bhv®

Юлий Кетков, Александр Кетков

САМОУЧИТЕЛЬ

Практика программирования: Visual Basic, C++ Builder, Delphi

Основы визуального программирования

Общие свойства компонентов визуальных сред

Около 100 готовых к исполнению программ

Содержит исходные тексты всех приведенных программ

Языки программирования в примерах

Юлий Кетков, Александр Кетков

Практика программирования: Visual Basic, C++ Builder, Delphi

Санкт-Петербург

«БХВ-Петербург»

2002

УДК 681.3.06
ББК 32.973.26-018.1
К37

Кетков Ю. Л., Кетков А. Ю.

К37 Практика программирования: Visual Basic, C++ Builder, Delphi. — СПб.: БХВ-Петербург, 2002. — 464 с.: ил.

ISBN 5-94157-191-7

Книга знакомит читателя с системами визуального проектирования Visual Basic, Borland C++ Builder, Delphi. В ней обсуждаются характеристики наиболее часто используемых компонентов (объектов). При этом особое внимание уделяется общности функциональных свойств рассматриваемых компонентов и способов их применения в приложениях, создаваемых в разных системах программирования. В книге содержится множество готовых решений и иллюстраций, которые помогут читателю легко разобраться в изложенном материале.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Анатолий Адаменко*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Петр Науменко*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Татьяна Звертановская*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.05.02.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 37,41.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-191-7

СОДЕРЖАНИЕ

Предисловие

Почему мы написали эту книгу

Эта книга является логическим продолжением самоучителя по алгоритмическим языкам Бейсик, Си и Паскаль, опубликованного издательством "БХВ-Петербург" в 2001 году. В нашей предыдущей работе основной упор делался на изучение наиболее популярных алгоритмических языков без особой привязки к операционной системе и конкретной среде программирования, хотя большинство примеров были ориентированы на работу под управлением MS-DOS. Основная задача новой книги — научиться создавать приложения в среде Windows 9*x*/NT/ME/2000/XP и овладеть техникой программирования в наиболее распространенных средах визуального программирования. Как и в предыдущей книге, мы стараемся подчеркнуть то, что объединяет различные визуальные среды. Поэтому мы сознательно пошли на некоторый параллелизм в описании функциональных возможностей тех или иных компонент разных систем программирования. Умение проводить аналогии между сходными по назначению структурами и программами поможет вам в дальнейшем быстрее адаптироваться к новым инструментальным средствам, которые неизбежно придут на смену сегодняшнему программному обеспечению.

Мы не ставим перед собой задачу познакомить читателей со всеми возможностями описываемых систем программирования. В рамках начального и даже более продвинутого курса по визуальному программированию это не представляется реальным — слишком велик объем информации и разнообразен круг приложений, в которых могут быть использованы те или иные стандартные средства визуальной среды. Количество компонентов, поставляемых с современными версиями систем, перешагивает границу в 150–200 элементов. А если учесть, что каждый из них может обладать полусотней свойств и обслуживаться несколькими десятками методов, то ни один учебник не в состоянии дать исчерпывающую информацию, сопроводив ее полноценными примерами на каждый случай в жизни. Из всего разнообразия компонент мы выбрали джентльменский набор, который, на наш взгляд, полезен для проектирования любого приложения. С него можно начинать

знакомство с различными свойствами и методами новых технологий. Но ни в коем случае не надо на этом останавливаться. Не делаем мы и особого акцента на выборе и сравнении различных версий визуальных сред. Большинство примеров, приводимых в книге, апробировалось в системах VisualBasic 6.0, Borland C++ Builder 5.0 и Delphi 6.0. Их несложно адаптировать к более ранним или более поздним версиям соответствующих систем программирования.

Несмотря на то, что фундаментом современных визуальных сред являются идеи объектно-ориентированного программирования (ООП) и их реализации в различных алгоритмических языках, мы не требуем от читателей непременного знакомства с ООП. Для большинства приложений, которые вам придется создавать, вполне достаточно умения использовать готовые объекты. Минимум сведений и терминов по ООП, которые будут встречаться в этой книге, занимает не более 1–2 страниц. Первая система визуального программирования Visual Basic, появившаяся в 1991 году, вообще не содержала и намека на такие понятия, как *инкапсуляция* (объединение данных и методов), *классы* (шаблоны с описанием данных и методов), *объекты* (наборы данных и методов, сформированные по шаблонам классов), *наследование* (сохранение фамильных черт в порожденных классах), *полиморфизм* (возможность создавать разные функции с одинаковыми именами). Даже такой тип данных, как *указатели* в Бейсике отсутствовал. Поэтому создание новых VBX компонентов в первых визуальных средах осуществлялось совершенно другими программными средствами, почти недоступными рядовому пользователю.

Для кого написана эта книга?

Для тех, кто изучил один из алгоритмических языков высокого уровня и собирается писать программы, функционирующие под управлением Windows. И, конечно же, хочет научиться создавать профессионально оформленные приложения Windows, пользующиеся спросом на рынке современных программных продуктов (на Западе, как и у нас — встречают по одежке).

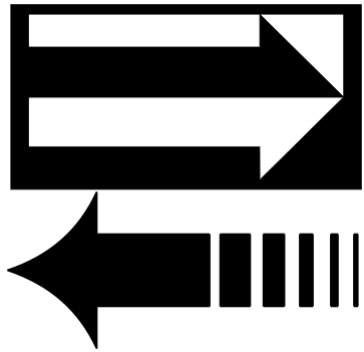
Для тех, кто владеет основами программирования в одной из визуальных сред и хочет познакомиться со спецификой работы в среде, использующей другой алгоритмический язык. Мы считаем, что профессиональный программист не должен замыкаться в рамках одного алгоритмического языка и, тем более, в рамках единственной системы программирования.

Для преподавателей информатики, осваивающих новые программные технологии.

Мы будем крайне признательны читателям за любые предложения по совершенствованию предлагаемых программ и расширению состава демонстрационных примеров.

Глава 1

Краткий обзор



1.1. Немного истории

Появление операционных оболочек типа Windows 3.*x*/95/NT существенно осложнило процесс разработки прикладных программ.

Во-первых, теперь любое приложение должно следовать канонам Windows в плане оформления пользовательского интерфейса, рабочим полем которого уже не является весь экран дисплея, допускавший в рамках MS-DOS прямую адресацию видеопамяти, а перемещаемое по экрану и изменяющее свои размеры типовое окно, снабженное большим количеством элементов управления.

Во-вторых, работа в среде Windows потребовала отказа от традиционной процедурно-ориентированной схемы, в которой последовательность действий приложения жестко запрограммирована. Прикладные программы, выполняющиеся под управлением Windows, организованы в виде набора процедур, откликающихся на адресованные им сообщения. В этом смысле их функционирование повторяет схему работы Windows, что заставляет программиста учитывать довольно большое количество типовых событий и связанных с ними форматов системных сообщений.

Снижению трудоемкости процесса создания Windows-приложений в значительной мере содействовала разработка библиотек типа *Object Windows Library (OWL)*, в которых накапливались как объекты интерфейсного типа, так и классы объектов, упрощающие реализацию типовых процедур обработки данных (графика, работа со списками и таблицами, редактирование текстов и т. п.). Несмотря на очевидный прогресс объектно-ориентированного подхода, традиционная работа с постоянно возрастающим количеством объектов и порождающих их классов требует знания многочисленных деталей — иерархии семейств, параметров и характеристик соответствующих методов и т. п.

Наиболее изящное решение указанных выше проблем, обеспечивающее минимальные затраты на создание Windows-приложений, предлагает технология визуального программирования. Визуальная среда принимает на себя все заботы по оформлению типовых окон — главное окно прикладной программы, диалоговые окна ввода строк и вывода стандартных сообщений. Она предоставляет программисту возможность без каких-либо хлопот встроить в приложение тот или иной объект, приписать ему необходимые свойства, сформировать схему поведения объекта в случае возникновения определенных событий. Прилагательное *визуальный* при этом означает, что основное большинство процедур по заготовке объектов и установке их свойств выполняются по принципу "вижу то, что делаю". Ни одного оператора в своем приложении для этих целей писать не приходится. Указав элемент в палитре компонентов, вы можете с помощью мыши "отбуксировать" объект на нужное место и изменить его размеры. Установка свойств объекта не сложнее выбора строки меню или набора нужного значения в поле ввода. Заботы по отображению установленных объектов и управлению их поведением во время работы приложения система берет на себя.

Впервые, в законченном виде визуальный подход к созданию Windows-приложений был реализован фирмой Microsoft в системе Visual Basic (сокращенно — VB), появившейся в 1992 г. и мгновенно завоевавшей симпатии программистов. В течение двух следующих лет система VB неизменно входила в состав лучших программных продуктов года. Первые три версии VB обновлялись примерно через год. Дольше других задержалась на рынке версия VB 3.0 — последняя полностью интерпретирующая система. Приложения, которые разрабатывались в этой среде, могли работать только в связке с динамически загружаемой библиотекой VBRUN*xx*.DLL (символы *xx* соответствуют номеру версии — 10, 20, 30). Интерпретация Basic-программы существенно замедляла работу приложений по сравнению с аналогичными exe-файлами, изготовленными в других системах компилирующего типа. Только последние версии VB 5.0 и 6.0 наряду с традиционной схемой создания и интерпретации p-кода допускают создание исполняемых exe-файлов. За рубежом Basic-системы пользуются огромной популярностью. По утверждению некоторых специалистов из 5 миллионов программирующих пользователей около 2 миллионов отдают предпочтение Basic. Библиотеки инструментов VB-систем (VBX- и OCX-компоненты) активно расширяются, на их базе появляются новые профессиональные пакеты и наборы инструментальных средств.

Пакет VB распространяется как самостоятельная среда программирования, так и в составе комплекса разработчика Visual Studio. Несколько огорчает решение фирмы Microsoft поставлять современные версии VB без хорошо продуманного встроенного файла помощи. Вместо этого вы должны приобрести и установить автономную справочную систему MSDN (Microsoft Developer Network Library), современная версия которой занимает 3 ком-

пакт-диска. Существует довольно много ситуаций, когда в установленной части справочных материалов не оказывается ответа на возникший вопрос, и тогда вас просят либо установить один из дисков MSDN, либо обратиться за помощью на фирменный сайт. Таким образом, ваш компьютер должен быть подключен к Интернету, и вы будете нести соответствующие расходы за использование каналов связи. Параллельно с автономно функционирующими системами VB фирма Microsoft активно продвигает урезанные версии VBA (Visual Basic for Applications — выпускается с 1993 г.), которые с 1995 г. встраиваются в такие популярные продукты, как MS Word, MS Access, MS Excel. Еще более ограниченное подмножество, известное под названием VBScript, предназначено для создания Web-приложений.

Версия VB 1.0 поставлялась всего на двух 5-дюймовых дискетах. Она могла работать под управлением Windows 3.1 на IBM-совместимых ПК с процессором 80286, оперативной памятью 1 Мбайт и занимала на винчестере всего 2 Мбайта. По мере расширения возможностей VB-систем разрастались и требования к оборудованию ПК. Начиная с версии 4.0, Visual Basic поставляется в одной из трех редакций — учебная (Learning Edition), профессиональная (Professional Edition) и промышленная (Enterprise Edition). *У*чебная версия Visual Basic позволяет создавать большинство приложений для MS Windows и Windows NT. Она включает все наиболее важные элементы управления, компоненты для работы с таблицами и базами данных. Профессиональная редакция включает в себя все возможности учебной версии, дополнительные компоненты ActiveX, средства разработки интернет-приложений, интегрированную базу данных и конструктор динамических страниц HTML. Промышленная редакция позволяет профессионалам создавать распределенные серверные приложения, работающие не только на IBM-совместимых компьютерах под управлением различных операционных систем.

В рамках современных операционных систем семейства Windows среда VB не выдвигает каких-либо дополнительных требования к объему оперативной памяти. Эти ограничения диктуются, главным образом, операционной системой. Как правило, чем больше оперативная память, тем быстрее работают все приложения. Однако на винчестере VB занимает от 25—35 Мбайт в минимальной установке до 80—140 Мбайт в полной установке. К этому следует добавить порядка 120—200 Мбайт для установки документации, включающей файлы помощи, учебные пособия и справочную библиотеку MSDN.

Наиболее серьезную конкуренцию продукции фирмы Microsoft составила система визуального программирования Delphi, разработанная в 1995 г. компанией Borland International на базе языка Object Pascal. Объектно-ориентированный подход к созданию компонент был серьезным шагом вперед по сравнению с чрезмерно сложной технологией разработки и модификации VBX-компонент. Дополнительный выигрыш обеспечивался за счет нормальной компиляции, обеспечивающей получение более производительных

программ. Кроме того, запросы по ресурсам у системы Delphi 1.0 были намного скромнее — процессор 80386, оперативная память от 6 до 8 Мбайт, винчестер от 30 Мбайт в минимальной установке до 80 Мбайт в полной. Первые две версии Delphi довольно быстро завоевали симпатии не только вузовских аудиторий, где Pascal пользовался особым уважением, но и среди профессионалов. Это подтолкнуло одно из подразделений фирмы Borland International на перенос визуальной технологии в среду C++. Так, почти одновременно с появлением Delphi 2.0 на рынке появилась первая версия Borland C++ 6Builder. Основу первой версии BCB составила библиотека визуальных компонент VCL (Visual Component Library), перенесенная без изменений из Delphi 2. Интерфейсы сред Delphi и BCB похожи друг на друга как близнецы, да и большая часть BCB была разработана на языке Object Pascal в среде Delphi. Благодаря своему происхождению система BCB оказалась двуязычной. Кроме своего основного языка программирования она позволяет практически без каких-либо доработок использовать формы, объекты и модули, разработанные в среде Delphi. Чтобы еще больше расширить сферу влияния среды BCB, ее авторы в последующих версиях обеспечили возможность использования библиотеки классов MFC (Microsoft Foundation Classes), разработанной фирмой Microsoft и обычно применяемой в связке с системой Visual C++.

Выход версии Delphi 3.0 в 1997 г. лишь чуть-чуть опередил появление BCB 3.0, а в последующие годы соответствующие подразделения фирмы Borland практически одновременно выдали четвертую (начало 1999 г.) и пятую (январь 2000 г.) версии обеих систем. Совсем недавно появилась шестая версия Delphi, тогда как параллельная ей версия BCB несколько задержалась.

По сложившейся традиции фирма Borland International (ее новое название — Inprise Corporation) выпускает каждую версию систем визуального программирования Delphi и BCB в нескольких редакциях, начиная с минимальной конфигурации под названием Standard.

Стандартная конфигурация позволяет создавать 32-разрядные приложения общего назначения, функционирующие под управлением Windows 95/98/NT/Me/2000/XP. Между такими приложениями могут использоваться механизмы обмена данными OLE (Object Linking and Embedding, связывание и внедрение объектов Windows) и COM (Component Object Model, компонентная модель объектов). Они могут получать доступ к некоторым локальным базам данных и готовить отчеты на базе этой информации.

Профессиональная конфигурация (Professional Edition) расширена набором средств по управлению локальными базами данных типа Access, Paradox, dBase. Она поддерживает механизм связи с базами данных *ODBC (Open Database Connectivity)* и включает некоторые объекты типа ActiveX.

Конфигурация "клиент-сервер" (Client/server Suite) расширяет возможности создания сетевых приложений, функционирующих как в памяти одного

компьютера, так и в памяти нескольких компьютеров. Приложение "Клиент" имеет возможность передать запрос на обработку данных приложению "Сервер". Сервер выполняет задание и возвращает результаты клиенту. Такая технология существенно разгружает каналы связи. Кроме того, сервер, находящийся на компьютере с высокой производительностью, может располагать более мощными средствами обработки данных и одновременно обслуживать несколько клиентов. Конфигурация "клиент-сервер" расширена набором драйверов, обеспечивающих поддержку унифицированного языка запросов к базам данных *SQL (Structured Query Language)*, средствами создания интернет-приложений с различными протоколами обмена, возможностью унифицированной обработки многомерных данных (Decision Cube). В ее состав входит сервисный набор InstallShield, упрощающий создание программы инсталляции разработанного приложения.

Самая мощная редакция BCB 5.0 называется Enterprise Edition (возможно, сказывается подражание фирме Microsoft). В эту редакцию дополнительно включены средства защиты разрабатываемых приложений и их данных, объекты и методы для создания многоплатформенных приложений, функционирующих на компьютерах разного типа под управлением соответствующих операционных систем.

Большинство из описанных выше редакций допускает 3 варианта установки, предусматривающие подключение минимальных средств системы (Compact), стандартного (Typical) и полного (Full) наборов функциональных возможностей. Выбор того или иного варианта определяется наличием свободного места на винчестере вашего компьютера. Минимальные требования к оборудованию ПК, предъявляемые различными версиями BCB, приведены в табл. 1.1. Соответствующие версии Delphi мало чем отличаются от аналогичных разработок своих партнеров.

Таблица 1.1. *Минимальные требования конфигурации ПК, предъявляемые различными версиями среды Borland C++ Builder*

Версия BCB	Процессор (минимальный)	Оперативная память: минимальная (рекомендуемая), Мбайт	Винчестер: Compact, Typical, Full, Мбайт	Версия Windows
1.0 Professional	80486	16 (24)	80, N/A, 125	95
1.0 Client/server	80486	24	N/A, N/A, 195	95
3.0 Professional	80486/100	24 (32)	125, 210, 300	95, 98, NT
3.0 Client/server	80486/100	32	N/A, N/A, 300	95, 98, NT
4.0 Professional	80486/100	32 (64)	175, 210, 300	95, 98, NT
4.0 Client/server	Pentium-90	64	N/A, N/A, 300	95, 98, NT

Таблица 1.1 *(окончание)*

Версия BCB	Процессор (минимальный)	Оперативная память: минимальная (рекомендуемая), Мбайт	Винчестер: Compact, Typical, Full, Мбайт	Версия Windows
5.0 Standard	Pentium-90	32 (64)	120, N/A, 185	95-2000, NT
5.0 Professional	Pentium-90	32 (64)	240, N/A, 360	95-2000, NT
5.0 Enterprise	Pentium-166	64	253, N/A, 388	95-2000, NT

Примечание: N/A — информация отсутствует.

1.2. Типы данных

По сравнению со своими предшественниками из MS-DOS визуальные системы программирования располагают довольно широкими возможностями по представлению данных. В этом разделе мы ограничимся краткой справкой о типах данных, используемых в системах Visual Basic, C++ Builder и Delphi. Более подробная информация будет приведена в последующих главах.

1.2.1. Основные типы данных в системе Visual Basic

Типы данных в VB-программах устанавливаются одним из четырех способов:

- явно при непосредственном объявлении тех или иных переменных, например —

```
Dim X As Integer
k As Long
F1 As String
```

- явно с помощью спецсимвола, завершающего имя переменной, например —

```
X%
k!
F1$
```

- неявно — путем определения интервала букв, с которых могут начинаться имена переменных, не объявленные явным образом, например —

```
DefInt I-N
DefStr S,V
DefDbl X-Z
```

- неявно — путем объявления без указания типа (`Dim Alpha, Beta`) или включения в программу без какого-либо объявления. В любом случае такие данные относятся к типу `Variant`.

Перечень служебных слов, определяющих типы данных, и набор соответствующих указаний о значении первого и последнего символа имени приведены в табл. 1.2.

В системе Visual Basic появились однобайтовые целые числа без знака (`Byte`), которые в ряде случаев позволят расположить в оперативной памяти массивы большего размера.

Условно целочисленные данные типа `Currency` ориентированы на обработку денежных сумм. По сути дела операции над этими данными выполняются как над целыми числами, а результат делится на 10 000. Целая часть значений типа `Currency` представляет базовые денежные единицы (рубли, доллары, марки и др.), а дробная — мелкие (копейки, центы, пфенниги и др.).

Basic дольше других алгоритмических языков сопротивлялся введению логических переменных, хотя неявно использовал для этой цели нулевые (`False`) и ненулевые (`True`) целые числа. Теперь логические переменные вступили в свои законные права, и в программах можно использовать обычные логические выражения над логическими операндами.

Для решения задач, использующих календарные даты, предлагаются данные типа `Date`, подробно описанные в *главе 9*.

Строки фиксированной длины объявляются с указанием точного количества символов, составляющих значение строки:

```
Dim S1 As String *20
```

Если присваиваемое значение содержит меньше 20 символов, то оно дополняется пробелами. При попытке присвоить переменной `S1` слишком длинную цепочку символов, лишний хвост будет отрезан. Если в операторе `Dim` указание о максимальной длине отсутствует, то реальным ограничением сверху может стать лишь объем оперативной памяти.

Самым интересным изобретением системы Visual Basic являются универсальные данные типа `Variant`. Переменным этого типа можно присваивать значения любых типов кроме записей. Переменные типа `Variant` хранят информацию о типе и месте расположения текущего значения. Иногда текущее значение находится внутри поля, отведенного под переменную типа `Variant`. Как правило, это относится к значениям числового типа. В случае более сложных переменных, длина которых может изменяться во время работы программы, на поле переменной хранится ссылка на соответствующее значение. Значением переменной типа `Variant` может быть даже массив. Естественно, что операции над такими данными реализуются специальными подпрограммами, к большинству из которых пользователь напрямую не обращается. Более точно, переменные типа `Variant` представляют собой объекты класса данных, представленных связкой (тип, значение). Над этими данными определен довольно большой набор операций, методов и преобразований. И хотя универсальные данные позволяют делать много необычного,

хранение таких значений связано с дополнительными затратами оперативной памяти, да и скорость выполнения операций оставляет желать лучшего.

Переменные типа `Variant` можно объявить и явно с указанием типа:

```
Dim Delta As Variant
```

Данные, представленные последовательностью полей со значениями фиксированного типа и больше известные под названиями структуры (терминология C) или записи (терминология Pascal), относятся к пользовательским типам данных (*user-defined type*). Описание шаблона структуры с именем `Book` в VB-программе может иметь вид:

```
Type Book
  Title As String *40
  Author As String *20
  Year As Integer
  Price As Single
End type
```

После такого описания можно объявить, например, переменные `b1` и `b2`, имеющие структуру типа `Book`, и присвоить соответствующим полям нужные значения:

```
Dim b1 As Book, b2 As Book
b1.Title="Visual Basic 6.0: Разработка приложений"
b1.Author="А. Гарнаев"
b1.Year=2001
b1.Price=135.5
```

Массивы данных любого типа помимо памяти, занимаемой каждым элементом массива, требуют дополнительно 20 байт на общую шапку и по 4 байта на каждое измерение. Например, массив `A`, объявленный как `Dim A(5,12) As Double`, занимает $6 \times 13 \times 8 + 20 + 2 \times 4 = 652$ байта. В табл. 1.2 описаны основные типы данных, принятые в среде Visual Basic 6.0.

Таблица 1.2. *Описание основных типов данных среды Visual Basic 6.0*

Тип данных	Длина, байт	Диапазон допустимых значений	По первой букве	Спец-символ
Целые числа				
`Byte`	1	от 0 до 255	DefByte	
`Integer`	2	от –32 768 до 32 767	DefInt	%
`Long`	4	от –2 147 483 648 до 2 147 483 647	DefLng	&
`Currency` (целые×10-4)	8	от –922337203685477.5808 до +922337203685477.5807	DefCur	@

Таблица 1.2 *(окончание)*

Тип данных	Длина, байт	Диапазон допустимых значений	По первой букве	Спец-символ
Вещественные числа				
`Single`	4	от -3.4×10^{38} до -1.4×10^{-45} и от 1.4×10^{-45} до 3.4×10^{38}	DefSng	!
`Double`	8	от -1.8×10^{308} до -4.9×10^{-324} и от 4.9×10^{-324} до 1.8×10^{308}	DefDbl	#
Строки				
`String` (фикс. длина)	n	n — объявленная длина строки (n <64 k)		$
`String` (перем. длина)	10+m	m — текущая длина строки ($m<2^{31}-1$)	DefStr	$
Логические данные				
`Boolean`	2	`True` или `False`	DefBool	
Календарные даты				
`Date`	8	от 1.01.100 до 31.12.9999	DefDate	
Пользовательский тип данных (структуры, записи)				
`Type`	Сумма длин полей	Длина поля зависит от его типа		
Указатели (ссылки)				
`Object`	4	Ссылка (адрес) на объект	DefObj	
Универсальный тип данных				
`Variant` (число)	16	Любое число из диапазона `Double`	DefVar	
`Variant` (строка)	22+m	m — текущая длина строки	DefVar	

Для явного объявления типа, кроме оператора `Dim`, могут использоваться и другие операторы, например `Private` (при объявлении переменных, недоступных другим модулям), `Public` (при объявлении переменных,

доступных другим модулям), `Static` (при объявлении локальных переменных, сохраняющих свои значения после выхода из процедуры).

Указание о типе данных (`As ...`) можно встретить при объявлении типа констант, типа параметров функций и подпрограмм, типа возвращаемого значения:

```
Const e As Single = 2.718
Function Square(x As Double) As Double
```

Последние версии Visual Basic сделали попытку приблизить Basic к общепринятому правилу других алгоритмических языков — обязательному описанию всех используемых переменных. Если в начале модуля встречается оператор `Option Explicit`, то все переменные данного модуля должны быть объявлены явно. Система может автоматически добавлять указанный оператор, если в окне **Options** (Опции) (вызывается из меню **Tools** (Инструменты)) на вкладке **Editor** (Редактор) (рис. 1.1) помечен режим **Require Variable Declaration** (Требуется Объявление Переменных).

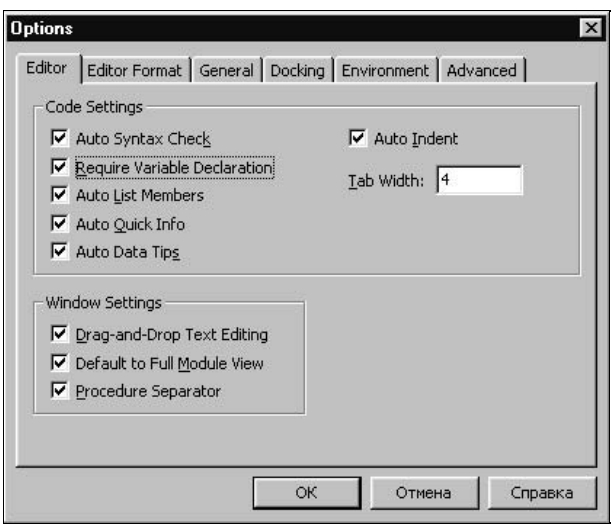


Рис. 1.1. Режим обязательного описания переменных в VB

1.2.2. Основные типы данных в системах BCB и Delphi

В программах на C и Pascal все переменные должны быть объявлены. Причем Pascal предъявляет гораздо более жесткие требования — описание переменных должно предшествовать первому исполняемому оператору программы. Язык C++ более либерален — он позволяет объявлять переменные по ходу программы, но использует специальное соглашение по поводу

области видимости (доступности) таких переменных. Например, переменная `j`, объявленная в цикле, после выхода из цикла становится недоступной:

```
for(int j=0, s=0; j<n; j++)  s=s+a[j];
```

Зоной видимости переменной, объявленной внутри функции C, является блок, в котором такое описание появилось. Это не согласуется с тем, что было реализовано в системе Borland C++ 3.1. Там зона видимости переменной распространялась и на ту часть функции, которая находилась ниже блока. Но в BCB стандарт C++ соблюдается более четко.

Принципиальной новинкой в Delphi является возможность задания начального значения переменной в строке объявления. Раньше Pascal допускал такое присвоение только в разделе `const`, и из книги в книгу кочевали такие несуразицы как *типизированные константы*. На самом деле, это были обычные переменные, но уж слишком долго Pascal не разрешал того, что было принято в большинстве алгоритмических языков. Кстати, Visual Basic пока еще до этого удобного способа не созрел.

По сравнению с системой Visual Basic целочисленные данные в BCB и Delphi в каждом из форматов (1, 2 и 4 байт) допускают по два варианта представления — со знаком и без знака (табл. 1.3). Кроме того, здесь появились целочисленные данные учетверенной длины (8 байт), которые немного отличаются от данных типа `Currency`.

Диапазон вещественных данных по сравнению с Visual Basic здесь расширен за счет 10-байтовых чисел с плавающей точкой, имеющих до 19—20 верных значащих цифр.

Логические значения, похожие на общепринятые `true` или `false`, а также логические операции, в том числе и поразрядные, в C были известны с самого начала. Однако в качестве логических значений выступали данные целочисленного типа или их отдельные биты, а знаки общепринятых логических операций (`OR`, `AND`, `XOR`, `NOT`) были заменены очень специфической символикой (`|`, `&`, `^`, `!`, `~`, `&&`, `||`). Появление данных типа `bool` приблизило C к общепринятой схеме обозначений, хотя и не внесло ничего принципиально нового.

Данные типа `Currency` перекочевали из входного языка VB сначала в систему Delphi, а потом и в BCB. Над денежными единицами в классе `Currency` определены все арифметические операции, свойственные целочисленным данным в языке C:

```
+, +=, ++, -, -=, --, *, *=, /, /=, %, %=
```

Данные типа `Currency` можно сравнивать (`>`, `>=`, `<`, `<=`, `==`, `!=`). Им можно присваивать любые целочисленные или вещественные значения из допустимого интервала. Для преобразования денежных единиц в строку программа может воспользоваться функцией `AnsiString`:

```
Currency s=s1+s2;
ShowMessage(AnsiString(s));
```

Множества впервые появились в Pascal. В соответствующий класс BCB их идеология перенесена в несколько урезанном виде. Элементами класса `Set` могут быть данные типа `int`, `char` или `enum`, принадлежащие ограниченному интервалу от минимального значения (`min`) до максимального значения (`max`) и содержащие не более 256 элементов. Минимальное значение должно быть не менее 0, а максимальное — не более 255.

Объявить множество можно явно или косвенно:

```
Set <char,'A','Z'> s1;     //множество s1 больших букв от A до Z
typedef Set <int, 10,20> Num10_20;
Num10_20 s2,s3;            //два множества целых чисел от 10 до 20
enum M_K_Key (ssShift, ssAlt, ssCtrl, ssLeft, ssRight,
              ssMiddle, ssDouble); //управляющие клавиши
typedef Set<M_K_Key,ssShift,ssDouble> TShiftState;
TShiftState Shift;         //множество управляющих клавиш
```

После объявления любое множество является пустым и его заполнение возлагается на программу. Подключение элемента из допустимого интервала осуществляется с помощью операции `<<`:

```
s1 << 'A'<< 'C';
s2 << 12 << 10 << 19;
MessageDlg(as,mtInformation,TMsgDlgButton<<mbOk<<mbCancel,0);
```

Операция `>>` используется для исключения элемента из множества:

```
s2 >> 19;
```

С помощью метода `Clear` можно удалить все данные из множества:

```
s1.Clear();
```

Для проверки на принадлежность элемента множеству предназначен метод `Contains`:

```
bool b1=s1.Contains('C');
```

Он возвращает значение `true`, если указанный элемент в данный момент принадлежит множеству.

Над элементами двух множеств определены стандартные операции объединения (операция `+` или `+=`), пересечения (операция `*` или `*=`) и вычитания (операция — или `-=`). Два множества можно сравнивать на совпадение (операция `==`) или несовпадение (операция `!=`).

Данные типа `Variant`, впервые появившиеся в VB, были предназначены для обмена разнотипной информацией. При каждом таком значении хранятся сведения о его типе, а класс `Variant` снабжен сервисными программами по преобразованию данных одного типа в другой. В качестве значений типа `Variant` система BCB допускает использование практически всех числовых данных (`short`, `int`, `float`, `double`, `Currency`, `TDateTime`), логических

величин (bool, WordBool), указателей на строки (char *, AnsiString&, wchar_t *) и объекты типа *OLE Automation*. В последнем случае значение типа Variant может использоваться для опроса/установки свойств OLE-объекта или вызова его методов.

В табл. 1.3 дается описание основных типов принятых в средах Delphi 6.0 и Borland C++ Builder.

Таблица 1.3. *Описание основных типов данных сред Delphi 6.0 и Borland C++ Builder 5.0*

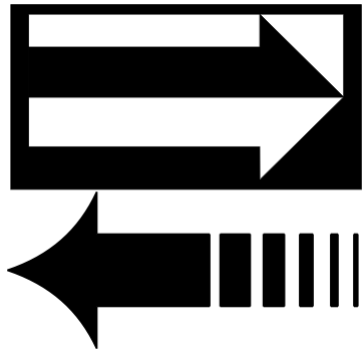
Тип Delphi	Тип BCB	Диапазон	Формат
Целые числа			
ShortInt	signed char	от –128 до 127	8 бит со знаком
Byte	unsigned char	от 0 до 255	8 бит без знака
SmallInt	short	от –32 768 до 32 767	16 бит со знаком
Word	unsigned short	от 0 до 65 567	16 бит без знака
Integer, Longint	integer, long	от –2147483648 до 2147483647	32 бита со знаком
Cardinal	unsigned int	от 0 до 4294967295	32 бита без знака
Int64	int64	от -2^{63} до $2^{63}-1$	64 бита со знаком
Currency	Currency	от –922337203685477.5808 до +922337203685477.5807	64 бита со знаком ($\times 10^{-4}$)
Вещественные числа			
Single, Real	float	от -3.4×10^{38} до -1.4×10^{-45}, 0 и от 1.4×10^{-45} до 3.4×10^{38}	32 бита, пл. зпт.
Double	double	от -1.8×10^{308} до -4.9×10^{-324}, 0 и от 4.9×10^{-324} до 1.8×10^{308}	64 бита, пл. зпт.
Comp	Comp	от -2^{63} до $2^{63}-1$	64 бита, пл.зпт.
Extended	long double	от $-1.1*10^{4932}$ до -3.4×10^{-4932}, 0 и от 3.4×10^{-4932} до 1.1×10^{4932}	80 бит, пл. зпт.
Логические данные			
Boolean	bool	true или false	8 бит
ByteBool	unsigned char	true/false или целое без знака	8 бит
WordBool		true/false или целое без знака	16 бит
LongBool	BOOL	true/false или целое без знака	32 бита

Таблица 1.3 *(окончание)*

Тип Delphi	Тип BCB	Диапазон	Формат
Символьные (строковые) данные			
`Char, AnsiChar`	`char`	8-битовый символ	8 бит
`WideChar`	`wchar_t`	16-битовый символ (Unicode)	16 бит
`String[n]`	`SmallString<n>`	Строка до 255 символов	
`String, AnsiString`	`AnsiString`	Длинная строка до 2 Гбайт	
`PChar`	`unsigned char *`	Указатель на строку	32 бита
Указатель без типа			
`Pointer`	`void *`		32 бита
Календарные даты и время			
`TDateTime`	`TDateTime`		64 бита

Глава 2

С чего начать



2.1. Знакомство с IDE

Интегрированная среда разработки IDE (Integrated Development Environment) — это рабочее место и набор инструментов, участвующих в создании программы. IDE в MS-DOS по сравнению с IDE в Windows выглядит как золушка рядом с принцессой, хотя функциональное назначение у них одно и то же. Более того, вы увидите много знакомых слов в обозначениях разделов и команд главного меню и встретите похожие термины — окно редактора программы, окна отладчика, контекстная помощь и др. Однако в графической среде Windows все это выглядит намного красивее. Идеи визуального программирования пополнили процесс построения программ технологией сборки, напоминающей игру с детским конструктором. Здесь даже прижилась инженерная терминология — *проектирование (конструирование) программы*. А в названиях отдельных окон сплошь и рядом фигурируют дизайнеры, менеджеры и тому подобные профессии.

Очутившись в IDE, вы вправе перемещать составные части среды, изменять их структуру, размеры и цвет, поступая примерно так же, как новосел, обустраивающий свою квартиру. Злоупотреблять этим правом особенно на первых порах не стоит, однако по мере освоения тех или иных возможностей системы вы настроите IDE по своему вкусу. А сначала попробуем разобраться в главных элементах IDE.

Основные элементы визуальной среды появляются на экране дисплея после запуска соответствующей среды программирования (рис. 2.1 и 2.2). Скрытые и вспомогательные окна, количество которых доходит до десятка, могут быть вызваны по мере необходимости.

В разных версиях и редакциях визуальной среды эти картинки могут отличаться от приведенных, да и пользователь вправе изменить расположение и размеры тех или иных элементов. Однако, главное не в этом. Наиболее

важные компоненты, всегда участвующие в конструировании приложения, в рассматриваемых средах имеют функционально сходное назначение и достаточно близки по внешнему виду. В их составе:

- главное меню среды с панелью быстрых кнопок для исполнения наиболее часто используемых команд;
- форма конструируемого приложения;
- хранилище визуальных компонентов, используемых при создании приложения;
- окно компонентов, включенных в состав приложения, и значения их свойств.

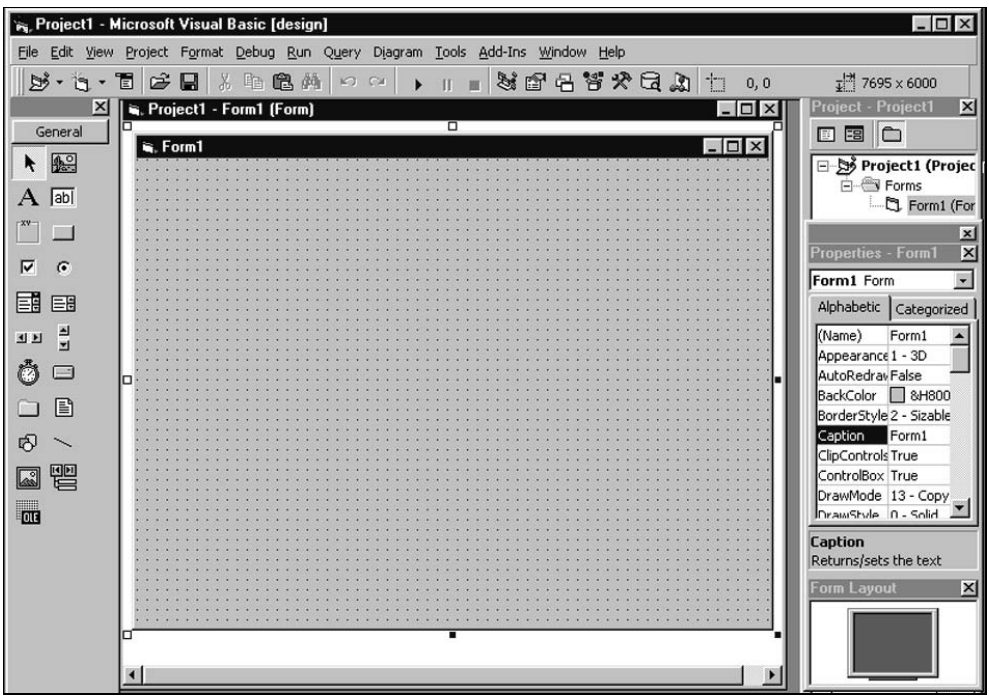


Рис. 2.1. Общий вид визуальной среды VB-6.0

Одним из наиболее важных элементов визуальной среды является окно редактора программы (в большинстве книг его принято называть окном редактора кода). Обычно это окно закрыто формой создаваемого приложения, и появляется оно на переднем плане в тот момент, когда пользователь вызывает его соответствующей командой меню или щелчком мыши сообщает о своем желании набрать текст программы обработки того или иного события. Внешний вид окна редактора программы среды VB приведен на рис. 2.3. Оно немного отличается от аналогичного редактора кода в среде

BCB/Delphi (рис. 2.4) тем, что нагружено двумя выпадающими списками, используемыми для переключения на тот или иной компонент приложения и указания интересующего нас события.

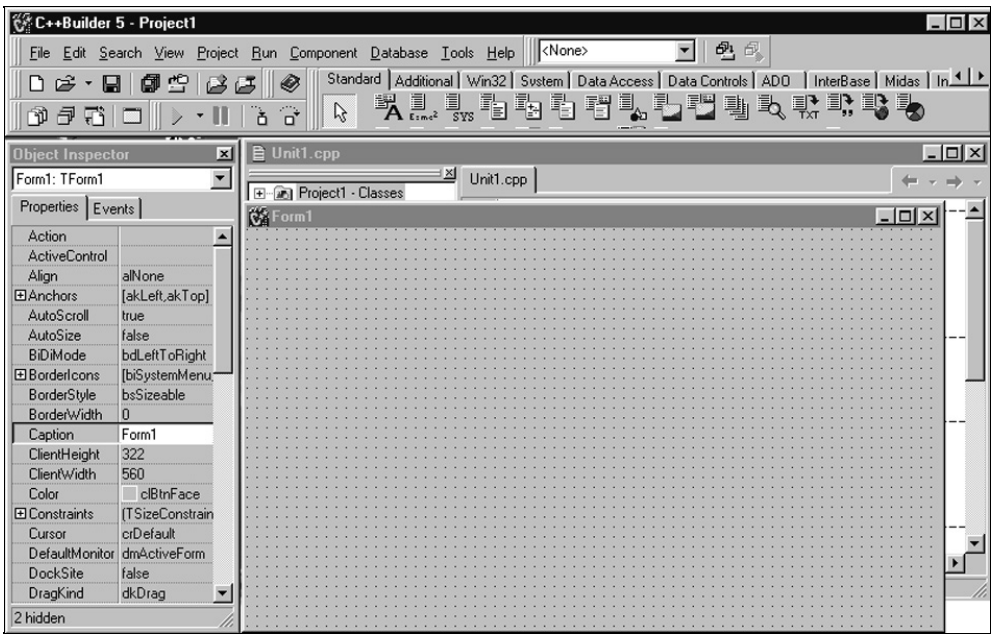


Рис. 2.2. Общий вид визуальной среды BCB/Delphi

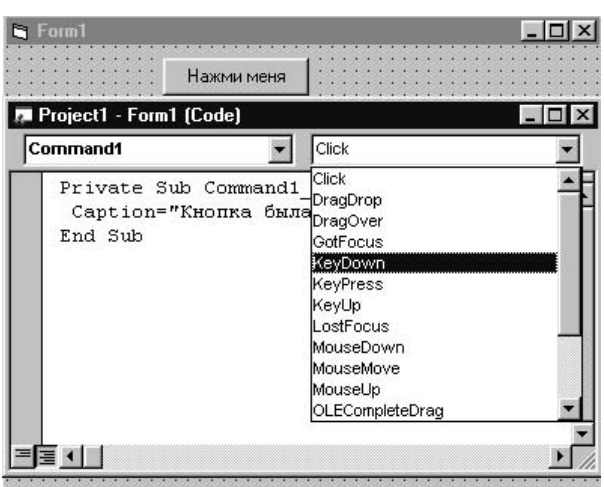


Рис. 2.3. Окно редактора программы в среде VB-6.0

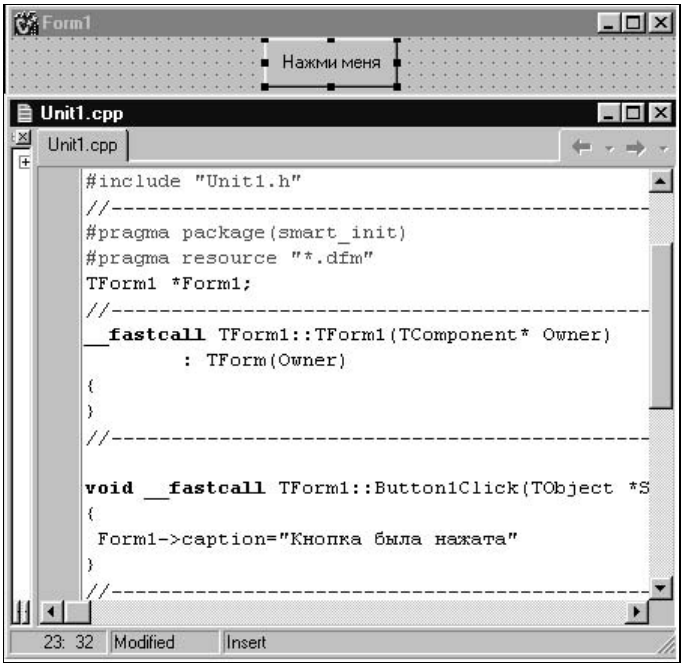


Рис. 2.4. Внешний вид окна редактора программы в среде BCB

Работа в поле редактора программы мало чем отличается от техники набора текстов в таком общеизвестном офисном пакете как MS Word. На поле редактора можно вызвать любой текстовый файл и запомнить отредактированный документ на диске. Здесь можно выделять, вырезать и копировать фрагменты, использовать *Clipboard*. Однако нет необходимости манипулировать параметрами шрифтов. Определенное количество строк в текст программы система вставляет автоматически. Без крайней необходимости не надо такие строки удалять. Может оказаться, что вы набрали текст обработки не интересующего вас события. В этом случае можно удалить набранный вами текст, но не трогайте окаймляющие строки, которые появились автоматически. Получится пустая процедура (функция), которую система удалит при очередном сохранении текста программы.

Одним из приемов работы в визуальной среде является использование готовых компонент и настройка их свойств под нужды нашего приложения. Хранилище компонентов, из которого можно извлечь нужный элемент управления и поместить его на форме нашего будущего приложения, в средах VB, BCB/Delphi устроено по-разному.

Visual Basic предпочитает не загромождать экран пиктограммами всех своих компонент и разместил порядка 20 наиболее употребительных элементов в окне с заголовком **General** (Общие), расположенным в левой части экрана.

Однако общее количество компонент, включенных в профессиональную редакцию VB 6.0, достигает 200. Для подключения нужного объекта из общего хранилища в его видимую часть необходимо войти в позицию главного меню **Project** (Проект) и выполнить команду **Components** (Компоненты). В появившемся диалоговом окне (рис. 2.5) на вкладке **Controls** (Элементы управления) выбрать строку с обозначением интересующего вас элемента или группы элементов (на рисунке выбрана строка с компонентом **Masked Edit**) и щелчком мыши установить галочку в соответствующем квадратике слева. После нажатия кнопки **OK** новый компонент дополнит стандартный набор. Этим набором сможет пользоваться только данное приложение, в противном случае вновь добавляемые к разным приложениям компоненты будут засорять экран. На рис. 2.6 слева расположено стандартное окно **General** (Общие), а справа — оно же, пополненное компонентом **Masked Edit** (Редактирование по маске).

Следует отметить, что авторы VB сохранили вид пиктограмм главных элементов управления такими, какими они появились в первой версии. Это большой плюс в преемственности поколений программного обеспечения.

Авторы Delphi и BCB пошли по другому пути. Все компоненты, зарегистрированные в системе, вынесены на полоску с большим числом вкладок. Каждая из вкладок выдвигает на передний план линейку с пиктограммами компонент соответствующего раздела. В ранних версиях такая компоновка была достаточно наглядной, т. к. количество элементов раздела редко превышало 8–9. Однако в современных версиях общее количество компонент достигает 150–200, и на каждой планке пришлось добавить кнопки горизонтальной прокрутки, чтобы добраться до нужного элемента (рис. 2.7 и 2.8). В отличие от авторов VB создатели более поздних версий BCB пошли на изменение вида пиктограмм, что создает дополнительные неудобства для пользователей.

Перенос любой компоненты из ее хранилища на форму будущего приложения в визуальных средах производится одинаково. Самый быстрый способ заключается в двойном щелчке по пиктограмме выбранной компоненты. После этого аналогичное или эквивалентное изображение появляется в центре нашей формы. Остается только с помощью мыши растянуть элемент до нужных габаритов и "отбуксировать" на предполагаемое место жительства. Второй способ заключается в одинарном щелчке по нужной пиктограмме в хранилище с целью выделить компонент. Затем курсор мыши переводится в нужную точку формы, зажимается клавиша мыши, и изображение элемента растягивается до нужных размеров. Существуют и другие способы размножения элементов, уже выбранных на форму с использованием буфера обмена (Clipboard).

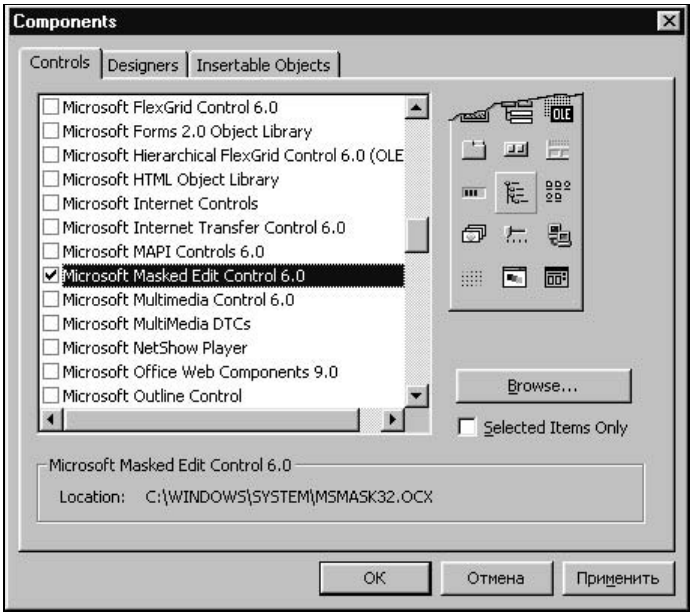


Рис. 2.5. Подключение компонента из общего хранилища к окну **General**

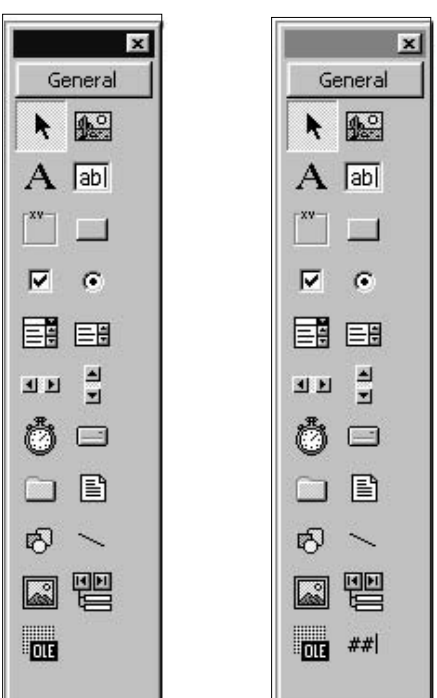


Рис. 2.6. Окна компонент VB