Никита Культин
Лариса Цой

Visual Basic
ДЛЯ СТУДЕНТОВ И ШКОЛЬНИКОВ

Санкт-Петербург
«БХВ-Петербург»
2010

УДК 681.3.068+800.92Visual Basic
ББК 32.973.26–018.1
К90

Культин, Н. Б.

К90 Visual Basic для студентов и школьников / Н. Б. Культин, Л. Б. Цой. — СПб.: БХВ-Петербург, 2010. — 416 с.: ил. + CD-ROM — (ИиИКТ)

ISBN 978-5-9775-0487-4

Рассматривается процесс создания программ различного назначения на языке программирования Visual Basic — от простейших до программ работы с графикой и базами данных. Последовательность изложения, дозировка материала, а также наличие контрольных вопросов и задач для решения соотносятся с учебным процессом. Демонстрируется среда разработки Visual Basic, приводится описание языка программирования Visual Basic, рассматриваются основные алгоритмические структуры, операции со строками, одномерными и двухмерными массивами и файлами, большое внимание уделено практике программирования, что позволит полноценно подготовиться к ЕГЭ по информатике по разделам, касающимся алгоритмизации и программирования. Приложение содержит справочник по языку программирования Visual Basic и базовым компонентам. На компакт-диске приводятся рассматриваемые в книге примеры программ и программа Экзаменатор, позволяющая автоматизировать процесс контроля и самоконтроля знаний.

Для образовательных учреждений

УДК 681.3.068+800.92Visual Basic
ББК 32.973.26–018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 04.12.09.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 33,54.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0487-4

Оглавление

Предисловие

Visual Basic — что это?

В последнее время возрос интерес к программированию. Это связано с развитием и внедрением в повседневную жизнь информационно-коммуникационных технологий. Если кто-то имеет дело с компьютером, то рано или поздно у него возникает желание, а иногда и необходимость, программировать.

Среди пользователей персональных компьютеров в настоящее время наиболее популярна операционная система Windows, и естественно, что тот, кто собирается программировать, стремится писать программы, которые будут работать в ней.

Раньше начинающему программисту оставалось только мечтать о создании собственных программ, работающих в Windows, т. к. средства разработки были явно ориентированы на профессионалов, обладающих серьезными знаниями и опытом.

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на так называемую "быструю разработку", пионером среди которых был пакет Microsoft Visual Basic.

В основе систем быстрой разработки (RAD-систем, Rapid Application Development — среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования. Суть этой технологии заключается в том, что среда разработки берет на себя большую часть рутинной работы, оставляя программисту работу по созданию диалоговых окон и функций обработки событий. Производительность программиста при работе в RAD-системе — фантастическая!

Microsoft Visual Basic — это среда быстрой разработки, в которой в качестве языка программирования используется Visual Basic.

В настоящее время, несмотря на появление новых версий Visual Basic, широко используется шестая версия пакета — Microsoft Visual Basic 6.0, которая стала "классикой".

В Visual Basic 6.0 можно создавать программы различного назначения: от простейших однооконных приложений, до программ, работающих с графикой, мультимедиа и базами данных.

Microsoft Visual Basic может работать в среде операционных систем от Windows 98 до Windows Vista. Особых требований, по современным меркам, к ресурсам компьютера пакет не предъявляет.

Об этой книге

В книге, которая посвящена программированию в конкретной среде разработки, необходим баланс между тремя линиями: язык программирования, технология программирования (программирование как таковое) и среда разработки. Уже при первом знакомстве со средой разработки, представлении ее возможностей у автора возникает проблема: чтобы описать процесс разработки программы, объяснить, как работает программа, нужно оперировать такими терминами как *объект, событие, свойство,* понимание которых на начальном этапе изучения программирования весьма проблематично. Как поступить? Сначала дать описание языка, а затем приступить к описанию среды разработки и процесса программирования? Очевидно, что это не лучший вариант решения. Поэтому при изложении материала принят подход, который можно назвать "от задачи". Суть его заключается в том что, берется конкретная задача и на ее примере рассматривается определенная технология, возможности среды разработки и особенности языка программирования, необходимые для решения этой, конкретной задачи.

Книга, которую вы держите в руках, — это не описание языка программирования и среды разработки Visual Basic. Это руководство по программированию в Microsoft Visual Basic. В нем рассмотрена вся цепочка, весь процесс создания программы: от разработки алгоритма, диалогового окна и процедур обработки событий до установки на компьютер пользователя.

Цель этой книги — научить программировать в Microsoft Visual Basic 6.0, т. е. создавать законченные программы различного назначения: от простых однооконных приложений до программ работы с базами данных.

Научиться программировать можно, только решая конкретные задачи. Поэтому, чтобы получить максимальную пользу от книги, вы должны работать с ней активно. Не занимайтесь просто чтением примеров, реализуйте их с помощью вашего компьютера. Не бойтесь экспериментировать — вносите изменения в программы. Чем больше вы сделаете самостоятельно, тем большему вы научитесь!

Глава 1

Среда программирования Visual Basic

Запускается Visual Basic обычным образом, т. е. выбором в меню **Пуск** команды **Программы ▸ Microsoft Visual Basic 6.0 ▸ Microsoft Visual Basic 6.0** (рис. 1.1).

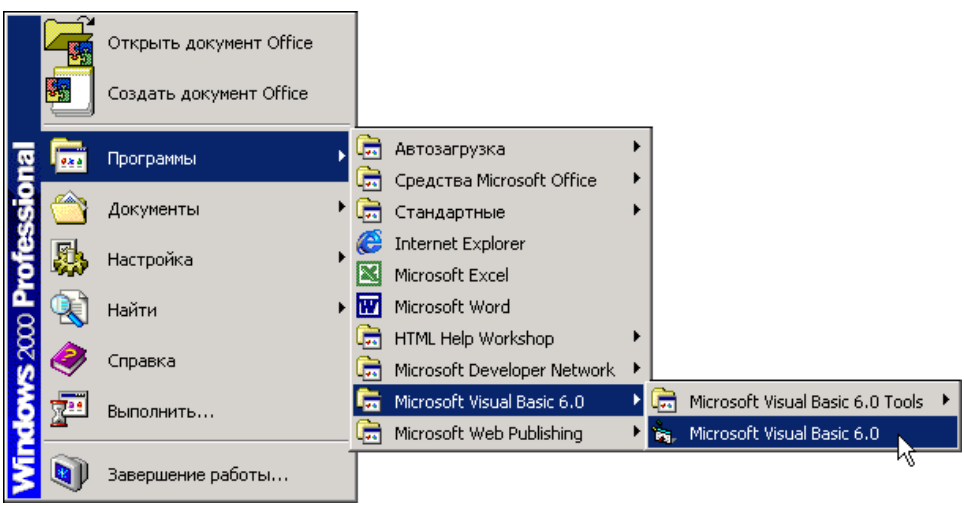


Рис. 1.1. Запуск Visual Basic

Перед тем как запустить Visual Basic первый раз, рекомендуется создать папку для проектов (программ) Visual Basic и указать ее в качестве рабочей. Папка (назвать ее можно, например, VBProjects) создается обычным образом. Чтобы указать, что папка является рабочей, надо раскрыть окно свойств команды запуска Visual Basic (сделать щелчок *правой* кнопкой мыши на команде запуска Visual Basic) и ввести имя в поле **Рабочая папка**.

Если Visual Basic запущен сразу после установки, то на фоне главного окна отображается окно **New Project** (рис. 1.2). В этом окне на вкладке **New** перечислены типы проектов (программ), которые можно создать в Visual Basic.

Чтобы приступить к работе над новой программой, или, как принято говорить, *приложением*, надо выбрать **Standard EXE** и сделать щелчок на кнопке **Открыть**.

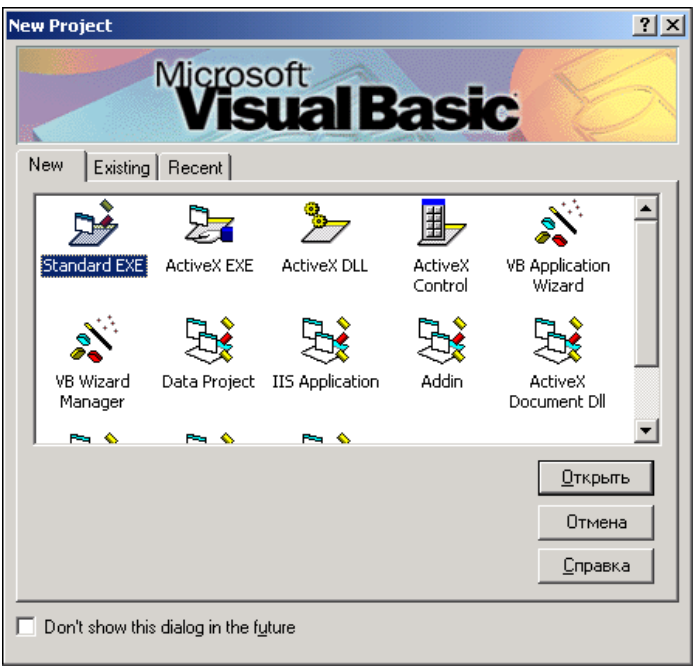


Рис. 1.2. Начало работы над новой программой (приложением)

Замечание

Если после запуска Visual Basic окно **New Project** на экране не отображается, то, для того чтобы начать работу над новой программой, надо в меню **File** выбрать команду **New Project**.

Окно Visual Basic в начале работы над новой программой приведено на рис. 1.3. В верхней части окна находится строка меню и панель инструментов, слева — *палитра компонентов*, в центре — окно *конструктора формы*, справа — окно *проекта*, окно *свойств* и окно *отображения положения формы*.

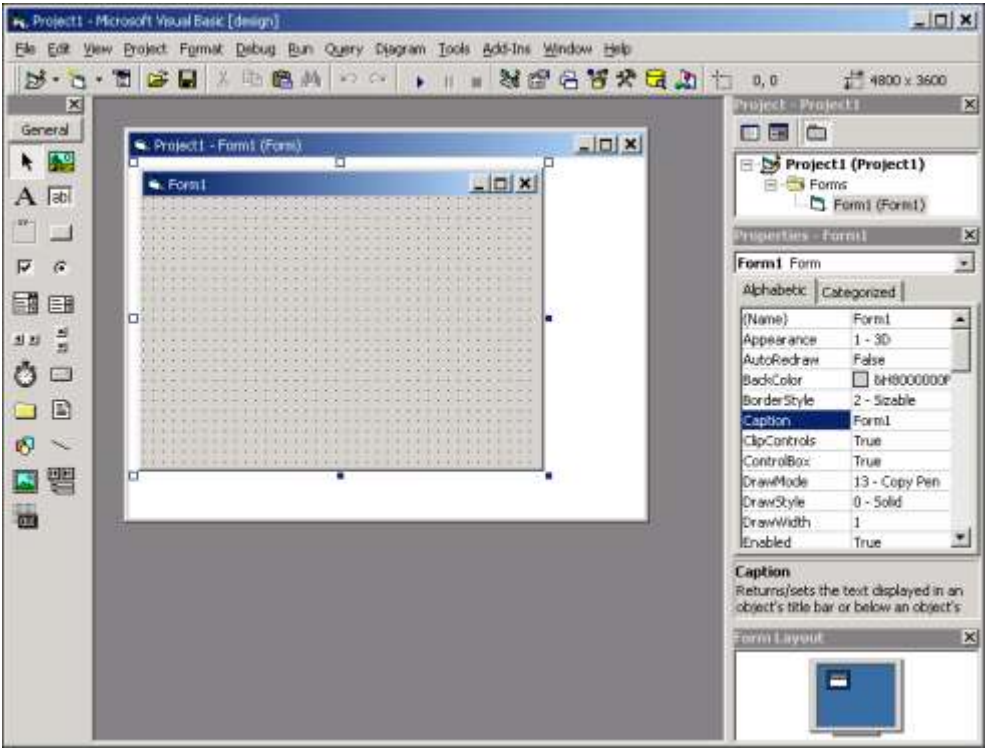


Рис. 1.3. Окно Visual Basic в начале работы над новой программой

На стандартной панели инструментов (рис. 1.4) находятся кнопки активизации наиболее часто используемых команд. Там же находятся кнопки, используя которые можно быстро сделать доступным окно палитры компонентов, менеджера проектов, свойств и др.

Рис. 1.4. Стандартная панель инструментов

Окно программы во время ее разработки принято называть формой.

В окне конструктора формы (рис. 1.5) находится *форма* — заготовка окна разрабатываемого приложения.

В палитре компонентов (рис. 1.6) отображаются значки компонентов, которые программист может поместить на форму.

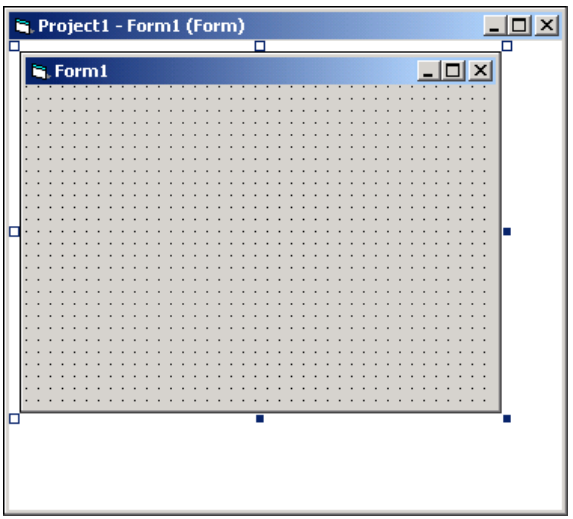


Рис. 1.5. Окно конструктора формы

Рис. 1.6. Палитра компонентов (окно **Toolbox**)

Замечание

Если палитра компонентов не отображается, то, для того чтобы она стала доступной, надо в меню **View** выбрать команду **Toolbox** или сделать щелчок на соответствующей кнопке панели инструментов.

В окне проекта (рис. 1.7) отображается структура (состав) проекта, над которым в данный момент идет работа.

Замечание

Если окно проекта не отображается, то, для того чтобы оно стало доступным, надо в меню **View** выбрать команду **Project Explorer** или сделать щелчок на соответствующей кнопке панели инструментов.

Окно **Properties** (рис. 1.8) предназначено для редактирования значений свойств объектов. В нем отображаются свойства выбранного в данный момент объекта (в начале работы над новой программой — формы). На вкладке **Alphabetic** свойства отображаются в алфавитном порядке,

а на вкладке **Categorized** свойства сгруппированы по функциональному признаку. Например, в группу **Appearance** объединены свойства, которые определяют вид объекта, а в группу **Position** — его размер и положение на экране (для формы) или поверхности формы (другие компоненты).

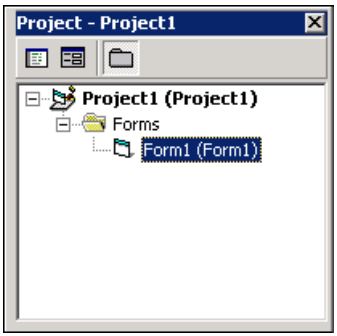


Рис. 1.7. В окне **Project** отображается структура (состав) проекта

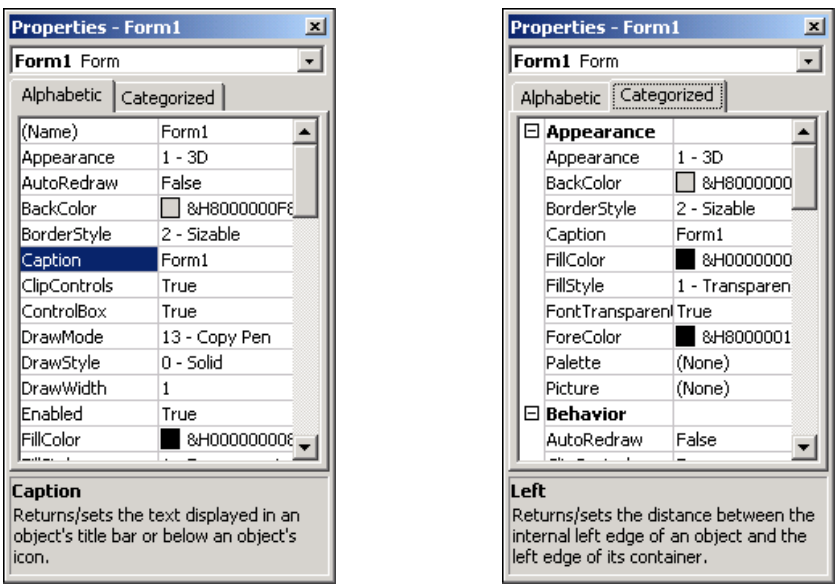


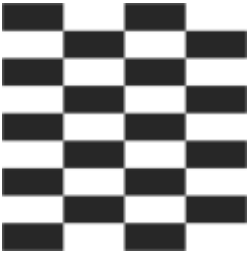
Рис. 1.8. В окне **Properties** перечислены свойства объекта и указаны их значения

В терминологии визуального проектирования *объект* — это диалоговое окно или элемент интерфейса пользователя (поле ввода, командная кнопка, пере-

ключатель и др.). *Свойство* — это характеристика, которая определяет (задает) внешний вид объекта. Например, значение свойства `Caption` задает заголовок формы, а свойств `Width` и `Height` — ее размер.

Контрольные вопросы

1. Что такое приложение?
2. Перечислите основные окна среды разработки Visual Basic.
3. Что надо сделать, если какое-либо из окон, например окно свойств, на экране не отображается?
4. Как начать работу над новой программой?
5. Что такое свойство?



Глава 2

Первый проект

Процесс создания программы в Visual Basic рассмотрим на примере. Создадим приложение, которое позволяет пересчитать цену из долларов в рубли, — Конвертер (рис. 2.1).

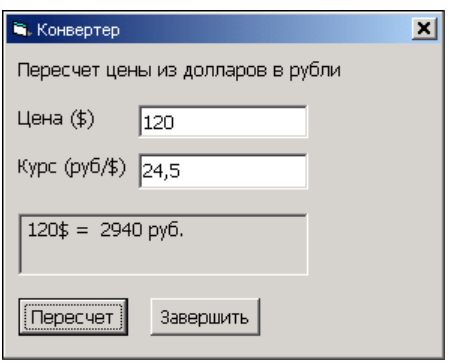


Рис. 2.1. Окно программы пересчета цены из долларов в рубли

Начало работы

Чтобы начать работу над новой программой, или, как принято говорить, проектом, надо сначала в меню **File** выбрать команду **New Project**, затем в открывшемся окне **New Project** указать тип создаваемого приложения (**Standard EXE**) и сделать щелчок на кнопке **OK**.

Форма

Работа над новой программой начинается с создания стартовой (главной) формы.

Стартовая форма создается путем изменения значений ее свойств (табл. 2.1) и добавления к форме необходимых компонентов (полей ввода, отображения текста, командных кнопок и т. д.).

Таблица 2.1. *Свойства формы*

Свойство	**Описание**
`Name`	Имя формы. Используется для доступа к форме и ее компонентам
`Caption`	Текст заголовка
`Width`	Ширина формы
`Height`	Высота формы
`StartUpPosition`	Положение формы при первом ее появлении на экране. Форма может располагаться в центре экрана (`Center Screen`), в центре родительской формы (`Center Owner`). Положение формы могут определять также значения свойств `Top` и `Left` (в этом случае значение свойства `StartUpPosition` должно быть равно `Manual`)
`Top`	Расстояние от верхней границы формы до верхней границы экрана или до верхней границы родительской формы
`Left`	Расстояние от левой границы формы до левой границы экрана или до левой границы родительской формы
`Icon`	Значок (картинка) в заголовке
`MaxButton`	Признак наличия в заголовке окна кнопки **Развернуть**
`MinButton`	Признак наличия в заголовке окна кнопки **Свернуть**
`BorderStyle`	Стиль (вид) границы. Граница может быть обычной (`Sizable`), тонкой (`Fixed Single`) (в этом случае изменить размер окна путем перемещения границы мышью нельзя) или вообще отсутствовать (`None`). Если значение свойства равно `Fixed Dialog`, то граница окна тонкая и кнопки **Развернуть** и **Свернуть** в заголовке не отображаются
`BackColor`	Цвет формы. Цвет можно задать, выбрав из палитры или указав привязку к элементу цветовой схемы операционной системы. Во втором случае цвет определяется текущей цветовой схемой и выбранным компонентом привязки и меняется при изменении цветовой схемы операционной системы
`ScaleMode`	Единица измерения размеров и координат компонентов, находящихся на форме. Размер и координаты компонентов могут измеряться в твипах (`Twip`), пикселах (`Pixel`), миллиметрах (`Millimeter`) и других единицах

Таблица 2.1 *(окончание)*

Свойство	Описание
`Font`	Шрифт, который по умолчанию используется находящимися на поверхности формы компонентами для отображения текста (например, надпись на командной кнопке, текст в поле редактирования или в поле отображения текста)

Для изменения значений свойств формы и компонентов используется окно **Properties**. В верхней части окна **Properties** указано имя объекта, значения свойств которого отображаются в данный момент. В левой колонке окна перечислены свойства объекта, в правой — указаны значения свойств.

Сначала надо задать заголовок формы — изменить значение свойства `Caption` с `Form1` на `Конвертер`. Чтобы это сделать, нужно в окне **Properties** выбрать свойство `Caption` и щелкнуть мышью в поле значения свойства. В результате этих действий в поле значения свойства (после слова `Form1`) появится курсор и можно будет ввести значение свойства (рис. 2.2).

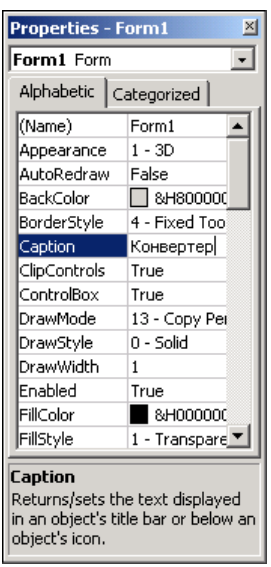


Рис. 2.2. Изменение значения свойства путем ввода строки

Следует обратить внимание, что ширина и высота формы измеряются в специальных единицах — твипах. Задавать значения свойств `Width` и `Height` в твипах неудобно. Проще захватить один из находящихся на границе формы

черных квадратиков и переместить границу (вертикальную, горизонтальную или обе сразу) в нужном направлении (рис. 2.3). По окончании перемещения границы значения свойств `Width` и `Height` автоматически изменятся и будут соответствовать установленному размеру формы.

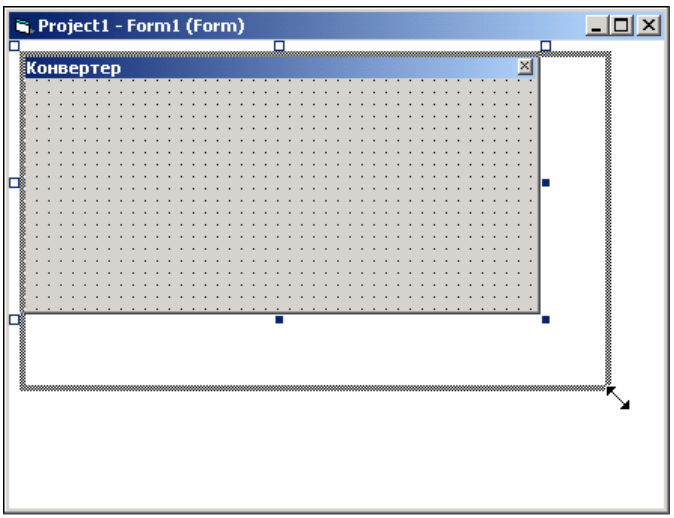


Рис. 2.3. Изменение размера формы путем перемещения границы

При выборе некоторых свойств, например `BorderStyle`, справа от текущего значения свойства появляется значок раскрывающегося списка. Очевидно, что значение таких свойств можно задать путем выбора из списка (рис. 2.4). Здесь следует обратить внимание, что в списке сначала указывается числовое значение константы, а затем — ее название. При этом не следует путать название константы и ее обозначение. Например, численное значение константы `Fixed Single` равно единице, а символьное значение — `vbFixedSingle`.

Рядом со значениями некоторых свойств отображается командная кнопка с тремя точками. Это значит, что для изменения значения свойства используется дополнительное диалоговое окно. Например, в результате щелчка на кнопке с тремя точками в строке свойства `Icon` открывается окно **Load Icon**, в котором можно открыть один из каталогов компьютера и выбрать ICO-файл — картинку, которая будет изображать системное меню в заголовке формы.

В табл. 2.2 приведены значения свойств стартовой формы разрабатываемой программы. Остальные свойства оставлены без изменения и в таблице не приведены.

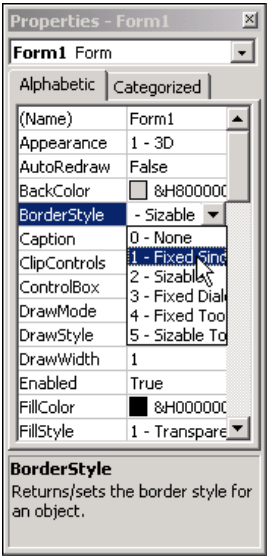


Рис. 2.4. Изменение значения свойства путем выбора из списка

Таблица 2.2. *Значения свойств формы*

Свойство	**Значение**
Caption	Конвертер
Width	4425
Height	3480
BorderStyle	FixedSingle
MaxButton	False
MinButton	False
StartUpPosition	CenterScreen
ScaleMode	Pixel
Font	Tahoma, обычный, 10 pt

Компоненты

Программа пересчета цены из долларов в рубли должна получить от пользователя исходные данные — цену в долларах и курс (соотношение рубля к доллару). Ввод данных с клавиатуры обеспечивает компонент `TextBox` — поле редактирования. Поэтому в форму надо добавить два компонента `TextBox`.

Чтобы добавить в форму компонент TextBox, нужно в палитре компонентов сделать щелчок на значке компонента (рис. 2.5). Затем установить указатель мыши в ту точку формы, в которой должен быть левый верхний угол компонента. Потом нажать левую кнопку мыши и переместить указатель мыши в точку, в которой должен быть правый нижний угол компонента. После того как кнопка мыши будет отпущена, на форме появится компонент.

Рис. 2.5. Значок компонента TextBox

Каждому компоненту, добавленному в форму, автоматически присваивается имя, которое формируется из стандартного имени компонента и его порядкового номера. Например, первый компонент TextBox получает имя Text1, второй — Text2. Программист, путем изменения значения свойства Name, может изменить имя компонента. В простых программах имена компонентов, как правило, не изменяют.

На рис. 2.6 приведен вид формы программы "Конвертер" после добавления двух компонентов TextBox, предназначенных для ввода исходных данных. Один из компонентов выделен. Свойства этого (выделенного) компонента отображаются в окне **Properties**. Чтобы увидеть свойства другого компонента, надо щелкнуть левой кнопкой мыши на его изображении в форме или выбрать имя нужного компонента в раскрывающемся списке, который находится в верхней части окна **Properties**.

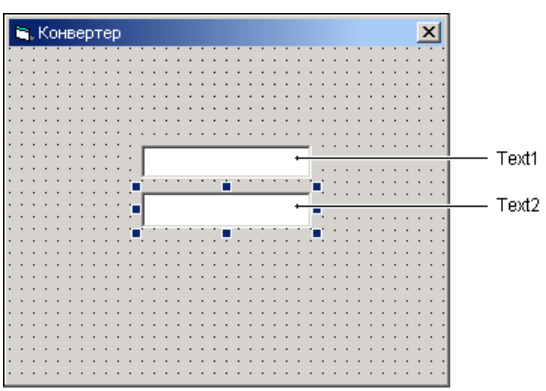


Рис. 2.6. Форма после добавления компонентов TextBox

В табл. 2.3 перечислены основные свойства компонента TextBox.

Таблица 2.3. *Свойства компонента* `TextBox`

Свойство	Описание
`Name`	Имя компонента. Используется для доступа к компоненту и его свойствам
`Text`	Текст, находящийся в поле редактирования
`Left`	Расстояние от левой границы компонента до левой границы формы
`Top`	Расстояние от верхней границы компонента до верхней границы формы
`Width`	Ширина поля редактирования
`Height`	Высота поля редактирования
`Font`	Шрифт, используемый для отображения текста в поле компонента
`ForeColor`	Цвет текста
`Locked`	Блокировка. Используется для ограничения возможности изменения текста в поле редактирования. Если значение свойства равно `True`, то текст в поле редактирования изменить нельзя
`MultiLine`	Разрешает (`True`) отображение текста в несколько строк
`ScrollBars`	Управляет отображением полос прокрутки. У компонента может быть полоса вертикальной прокрутки (`Vertical`), горизонтальной (`Horizontal`) или обе полосы прокрутки (`Both`). Если значение свойства равно `None`, то полосы прокрутки не отображаются
`Visible`	Позволяет скрыть компонент (`False`) или сделать его видимым (`True`)

Изменить размер и положение компонента на форме можно, присвоив нужные значения соответственно свойствам `Width`, `Height`, `Left` и `Top` или при помощи мыши.

Для того чтобы изменить положение компонента, необходимо установить указатель мыши на его изображении, нажать левую кнопку мыши и, удерживая кнопку нажатой, переместить контур компонента в нужную точку формы, затем отпустить кнопку мыши. Во время перемещения компонента (рис. 2.7) отображаются текущие значения координат левого верхнего угла компонента (значения свойств `Left` и `Top`).

Для того чтобы изменить размер компонента, необходимо установить указатель мыши на один из маркеров, помечающих границу компонента, нажать левую кнопку мыши и, удерживая кнопку нажатой, изменить положение гра-

ницы компонента. Затем отпустить кнопку мыши. Во время изменения размера компонента отображаются текущие значения свойств `Height` и `Width` (рис. 2.8).

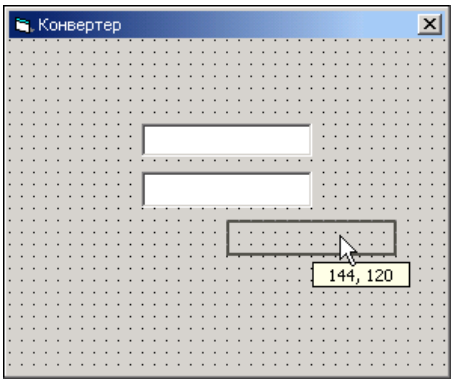


Рис. 2.7. Отображение текущих значений свойств `Left` и `Top` при изменении положения компонента

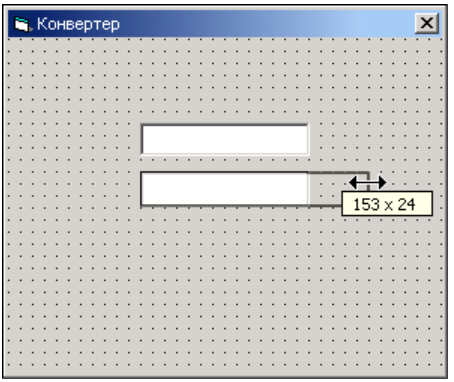


Рис. 2.8. Отображение текущих значений свойств `Width` и `Height` при изменении размера компонента

В табл. 2.4 приведены значения свойств компонентов `Text1` и `Text2`. Компонент `Text1` предназначен для ввода цены, `Text2` — курса доллара. Обратите внимание на то, что значением свойства `Text` обоих компонентов является пустая строка. Также следует обратить внимание, что координаты и размер компонентов указаны в пикселах.

Таблица 2.4. *Значения свойств компонентов* `Text1` *и* `Text2`

Компонент	Свойство	Значение
`Text1`	`Left`	88
	`Top`	47
	`Width`	113
	`Height`	22
	`Text`	—
`Text2`	`Left`	88
	`Top`	87
	`Width`	113
	`Height`	22
	`Text`	—

Помимо полей редактирования в окне программы должна находиться краткая информация о программе и назначении полей редактирования. Отображение текста на поверхности формы обеспечивает компонент `Label` (рис. 2.9). Добавляется компонент `Label` в форму точно так же, как и поле редактирования.

Свойства компонента `Label` перечислены в табл. 2.5.

Рис. 2.9. Компонент `Label` — поле отображения текста

Таблица 2.5. *Свойства компонента* `Label`

Свойство	Описание
`Name`	Имя компонента. Используется для доступа к компоненту и его свойствам
`Caption`	Текст, отображаемый в поле компонента
`Left`	Расстояние от левой границы компонента до левой границы формы
`Top`	Расстояние от верхней границы компонента до верхней границы формы
`Width`	Ширина компонента (поля отображения текста)
`Height`	Высота компонента (поля отображения текста)
`AutoSize`	Признак автоматического изменения размера компонента в соответствии с его содержимым
`WordWrap`	Признак необходимости переноса текста, который не помещается в текущей строке, в следующую строку (значение свойства `AutoSize` должно быть `False`)
`Alignment`	Способ выравнивания текста в поле компонента. Текст может быть выровнен по левому краю (`LeftJustify`), по центру (`Center`) или по правому краю (`RightJustify`)
`Font`	Шрифт, используемый для отображения текста. По умолчанию используется шрифт, заданный для формы
`ForeColor`	Цвет текста в поле компонента
`BorderStyle`	Задает вид границы компонента. По умолчанию граница отсутствует
`Visible`	Позволяет скрыть компонент (`False`) или сделать его видимым (`True`)

В форму разрабатываемого приложения надо добавить четыре компонента Label. Первый компонент Label обеспечивает вывод информационного сообщения, второй и третий — отображение информации о назначении полей редактирования, четвертый — отображение результата расчета.

После того как компоненты Label будут добавлены в форму, надо выполнить их настройку — установить значения свойств в соответствии с табл. 2.6.

Таблица 2.6. *Значения свойств компонентов* `Label1`, `Label2`, `Label3` *и* `Label4`

Компонент	Свойство	Значение
Label1	Caption	Пересчет цены из долларов в рубли
	Left	8
	Top	8
	AutoSize	True
Label2	Caption	Цена ($)
	Left	8
	Top	40
	AutoSize	True
Label3	Caption	Курс (руб/$)
	Left	8
	Top	72
	AutoSize	True
Label4	Caption	—
	Left	8
	Top	112
	Width	193
	Height	41
	BorderStyle	Fixed Single

Форма программы после настройки компонентов Label должна выглядеть так, как показано на рис. 2.10.

Последнее, что надо сделать на этапе создания формы, — добавить в форму две командные кнопки: **Пересчет** и **Завершить**. Назначение этих кнопок очевидно.

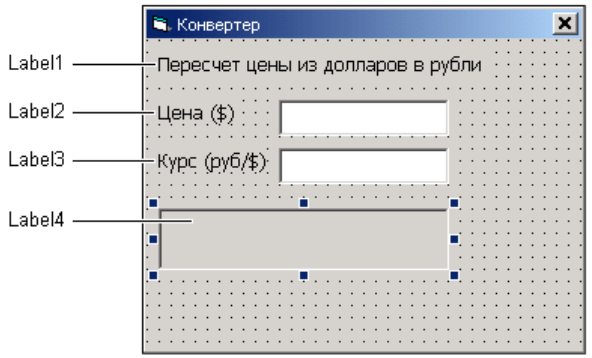


Рис. 2.10. Форма после добавления полей отображения текста

Рис. 2.11. Командная кнопка — компонент `CommandButton`

Командная кнопка — компонент `CommandButton` (рис. 2.11) — добавляется в форму точно так же, как и другие компоненты. Свойства компонента приведены в табл. 2.7.

***Таблица 2.7.** Свойства компонента* `CommandButton`

Свойство	**Описание**
`Name`	Имя компонента. Используется для доступа к компоненту и его свойствам
`Caption`	Текст на кнопке
`Left`	Расстояние от левой границы кнопки до левой границы формы
`Top`	Расстояние от верхней границы кнопки до верхней границы формы
`Width`	Ширина кнопки
`Height`	Высота кнопки
`Enabled`	Признак доступности кнопки. Если значение свойства равно `True`, то кнопка доступна и пользователь может ее нажать, сделав щелчок на кнопке левой кнопкой мыши или, если фокус находится на кнопке, нажав клавишу <Enter>. Если значение свойства равно `False`, то кнопка не доступна

***Таблица 2.7** (окончание)*

Свойство	Описание
Visible	Позволяет скрыть кнопку (False) или сделать ее видимой (True)
Style	Тип кнопки. Кнопка может быть обычной (Standard) или "графической" (Graphical). На поверхности "графической" кнопки отображается картинка
Picture	Картинка, которая отображается на "графической" кнопке (значение свойства Style должно быть равно Graphical)
DisabledPicture	Для "графической" кнопки задает картинку, которая отображается на кнопке в случае, если она не доступна (значение свойства Enabled равно False)
DownPicture	Для "графической" кнопки (значение свойства Style равно Graphical) задает картинку, которая отображается на кнопке в случае, если она нажата
ToolTipText	Задает текст подсказки, которая появляется при позиционировании указателя мыши на кнопке

После добавления к форме двух командных кнопок (компонентов CommandButton) нужно установить значения их свойств в соответствии с табл. 2.8.

***Таблица 2.8.** Значения свойств компонентов Command1 и Command2*

Компонент	Свойство	Значение
Command1	Caption	Пересчет
	Left	8
	Top	168
	Width	73
	Height	25
Command2	Caption	Завершить
	Left	96
	Top	168
	Width	73
	Height	25

Окончательный вид формы разрабатываемого приложения "Конвертер" приведен на рис. 2.12.

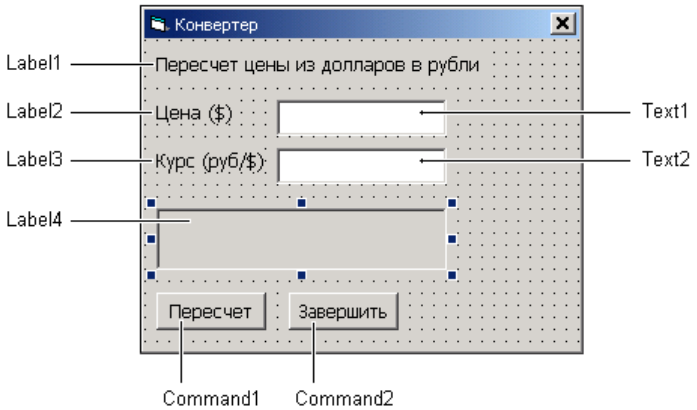


Рис. 2.12. Форма программы "Конвертер"

Завершив работу над формой приложения, можно приступить к программированию — созданию процедур обработки событий.

Событие и процедура обработки события

Вид созданной формы подсказывает, как работает приложение. Очевидно, что пользователь должен ввести в поля редактирования исходные данные и сделать щелчок на кнопке **Пересчет**. Щелчок на изображении командной кнопки — это пример того, что называется *событием*.

Событие (`Event`) — это то, что происходит во время работы программы. У каждого события есть имя. Например, щелчок кнопкой мыши на изображении командной кнопки — это событие `Click`, нажатие клавиши в процессе ввода строки текста в поле компонента `TextBox` — событие `KeyPress`.

В табл. 2.9 приведены некоторые события.

***Таблица 2.9.** События*

Событие	Описание
`Click`	Щелчок кнопкой мыши
`DblClick`	Двойной щелчок кнопкой мыши
`MouseDown`	Нажатие кнопки мыши
`MouseUp`	Отпускание кнопки мыши
`MouseMove`	Перемещение мыши
`KeyPress`	Нажатие клавиши

Таблица 2.9 *(окончание)*

Событие	Описание
KeyDown	Нажатие клавиши. События KeyDown и KeyPress — это чередующиеся, повторяющиеся события, которые происходят до тех пор, пока не будет отпущена удерживаемая клавиша (в этот момент происходит событие KeyUp)
KeyUp	Отпускание нажатой (удерживаемой) клавиши
Initialize	Создание объекта (например, формы). Процедура обработки этого события обычно используется для инициализации переменных и выполнения подготовительных действий
Activate	Событие происходит, когда элемент управления (форма) становится активным окном
Paint	Событие происходит при появлении окна на экране в начале работы программы, а также во время работы программы, когда окно вновь становится видимым, например, после того как пользователь развернет свернутое окно или отодвинет другое окно, которое перекрывает окно программы
Resize	Изменение размера формы или элемента управления
GotFocus	Получение элементом управления фокуса, например перемещение курсора в поле редактирования текста
LostFocus	Потеря элементом управления фокуса, например перемещение курсора из одного поля редактирования в другое

Следует обратить внимание, что действия пользователя, как правило, приводят к возникновению последовательности (цепочки) событий. Например, в начале работы программы возникает цепочка событий Initialize — Load — Activate — Resize — Paint, а в конце работы программы, когда пользователь сделает щелчок на кнопке **Закрыть**, — QueryUnload — Unload — Terminate.

Реакцией на событие должно быть какое-либо действие. В Visual Basic реакция на событие реализуется как *процедура обработки события*. Таким образом, для того чтобы программа выполняла некоторую работу в ответ на действия пользователя, программист должен написать процедуру обработки соответствующего события. Следует обратить внимание на то, что значительную часть обработки событий берет на себя компонент. Поэтому программист должен разрабатывать процедуру обработки события только в том случае, если реакция на событие отличается от стандартной или не определена. Например, если по условию задачи ограничений на символы, вводимые в поле редактирования (компонент TextBox), нет, то процедуру обработки