` и `Y1` можно использовать методы `DrawRectangle(Pen1, X1, Y1, 1, 1)` или `DrawEllipse(Pen1, X1, Y1, 1, 1)`, в которых аргументы `Width` и `Height` равны 1.

`Graph1.Clear()` — метод, заданным цветом стирающий изображения в области рисования. Например, белым цветом:

```
Graph1.Clear(Color.White)
```

Цвет. Цвет устанавливается как значение свойства `Color`. Можно установить цвет с использованием нескольких десятков цветовых констант. Ниже приведены примеры установки зеленого цвета для объекта `Pen1` (Перо) и желтого цвета для объекта `Brush1` (Кисть):

```
Pen1.Color = Color.Green
Brush1.Color = Color.Yellow
```

Для установки цвета в 24-битовой палитре цветов RGB используется метод `Color.FromArgb(Red, Green, Blue)`, аргументами которого являются три числа в диапазонах от 0 до 255 (интенсивности красного, зеленого и синего цветов). Например, так можно установить пурпурный цвет для объекта `Brush1` (Кисть):

```
Brush1.Color = Color.FromArgb(255,0,255)
```

Рисование текста. Метод `DrawString()` позволяет выводить текст в область рисования. Аргументами метода является строка текста, шрифт,

кисть и координаты начала строки. Объекты «Шрифт» (например, `drawFont`) и «Кисть» (например, `drawBrush`) необходимо объявить:

```
Dim drawFont As New Font("Arial", 16)
Dim drawBrush As New SolidBrush(Color.Black)
```

Рисование текста в поле рисования можно осуществить так:

```
Graph1.DrawString("Текст", drawFont, drawBrush, 10, 10)
```

Система координат. Рисование линий, прямоугольников и других фигур производится в компьютерной системе координат, начало которой расположено в верхнем левом углу формы или элемента управления. Ось X направлена вправо, а ось Y — вниз. Единицей измерения по умолчанию является точка (пиксель). Компьютерная система координат графического поля шириной 300 точек и высотой 200 точек приведена на рис. 13.

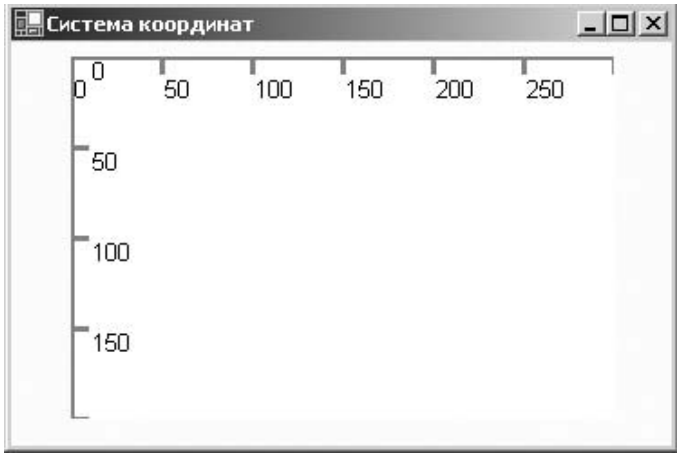


Рис. 13

При геометрических построениях и построении графиков функций удобнее использовать математическую систему координат, начало которой обычно находится в центре области рисования. Ось X направлена вправо, а ось Y — вверх. Математическая система координат графического поля шириной 300 точек и высотой 200 точек приведена на рис. 14.

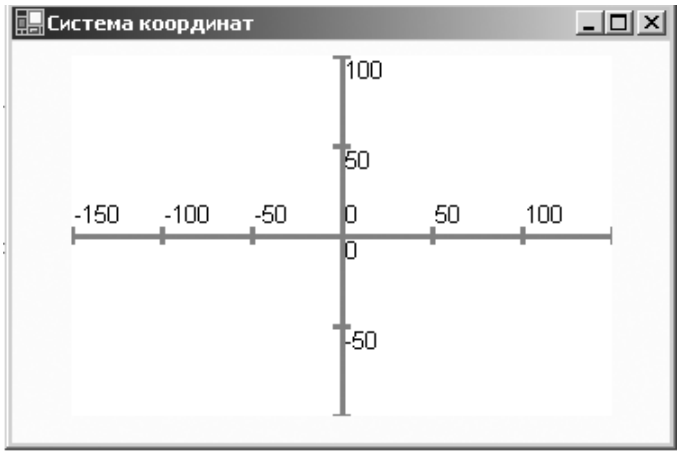


Рис. 14

Для преобразования компьютерной системы координат в математическую систему координат используется метод масштабирования и поворота осей `ScaleTransform()` и метод сдвига начала координат `TranslateTransform()`.

Метод `Graph1.ScaleTransform(1, -1)` обеспечивает поворот оси *Y*.

Метод `Graph1.TranslateTransform(150, -100)` обеспечивает сдвиг по оси *X* на 150 точек вправо и сдвиг по оси *Y* на 100 точек вниз.

В системе координат можно также изменять единицу измерения путем установки значения свойства `PageUnit`.

Например, так устанавливается единица измерения, равная 1/300 дюйма:

```
Graph1.PageUnit = GraphicsUnit.Document
```

А так устанавливается единица измерения, равная 1 миллиметру:

```
Graph1.PageUnit = GraphicsUnit.Millimeter
```

Приложение 3

Процедура установки операционной системы Windows XP Professional с загружаемого дистрибутивного компакт-диска[1]

Подготовка файла ответов для автоматизации установки

Чтобы выполнить эту процедуру в полностью автоматическом режиме, нужно подготовить специальный текстовый файл (называемый файлом ответов), который и использовать в дальнейшем. Для этого воспользуйтесь программой Диспетчер установки Windows. Она находится в архивном файле DEPLOY.CAB на дистрибутивном компакт-диске, поэтому сначала необходимо извлечь файлы из этого архива следующим образом:

1. На любом компьютере с операционной системой Windows XP (или Windows 2000) вставьте дистрибутивный компакт-диск Windows XP Professional в дисковод.
2. В открывшемся окне программы установки[2] щелкните сначала по пункту *Выполнение иных задач*, а затем — по пункту *Обзор этого компакт-диска*. На экране появится окно программы Проводник, в котором будет показано содержимое корневой папки компакт-диска.
3. В списке файлов и папок дважды щелкните сначала по пункту *SUPPORT*, затем — по пункту *TOOLS*, затем — по пункту *DEPLOY.CAB*.
4. Нажав комбинацию клавиш *Ctrl+A*, выделите все файлы архива, затем щелкните правой кнопкой по списку выделенных файлов и выберите в контекстном меню пункт *Извлечь*.
5. В окне *Выберите конечную папку* щелкните по значку *Мой компьютер*, затем — по значку *Локальный диск (C:)* после чего — по кнопке *Создать папку*, введите в поле ввода название создаваемой папки, например *Deploy*, затем нажмите клавишу *Enter*.

---

1 Для облегчения развертывания системы мы рекомендуем воспользоваться дистрибутивом для многократной установки (Volume License Media), в который уже интегрирован пакет обновлений SP2.

2 Если этого не произойдет, запустите программу SETUP.EXE из корневой папки компакт-диска.

6. Убедитесь, что папка *Deploy* выделена, и щелкните по кнопке *Извлечь*. В результате все файлы из архива будут переписаны в созданную на диске C: папку.
7. Закройте окно программы Проводник и щелкните по кнопке *Выход* в окне программы установки.

Теперь рассмотрим процедуру создания файла ответов:

1. Запустите программу Диспетчер установки (щелкните по кнопке *Пуск*, затем — по пункту *Выполнить*, в поле *Открыть* наберите `c:\deploy\setupmgr` и нажмите клавишу *Enter*).
2. Щелкайте по кнопке *Далее* до тех пор, пока на экране не появится страница *Взаимодействие с пользователем*. На ней установите переключатель в положение *Полностью автоматическая установка* и опять щелкните по кнопке *Далее*.
3. На странице *Дистрибутивный общий ресурс* установите переключатель в положение *Установить с компакт-диска*, затем щелкните по кнопке *Далее*.
4. На следующей странице установите флажок *Я принимаю условия лицензионного соглашения* и щелкните по кнопке *Далее*. После этого на экране останется диалоговое окно, в левой части которого будут перечислены группы параметров, задаваемых в ходе установки операционной системы, а в правой части вы можете определить их конкретные значения. Например, таким образом:

Группа параметров	Значения или рекомендации по их вводу
Имя и организация	Имя: Student Организация: Class
Параметры экрана	Цветовая палитра: High Color Область экрана: 1024 × 768 Частота обновления: 75 Гц
Часовой пояс	Выберите соответствующий вашему региону
Ключ продукта	Укажите 25-символьный ключ продукта для имеющегося у вас дистрибутива Windows XP Professional
Имена компьютеров	Установите флажок *Автоматически создавать имена компьютеров на основе названия организации, указанного на этапе «Имя и организация»*
Пароль администратора	Задайте сложный для подбора пароль, например Gfhjkm<XP>. Установите флажок *При загрузке конечного компьютера автоматически войти как администратор*
Сетевые компоненты	Обычные параметры
Рабочая группа или домен	В составе рабочей группы: WORKGROUP

Дополнительные параметры задавать не обязательно.

5. Щелкайте по кнопке *Далее*, пока не дойдете до раздела *Дополнительные команды* (либо переключитесь в него щелчком по названию), затем щелкните по кнопке *Готово*.
6. В поле *Путь и имя файла* наберите c:\deploy\winnt.sif и щелкните по кнопке *OK*.

Внимание! Файл ответов, применяемый при автоматической установке с компакт-диска, обязательно должен называться WINNT.SIF.

7. Закройте окно программы, щелкнув по кнопке *Отмена*.

Поскольку с ее помощью некоторые нужные параметры нельзя внести в файл ответов, потребуется дополнительно изменить его вручную. Для этого воспользуйтесь программой Блокнот:

1. Откройте файл ответов в Блокноте (щелкните по кнопке *Пуск*, затем — по пункту *Выполнить*, в поле *Открыть* наберите `notepad c:\deploy\ winnt.sif` и нажмите клавишу *Enter*).
2. Найдите в тексте раздел `[Unattended]` и добавьте в него две строки:

```
Repartition=Yes
UnattendSwitch=Yes
```

Первая заставляет программу установки заново разбить жесткий диск компьютера, создав на нем один-единственный раздел, а вторая указывает, что после установки надо пропустить этап *Добро пожаловать в Windows XP*.

3. Найдите в файле раздел [Identification] и добавьте перед ним следующие строки:

```
[RegionalSettings]
SystemLocale=1049
UserLocale=1049
InputLocale=0409:00000409, 0
InputLocale_DefaultUser=0409:00000409, 0419:00000419]
```

За счет этого в ходе автоматической установки будут настроены российские региональные стандарты и возможность ввода символов на английском (по умолчанию) и русском языках.

4. Сохраните измененный файл на диске (например, нажав комбинацию клавиш *Ctrl+S*) и закройте окно программы Блокнот.

5. Перепишите, например с помощью программы Проводник, файл WINNT.SIF в корневую папку заранее отформатированной дискеты.

Примечание. Если в вашем распоряжении имеется дистрибутив не локализованной русской версии, а, например, английской версии Windows XP Professional, в файл ответов может потребоваться внести дополнительные изменения. Информация о том, какие параметры можно указывать в нем, а также сведения о различных вариантах развертывания операционных систем семейства Windows приведены в справочных файлах DEPLOY.CHM и REF.CHM. После извлечения из архива DEPLOY.CAB они находятся в той же папке, что и программа Диспетчер установки.

Автоматическая установка Windows XP Professional с компакт-диска

Чтобы выполнить автоматическую установку Windows XP Professional, проделайте следующее:

1. Загрузите один из компьютеров в классе с дистрибутивного компакт-диска (для этого может потребоваться изменить порядок устройств для загрузки, заданный в настройках BIOS). Когда в процессе загрузки на экране появится сообщение «Press any key to boot from CD», нажмите любую клавишу на клавиатуре, чтобы программа установки начала свою работу.
2. Сразу после этого вставьте в дисковод дискету с записанным на нее файлом WINNT.SIF.

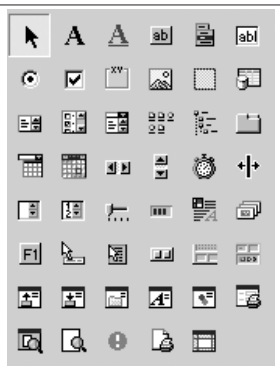
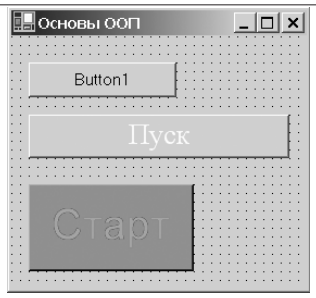
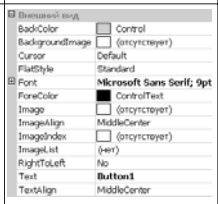
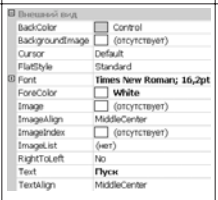
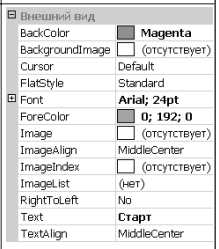
Внимание! В ходе автоматической установки операционной системы жесткий диск компьютера будет заново разбит на разделы и отформатирован. Чтобы не потерять ранее записанные на него данные, не забудьте создать их резервную копию.

3. Когда начнется форматирование жесткого диска, дискету нужно удалить из дисковода и подождать завершения процедуры установки операционной системы. Если на каком-то этапе эта процедура прервется, значит, при подготовке файла ответов были допущены ошибки. Исправьте их и добейтесь полностью автоматического выполнения установки.
4. Если установка завершилась успешно и после перезагрузки компьютера был выполнен автоматический вход в систему (от имени *Администратора*), переходите к следующему этапу.

Приложение 4

Таблицы по объектно-ориентированному языку программирования Visual Basic .NET

Таблица 1. Основы объектно-ориентированного программирования

<table>
<tr><th colspan="2">Классы объектов графического интерфейса</th><th colspan="3">Некоторые свойства класса</th></tr>
<tr><td colspan="2" rowspan="2"> 
Область элементов</td><td>TextBox</td><td>Button</td><td>PictureBox</td></tr>
<tr><td>Внешний вид
BackColor
BorderStyle
Cursor
Font
ForeColor
Lines
RightToLeft
ScrollBars
Text
TextAlign</td><td>Внешний вид
BackColor
BackgroundImage
Cursor
FlatStyle
Font
ForeColor
Image
ImageAlign
ImageIndex
ImageList
RightToLeft
Text
TextAlign</td><td>Внешний вид
BackColor
BackgroundImage
BorderStyle
Cursor
Image</td></tr>
<tr><th colspan="2">Экземпляры класса Button</th><th colspan="3">Некоторые значения свойств экземпляров класса Button</th></tr>
<tr><td colspan="2"> 
Проект в режиме конструирования</td><td>Button1
  </td><td>Button2
  </td><td>Button3
  </td></tr>
</table>

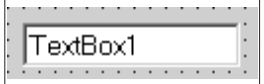
Окончание табл. 1

Свойства, методы и события	**Событийная процедура**		
	Событие: `Click`	**Событие:** `MouseMove`	**Событие:** `KeyDown`
	Свойство: `Visible`	**Метод:** `Hide`	**Свойство:** `Width` **Свойство:** `Height`


После запуска проекта и выполнения событийных процедур
	Private Sub `Button1_Click()`
	`Button1.Visible =` **False**
	End Sub
		Private Sub `Button2_MouseMove()`
		`Button2.Hide()`
		End Sub
			Private Sub `Button3_KeyDown()`
			`Button3.Width = 250`
			`Button3.Height = 50`
			End Sub

Таблица 2. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (начало)

Элемент управления	Внешний вид	Назначение	Свойства*	Методы**	События по умолчанию***
Форма (`Form`)		Является основой для создания графического интерфейса проекта	`Name` — имя. `Text` — надпись. `Location (X; Y)` — координаты верхнего левого угла формы на экране или элемента управления на форме. `Size` (`Width; Height`) — размер (ширина; высота). `BackColor` — цвет фона. `ForeColor` — цвет текста. `Font` (`Name; Size; Bold` и др.) — шрифт (название шрифта, размер и начертание). `Enabled` — возвращает или задает значение, показывающее, имеет ли элемент управления возможность отвечать на действия пользователя	`Show` — показывает элемент управления. `Hide` — скрывает элемент управления. `Focus` — помещает фокус на выбранный элемент управления. `Scale` — масштабирует форму или элемент управления. `Refresh` — перерисовывает форму или элемент управления	`Form1_Load` — происходит при загрузке формы
Надпись (`Label`)		Предназначен для отображения текста или изображений, которые нельзя менять в процессе выполнения проекта			`Label1_Click` — происходит при щелчке по надписи

Элемент управления	Внешний вид	Назначение	Свойства*	Методы**	События по умолчанию***
Текстовое поле (TextBox)		Используется для ввода или отображения данных	Name — имя. Text — надпись. Location (X; Y) — координаты верхнего левого угла формы на экране или элемента управления на форме. Size (Width; Height) — размер (ширина; высота). BackColor — цвет фона. ForeColor — цвет текста. Font (Name; Size; Bold и др.) — шрифт (название шрифта, размер и начертание). Enabled — возвращает или задает значение, показывающее, имеет ли элемент управления возможность отвечать на действия пользователя	Show — показывает элемент управления. Hide — скрывает элемент управления. Focus — помещает фокус на выбранный элемент управления. Scale — масштабирует форму или элемент управления. Refresh — перерисовывает форму или элемент управления	TextBox1_ TextChanged — происходит при изменении текста в текстовом поле
Кнопка (Button)		Щелчок по кнопке вызывает событийную процедуру			Button1_Click — происходит при щелчке по кнопке
Улучшенное текстовое поле (RichTextBox)		Используется для ввода и отображения форматируемого текста			RichTextBox1_ TextChanged — происходит при изменении текста в текстовом поле

* Каждый элемент управления обладает определенным набором свойств, большинство элементов управления обладают свойствами, перечисленными в таблице 1.

** Большинство элементов управления обладают набором методов, перечисленных в таблице 2.

*** Элементы управления реагируют на различные события, но у каждого элемента управления есть наиболее часто использующееся событие (событие по умолчанию).

Таблица 3. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (продолжение 1)

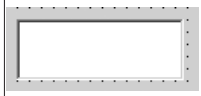
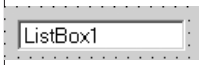
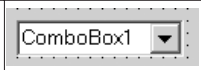
Элемент управления	Внешний вид	Назначение	Свойства	Методы	События по умолчанию
Графическое поле (PictureBox)		Предназначен для отображения графических объектов в различных форматах	Image — путь к графическому файлу, который отображается в графическом поле	Draw — рисование графических примитивов	PictureBox1_Click — происходит при щелчке в графическом поле
Список (ListBox)		Отображает список, в котором можно выбрать один или несколько элементов	Items — позволяет задать или получить значения элементов списка. Sorted — позволяет сортировать элементы списка	Add — добавляет элемент в список. Remove — удаляет элемент из списка	ListBox1_SelectedIndex-Changed — происходит при выборе элемента списка
Раскрывающийся список (ComboBox)		Используется для вывода данных в раскрывающемся списке			ComboBox1_Selected-IndexChanged — происходит при выборе элемента списка
Переключатель (RadioButton)		Два или более переключателей предназначены для выбора взаимоисключающих вариантов	Checked указывает, выбран или нет данный переключатель или флажок. AutoCheck — вызывает автоматическое изменение состояния переключателя или флажка при его выборе		RadioButton1_Checked-Changed — происходит при выборе переключателя
Флажок (CheckBox)		Используется для представления выбора «да/нет» или «истина/ложь»			CheckBox1_Checked-Changed — происходит при выборе флажка
Панель инструментов (ToolBar)		Используется для создания панелей инструментов проектов	Buttons — набор кнопок. ImageList — используемый набор изображений для кнопок		ToolBar1_ButtonClick — происходит при щелчке по кнопке на панели инструментов

Таблица 4. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (продолжение 2)

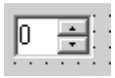
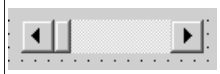
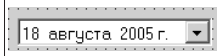
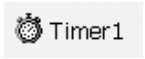
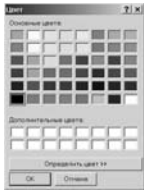
Элемент управления	Внешний вид	Назначение	Свойства	Методы	События по умолчанию
Ползунок (TrackBar)		Используется для визуальной настройки числовых параметров	Value — содержит число, соответствующее положению ползунка, счетчика или движка полосы прокрутки. Minimum — минимальное значение ползунка, счетчика или движка полосы прокрутки. Maximum — максимальное значение ползунка, счетчика или движка полосы прокрутки		TrackBar1_Scroll — происходит при установке ползунка
Счетчик (NumericUpDown)		Выводит и задает числовое значение в списке			NumericUpDown1_Value-Changed — происходит при установке значения счетчика
Горизонтальная прокрутка (HScrollBar)		Обеспечивают удобный просмотр длинных списков и больших массивов информации с помощью горизонтальной или вертикальной прокрутки			HScrollBar1_Scroll — происходит при установке индикатора горизонтальной прокрутки
Вертикальная прокрутка (VScrollBar)					VScrollBar1_Scroll — происходит при установке индикатора вертикальной прокрутки
Дата и время (DateTimePicker)		Позволяет выбрать в списке дату и время	Value — текущее значение даты и времени		DateTimePicker1_Value-Changed — происходит при установке даты и времени
Календарь (MonthCalendar)		Выводит сетку, содержащую дни месяца, разбитые на столбцы по дням недели	TodayDate — текущее значение даты. MaxDate — максимальное значение даты. MinDate — минимальное значение даты		MonthCalendar1_Date-Changed — происходит при установке даты
Таймер (Timer)		Вызывает событие через определенные интервалы времени, не отображается в процессе выполнения проекта	Interval — интервал времени между событиями в миллисекундах	Start — запускает таймер	Timer1_Tick — происходит после запуска проекта

Таблица 5. Элементы управления, входящие в базовую поставку Visual Basic .NET: свойства, методы и события (продолжение 3)

Элемент управления	Внешний вид*	Назначение	Свойства	Методы	
Шрифт (`FontDialog`)	FontDialog1	Позволяет задать параметры шрифта	`Font` — параметры шрифта, установленные по умолчанию	`ShowDialog` — выводит диалоговое окно *Шрифт*	
Цвет (`ColorDialog`)	ColorDialog1	Позволяет задать цвет	`Color` — цвет, установленный по умолчанию	`ShowDialog()` — выводит диалоговое окно *Цвет*	
Открытие файла (`OpenFileDialog`)	OpenFileDialog1	Позволяет в иерархической файловой системе выбрать файл	`InitialDirectory` папка, открывающаяся в диалоговом окне	`ShowDialog()` — выводит диалоговое окно *Открыть*	
Сохранение файла (`SaveFileDialog`)	SaveFileDialog1	Позволяет в иерархической файловой системе сохранить файл	`InitialDirectory` — папка, открывающаяся в диалоговом окне	`ShowDialog()` — выводит диалоговое окно *Сохранить как*	
Меню (`MainMenu`)	MainMenu1	Позволяет создать иерархическое меню проекта	`Text` — позволяет вводить названия пунктов иерархического меню		
Список рисунков (`ImageList`)	ImageList1	Используется для хранения рисунков, которые могут отображаться другими элементами управления	`Images` — позволяет создать коллекцию рисунков		

* Элементы управления не отображаются на форме в процессе выполнения проекта.

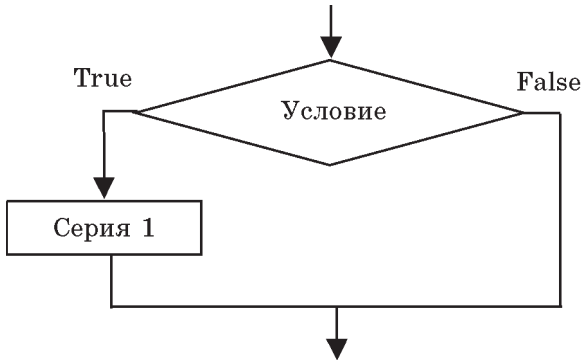
Таблица 6. Типы переменных

Тип переменной	Занимаемая память	Диапазон значений
Целочисленные типы		
`Byte`	1 байт	От 0 до 255 (без знака)
`Short`	2 байта	От –32 768 до 32 767
`Integer`	4 байта	От –2 147 483 648 до 2 147 483 647
`Long` (длинное целое)	8 байтов	От –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
Типы с плавающей запятой		
`Single` (число одинарной точности с плавающей запятой)	4 байта	От –3,4028235E+38 до –1,401298E-45 для отрицательных значений; от 1,401298E-45 до 3,4028235E+38 для положительных значений
`Double` (число двойной точности с плавающей запятой)	8 байтов	От –1,79769313486231570E+308 до –4,94065645841246544E-324 для отрицательных значений; от 4,94065645841246544E-324 до 1,79769313486231570E+308 для положительных значений
`Decimal`	16 байтов	От 0 до +/–79 228 162 514 264 337 593 543 950 335 без десятичной запятой; от 0 до +/–7,9228162514264337593543950335 с 28 знаками после запятой; наименьшее ненулевое значение — +/–0,0000000000000000000000000001 (+/–1E–28)
Строковые типы		
`Char` (символ)	2 байта	От 0 до 65535 (без знака)
`String` (строка переменной длины)	Зависит от реализующей платформы	От 0 до приблизительно 2 миллиардов знаков кодировки Unicode
Логический тип		
`Boolean`	2 байта	`True` или `False`
Дата и время		
`Date`	8 байтов	От 0:00:00 1 января 0001 года до 11:59:59 PM 31 декабря 9999 года
Любой тип		
`Object`	4 байта	В переменной с типом `Object` может храниться любой тип

Таблица 7. Алгоритмическая структура «ветвление» и ее кодирование на языке программирования

Неполное ветвление

Блок-схема	Язык программирования Visual Basic .NET

	Многострочная запись:
	
	`If Условие Then`
	`  Серия 1`
	`End If`
	
	Однострочная запись:
	
	`If Условие Then Серия 1`

Полное ветвление

Блок-схема	Язык программирования Visual Basic .NET

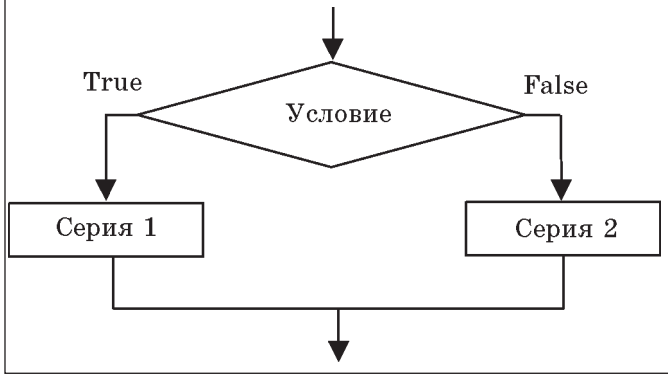
	Многострочная запись:
	
	`If Условие Then`
	`  Серия 1`
	`Else`
	`  Серия 2`
	`End If`
	
	Однострочная запись:
	
	`If Условие Then Серия 1 Else Серия 2`

Таблица 8. Алгоритмическая структура «выбор» и ее кодирование на языке программирования

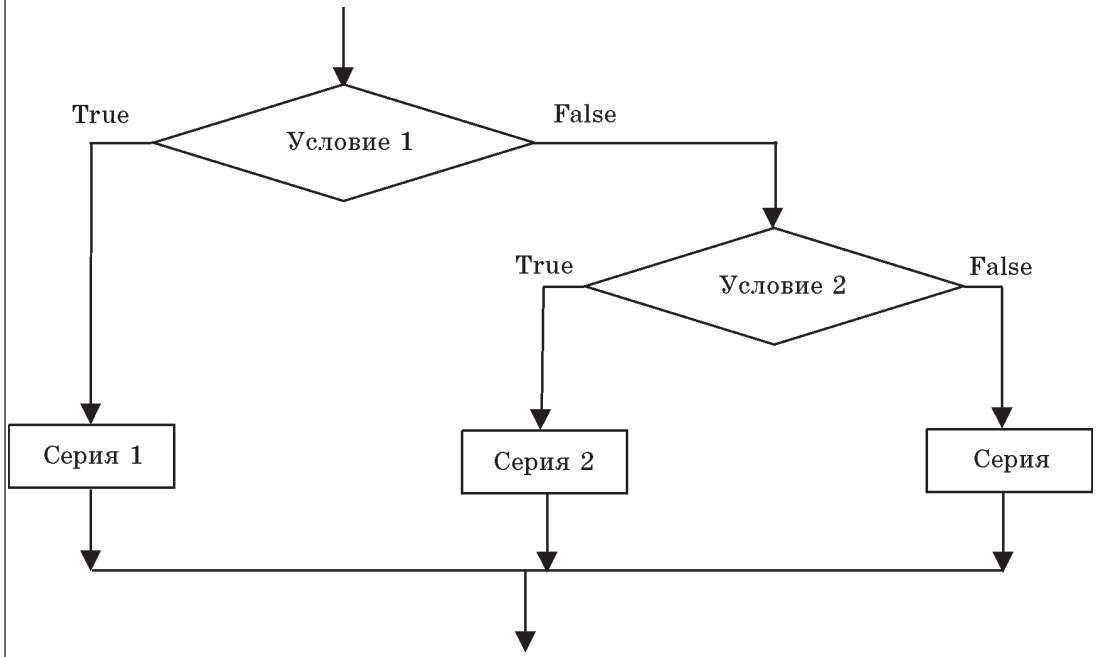
Блок-схема	Язык программирования Visual Basic .NET
	**Select Case** Выражение **Case** Условие1 Серия 1 **Case** Условие2 Серия 2 [**Case Else** Серия] End Select

Таблица 9. Алгоритмическая структура «цикл со счетчиком» и ее кодирование на языке программирования

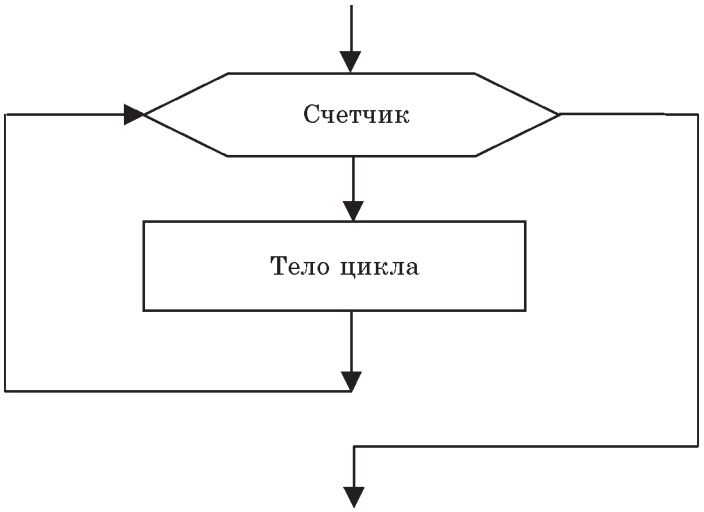
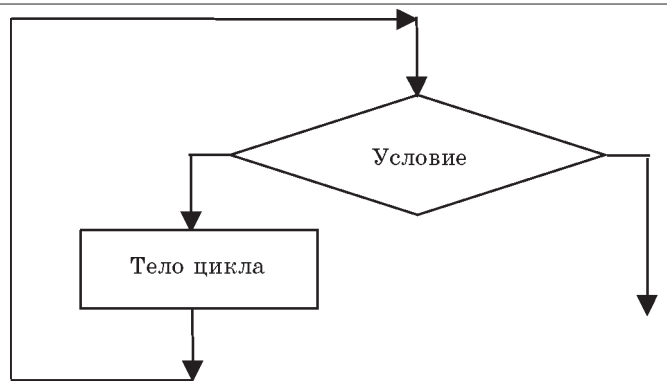
Блок-схема	Язык программирования Visual Basic .NET
	**For** Счетчик=НачЗнач **To** КонЗнач [**Step шаг**] Тело цикла **Next** [Счетчик]

Таблица 10. Алгоритмическая структура «цикл с условием» и ее кодирование на языке программирования

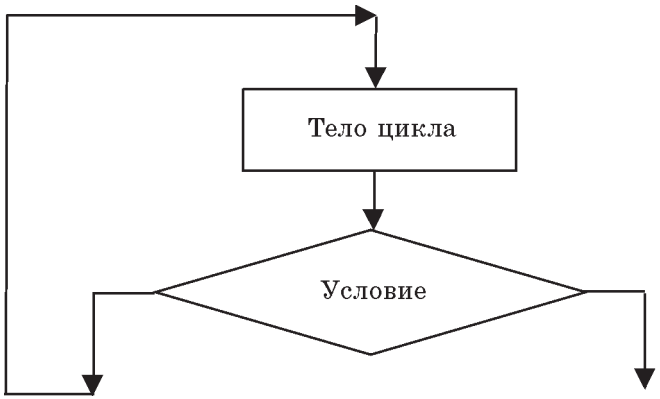
Цикл с предусловием

Блок-схема	Язык программирования Visual Basic .NET

	**Do While** Условие
	  Тело цикла
	Loop
	
	Do Until Условие
	  Тело цикла
	Loop

Цикл с постусловием

Блок-схема	Язык программирования Visual Basic .NET

	**Do**
	  Тело цикла
	Loop While Условие
	
	Do
	  Тело цикла
	Loop Until Условие

Содержание

Введение 3
1. История развития языков программирования 4
2. Введение в объектно-ориентированное программирование 6
3. Краткий обзор .NET Framework и Visual Studio .NET 7
4. Учебные материалы 11
Учебное пособие 12
Компакт-диск 14
5. Подготовка учебного класса к занятиям 15
Системные требования 15
Установка операционной системы Windows XP Professional 17
Дополнительные настройки компьютера 17
Установка разрешений на корневую папку диска C: 17
Настройка квотирования дискового пространства 18
Копирование файлов с примерами 19
Установка Visual Basic .NET 2003 19
Проверка работы Visual Basic .NET 2003 21
Создание учетных записей для учеников 24
Установка и использование Visual Basic 2005 Express Edition 25
Дополнительные материалы для подготовки класса 27
6. Рекомендации по преподаванию курса «Основы программирования на примере Visual Basic .NET» 28
Пояснительная записка 28
Тематическое планирование курса 29
Модуль 1. Программы вокруг нас 35
Модуль 2. Система программирования Visual Basic .NET 36
Модуль 3. Алгоритмы и программы 37
Модуль 4. «Формы и элементы управления» 38
Модуль 5. «Свойства и методы» 40

Модуль 6. Присваивание и переменные 41
Модуль 7. Операции 42
Модуль 8. Ветвление: неполная форма 44
Модуль 9. Ветвление: полная форма 45
Модуль 10. Циклы со счетчиком. 46
Модуль 11. Циклы с условием 47
Модуль 12. Подпрограммы и функции 48
7. Дополнительная литература 49
Книги 49
Ссылки на информацию о .NET Framework, Visual Studio .NET и Visual Basic .NET в Интернете 49
Приложение 1. Реализация объектно-ориентированного программирования в Visual Basic .NET 51
Приложение 2. Справочник по языку Visual Basic .NET 58
1. Переменные в языке программирования Visual Basic .NET 58
2. Массивы в языке программирования Visual Basic 60
3. Основные типы алгоритмических структур и их кодирование на языке Visual Basic 61
3.1. Алгоритмическая структура «ветвление» 61
3.2. Алгоритмическая структура «выбор» 63
3.3. Алгоритмическая структура «цикл» 65
4. Функции в языке программирования Visual Basic .NET. 69
4.1. Функции преобразования типов данных 69
4.2. Математические функции 70
4.3. Строковые функции 70
4.4. Функции ввода и вывода данных 73
5. Возможности работы с графикой в Visual Basic .NET 2003 и Visual Basic 2005 Express Edition 76
Приложение 3. Процедура установки операционной системы Windows XP Professional с загружаемого дистрибутивного компакт-диска. 81
Подготовка файла ответов для автоматизации установки. 81
Автоматическая установка Windows XP Professional с компакт-диска . . . 84
Приложение 4. Таблицы по объектно-ориентированному языку программирования Visual Basic .NET 85