Оглавление

Введение

Microsoft Visual Basic 2005 – это мощная система визуального проектирования, предназначенная для создания программ, работающих в операционной системе Windows. Visual Basic 2005 не является самостоятельной системой разработки, а входит в состав Microsoft Visual Studio 2005. В основе Visual Basic 2005 лежит современный диалект языка Basic[1] – Visual Basic. Система проектирования Visual Basic 2005 позволяет разрабатывать приложения любой степени сложности для решения задач различных предметных областей. При этом интерфейс приложений полностью удовлетворяет всем требованиям Windows.

Visual Studio 2005 является средой визуального программирования. Она позволяет разработчику прямо на экране конструировать интерфейс приложения, используя стандартные компоненты.

Visual Basic 2005 поддерживает событийно-управляемое программирование. При таком подходе разработка программы заключается в создании фрагментов программных кодов, каждый из которых связывается с определенным событием. Событие – это любое действие пользователя: щелчок кнопкой мыши, ввод текста, изменение размеров или положения окна и т.д. В результате создается не одна большая программа, а приложение, состоящее из набора взаимосвязанных подпрограмм. После запуска приложение переходит в состояние ожидания события. При наступлении какого-либо события вызывается соответствующая подпрограмма – обработчик события. Затем приложение опять переходит в состояние ожидания события. Таким образом, между отдельными частями приложения нет жесткой связи. Последовательность их вызова определяется последовательностью наступления тех или иных событий.

1. Этапы создания Windows-приложения

Процесс создания Windows-приложения состоит из пяти основных этапов.

1. Постановка задачи. На этом этапе составляется словесное описание того, как должна работать будущая программа, и что должен делать пользователь в процессе

[1] Название языка BASIC является аббревиатурой. Оно образовано из первых букв словосочетания Beginner's All-purpose Symbolic Instruction Code (универсальный язык программирования для начинающих).

Оглавление

ее работы. Определяются исходные данные для программы и форма представления результатов ее работы.

2. Разработка интерфейса. На этом этапе создаются окна приложения (экранные формы), определяется, какие элементы управления (объекты) будут находиться на форме, и какими свойствами они будут обладать.
3. Программирование. Здесь выбираются события, на которые будет реагировать приложение, составляются алгоритмы обработки этих событий и разрабатываются программные коды, реализующие эти алгоритмы.
4. Отладка. На этом этапе выполняется поиск и устранение ошибок в составленных программах. Целью отладки является достижение работоспособности приложения. Иными словами, полностью отлаженное приложение работает точно так, как было определено на этапе постановки задачи.
5. Компиляция. Последний этап разработки Windows-приложения. Здесь разрабатываемая программа преобразуется в исполняемый файл, который может быть запущен за пределами среды проектирования Visual Studio 2005.

2. Среда Visual Basic 2005

Среда разработки программ Visual Basic 2005 содержит все необходимые инструменты для проектирования, запуска и отладки программ. Она позволяет упростить процесс создания программных приложений. Среда Visual Basic 2005 включает в себя средства построения интерфейса программ, редактор кодов, отладчик, а также множество дополнительных инструментов.

Приложение, разработанное в среде Visual Basic 2005, называется *проектом*. Проект включает в себя информацию об интерфейсе программы и программные коды для обработки различных событий.

2.1. Структура среды Visual Basic 2005

Для запуска Visual Basic 2005 необходимо выбрать пиктограмму Visual Basic 2005 из меню «Пуск», осуществив такую последовательность действий: Программы → Microsoft Visual Studio 2005 → Microsoft Visual Studio 2005. После этого на экране появится стартовое окно среды разработки (рис. 1). Оно позволяет открыть один из последних проектов, находившихся в разработке (блок Recent Project расположен в левом верхнем углу стартового окна), открыть проект (команда Open Project находится там же) или создать новый проект (команда Create Project находится в том же блоке). Также это окно позволяет обратиться к разделам справочной системы, посвященным

Оглавление

первоначальному ознакомлению со средой разработки (блок Getting Started расположен по центру левой части стартового окна) и ознакомиться с заголовками новостей в области разработки программного обеспечения (блок MSDN[2] расположен в правой части стартового окна и занимает его основную часть).

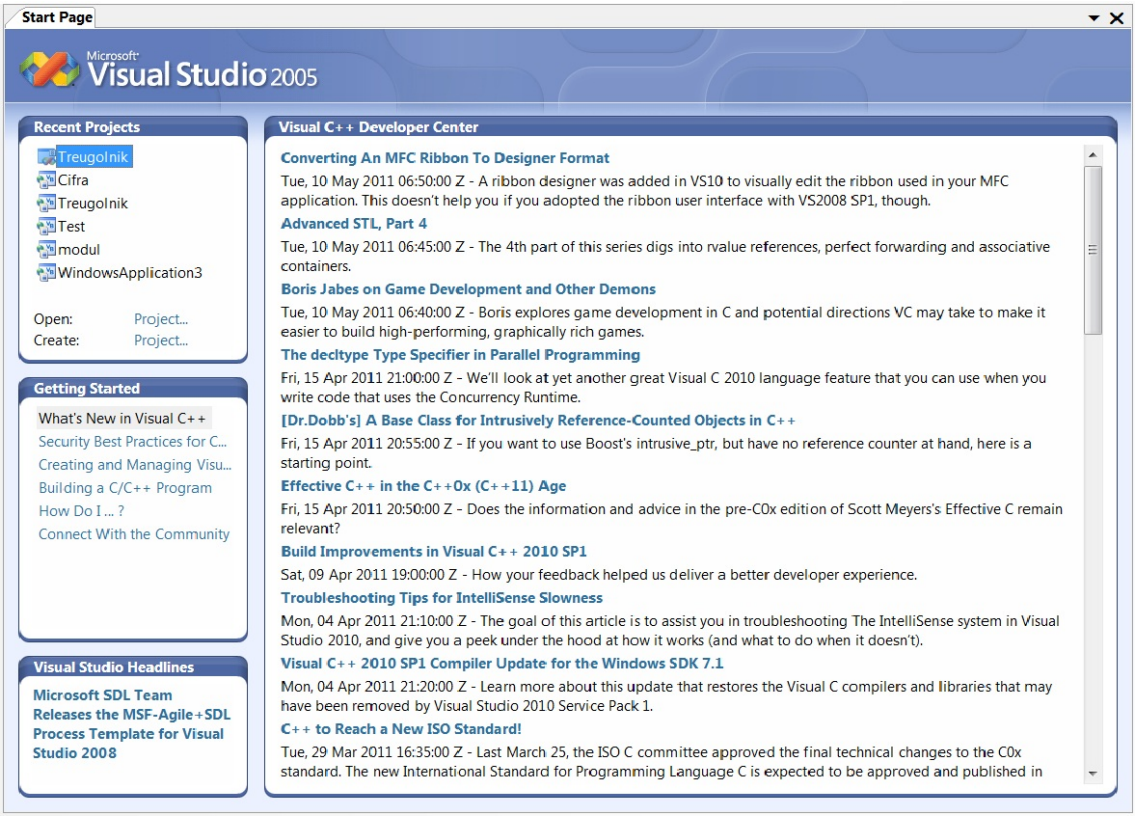


Рис. 1. Стартовое окно Microsoft Visual Studio 2005

Если стартовое окно не нужно, то его закрывают, нажав на крестик в верхнем правом углу окна. После этого открываются основные окна среды Visual Basic 2005 (рис. 2). При различных настройках среды Visual Basic 2005 набор и расположение этих окон могут отличаться от представленных на рисунке.

[2] MSDN – Microsoft Developer Network – подразделение компании Microsoft, ответственное за взаимодействие фирмы с разработчиками программного обеспечения, работает как информационный сервис для разработчиков.

Оглавление

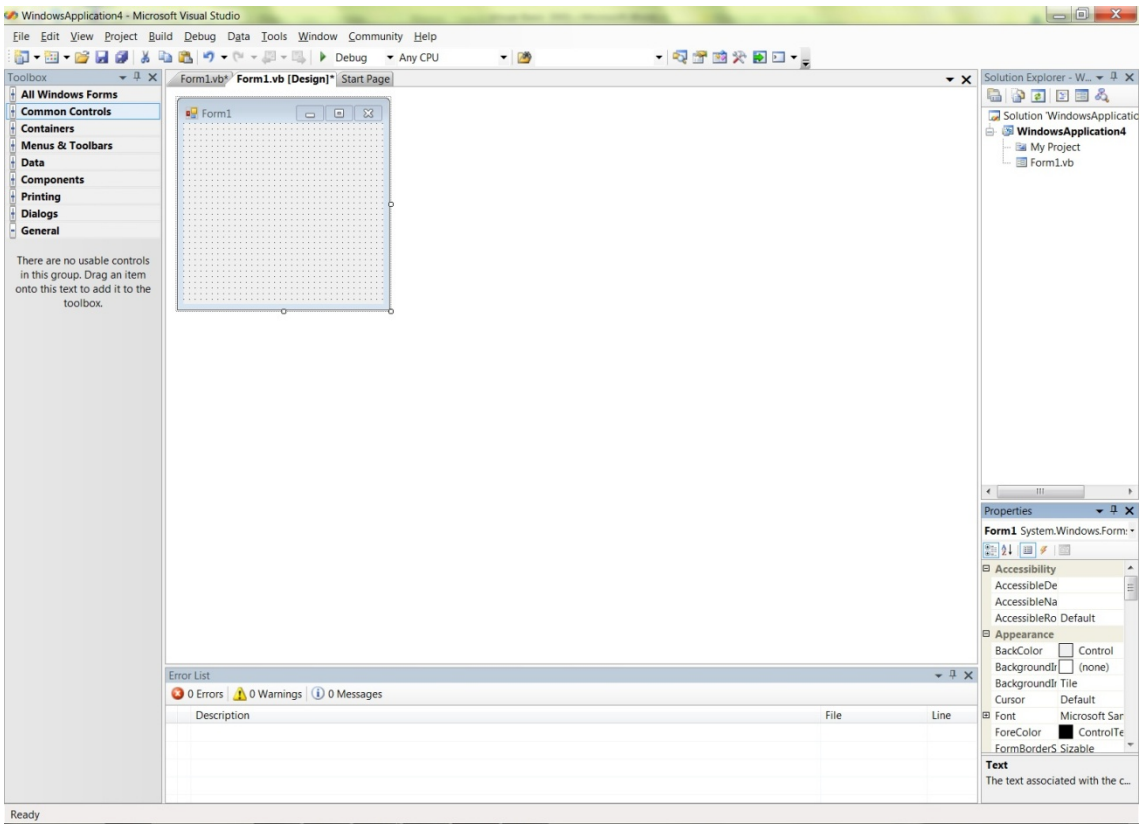


Рис. 2. Основные окна среды Visual Basic 2005

Рассмотрим основные окна среды Visual Basic 2005.

Главное меню (рис. 3) расположено в верхней части экрана под заголовком окна и содержит все команды среды Visual Basic 2005. Основные блоки главного меню:

- File – содержит команды для организации доступа к файлам, позволяет открывать, сохранять и закрывать файлы и проекты;
- Edit – содержит стандартные команды редактирования: отменить, вырезать, скопировать, вставить, а также команды поиска и замены;
- View – содержит команды для вызова окон среды Visual Basic 2005, позволяет открывать окна редактора программного кода, конструктора формы и другие окна;
- Project – содержит команды, позволяющие добавлять в проект новые элементы (формы, модули и проч.) и удалять их;
- Build – содержит команды, позволяющие компилировать и компоновать проекты.
- Debug – содержит команды, предназначенные для отладки и запуска приложения;
- Data – содержит команды для организации работы с базами данных;
- Format – содержит команды, управляющие выравниванием текста и элементов управления, позволяет задавать размеры элементов управления и интервалы между ними. Этот пункт меню доступен только при работе в конструкторе форм.

Оглавление

- Tools – содержит средства для настройки среды разработки, создания макросов, а также команды запуска дополнительных утилит;
- Window – содержит команды для управления открытыми на экране окнами, позволяет упорядочивать, активизировать и скрывать окна, а также переходить из одного окна в другое.
- Community – позволяет связаться с разработчиками Microsoft Visual Studio 2005 в интерактивном режиме, обсудить вопросы разработки на форуме, а также осуществить поиск примеров и дополнительных элементов управления;
- Help – содержит команды для работы со справочной системой.

Рис. 3. Главное меню

Панель инструментов (рис. 4) расположена сразу под главным меню. Она содержит кнопки, дублирующие наиболее часто используемые команды главного меню, что позволяет повысить эффективность работы в среде Visual Basic 2005. По умолчанию она всегда присутствует в главном окне среды Visual Basic 2005, если она была удалена ранее. Для того чтобы восстановить на экране панель инструментов, надо из пункта меню **View** выбрать команду **Toolbars → Standart**. После установки панель инструментов находится в верхней части главного окна, но она, как и все другие панели инструментов, может быть перемещена в любое место экрана.

Рис. 4. Панель инструментов

Окно конструктора форм (рис. 5) является основным рабочим окном, в котором выполняется визуальное проектирование интерфейса приложения. Вызвать это окно можно из главного меню: **View → Designer**. Другой способ – дважды щелкнуть мышью по имени формы в окне проводника проекта (решения) **Solution Explorer.** В окне форм визуально создаются все формы приложения. В начале работы экранная форма пуста. В процессе проектирования интерфейса на ней располагаются различные элементы управления. Для их точного расположения можно использовать сетку или команды из меню **Format.**

Оглавление

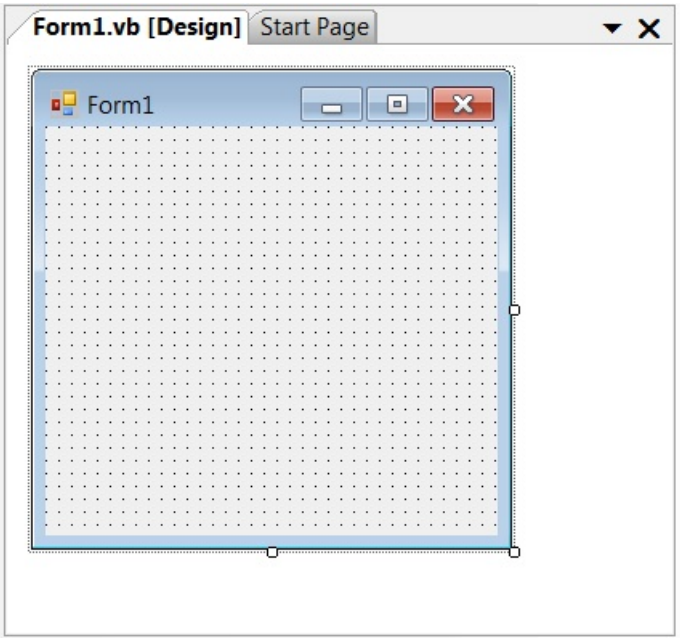


Рис. 5. Окно конструктора форм

Панель элементов управления (рис. 6) – основной рабочий инструмент при визуальной разработке интерфейса приложения (экранных форм). По умолчанию панель элементов управления находится в левой части главного окна среды Visual Basic 2005. Если панель инструментов недоступна, ее можно открыть с помощью команды **Toolbox** из меню **View**. Как только указатель мыши покидает окно панели элементов управления, она исчезает с экрана. Чтобы закрепить панель элементов управления на экране, надо нажать на среднюю из трех кнопок, расположенных в правом верхнем углу окна. Эта кнопка называется Auto Hide ( ).

Панель элементов управления состоит из различных разделов, в которых расположены элементы управления, доступные в данный момент разработчики. Рассмотрим эти разделы.

- **All Windows Forms** (Все элементы управления) – в этом блоке собраны все доступные элементы управления. Как правило, элементы управления расположены в алфавитном порядке.
- **Common Controls** (Основные элементы управления) – в этом блоке хранятся наиболее часто используемые элементы управления.
- **Containers** (Контейнеры) – блок объединяет элементы управления, которые могут содержать в себе другие элементы управления.

Оглавление

- **Menus & Toolbars** (меню и панели инструментов) – содержит такие элементы управления как обычное и контекстное меню, панели инструментов и строка состояния.
- **Data** (Данные) – в этом блоке собраны элементы управления, предназначенные для организации доступа к данным и источникам данных.
- **Components** (Компоненты) – в этом блоке хранятся элементы, которые позволяют выполнять мониторинг файловой системы, запись информации об ошибках, возникающих в процессе выполнения приложения и так далее.
- **Printing** (Печать) – содержит элементы управления, которые используются для организации печати.
- **Dialogs** (Диалоговые окна) – блок объединяет стандартные диалоговые окна: открытия и сохранения файла, настройки шрифта и цвета.
- **General** (Общие) – в этом блоке расположены специальные управляющие элементы.

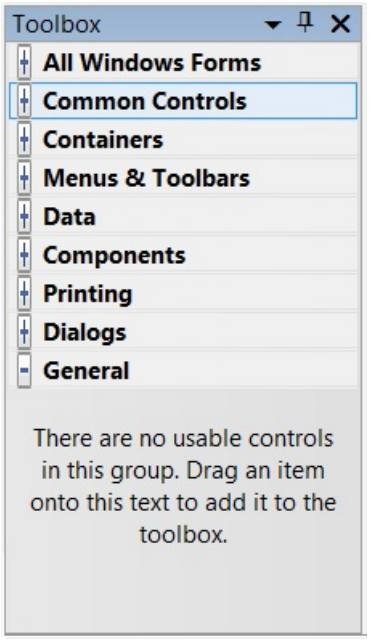


Рис. 6. Панель элементов управления

Окно свойств (рис. 7) предназначено для отображения и настройки свойств объектов. В нем отображаются все выбранного объекта, включая положение на форме, геометрические размеры, цветовое и шрифтовое оформление. Объектами в среде Visual Basic 2005 является все: экранная форма, элементы управления, файлы и так далее. Каждый объект имеет свое собственное уникальное имя, которое задается

программистом в соответствии с правилом имен. Оно состоит из шести следующих пунктов.

1. В имени можно использовать буквы[3], цифры и знак подчеркивания.
2. Первым символом имени должна быть буква.
3. Остальные символы имени – буквы, цифры и знак подчеркивания.
4. Имя не должно содержать пробелы, скобки, знаки препинания и математических операций.
5. Длина имени не должна превышать 255 символов.
6. Имя не должно совпадать ни с одним ключевым словом Visual Basic 2005.

Свойство объекта – это качественная или количественная характеристика объекта. Разные объекты обладают разным набором свойств.

Открыть окно свойств можно двумя различными способами.

1. Щелкнуть правой кнопкой мыши на нужном элементе управления и выбрать из контекстного меню команду **Properties**.
2. Выбрать команду **Properties Window** из меню **View.**

В верхней части окна расположен раскрывающийся список, в котором в алфавитном порядке приведены все элементы управления, находящиеся на форме, включая саму экранную форму. Имя объекта выводится полужирным шрифтом, а его тип – обычным. Свойства выбранного объекта представляются в окне свойств в виде таблицы, состоящей из двух колонок. В левой колонке перечислены свойства объекта, а в правой – их значения. Свойства могут быть перечислены в алфавитном порядке (кнопка **Alphabetical**  **)** или собранны в группы по категориям (кнопка **Categorized** ). Для того чтобы изменить значение какого-либо свойства надо щелкнуть левой кнопкой мыши по значению свойства и в этой же строке ввести новое значении свойства. Ввод заканчивается нажатием клавиши **Enter.**

В нижней части окна появляется подсказка, поясняющая значение выбранного свойства объекта.

[3] Допускается использование в именах как латинских, так и русских букв. Но желательно ограничиться только латинским алфавитом, чтобы уменьшить количество ошибок в программе.

Оглавление

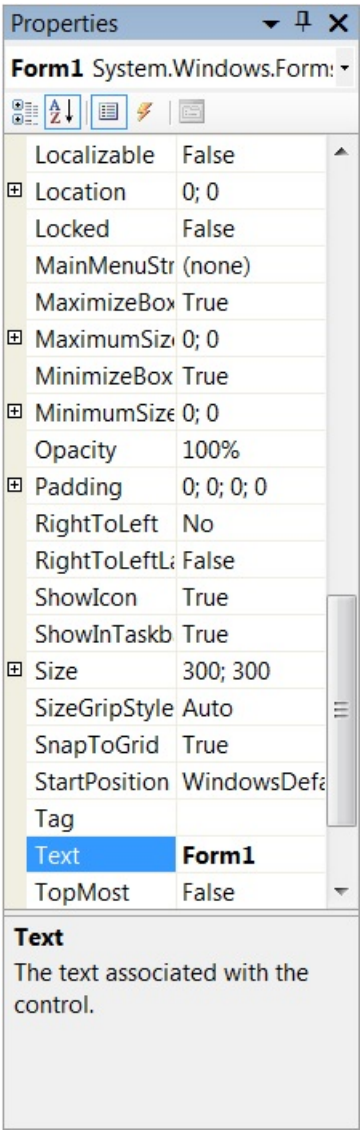


Рис. 7. Окно свойств

Окно проводника проекта (решения) (рис. 8) предназначено для организации доступа к компонентам, входящим в состав проекта (решения). Открыть это окно можно двумя способами.

1. Выбрать команду **Solution Explorer** из меню **View**.
2. Нажать кнопку **Solution Explorer** на стандартной панели инструментов ( ).

Проводник проекта (решения) содержит иерархическую структуру всех компонентов проекта (решения): форм, модулей и прочих файлов. Для того чтобы открыть любой компонент проекта (решения) надо дважды щелкнуть по нему левой кнопкой мыши. Каждый компонент открывается в соответствующем окне.

Оглавление

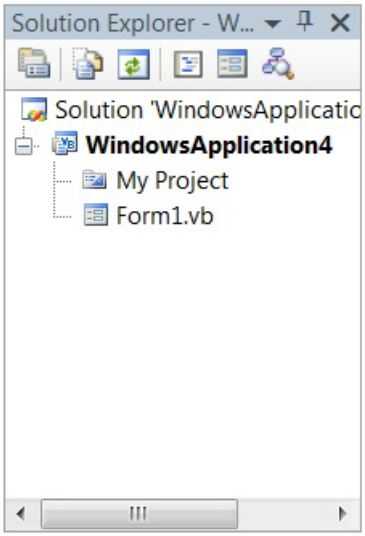


Рис. 8. Окно проводника проекта (решения)

Окно программного кода (рис. 9) – это окно редактора, в котором создается программный код. Редактор программного кода – это мощный текстовый редактор, с большим количеством функциональных возможностей, который является основным инструментов программиста.

В верхней части окна расположены два раскрывающихся списка. В левом списке находится перечень всех объектов приложения. В правом списке для выбранного объекта перечислены события, которые могут с ним произойти. Последовательный выбор объекта и связанного с ним события приводят к созданию шаблона (заготовки) процедуры обработки этого события.

Редактор программного кода можно открыть двумя способами.

1. Выбрать команду **Code** из меню **View**.
2. Дважды щелкнуть левой кнопкой мыши по соответствующему элементу правления.

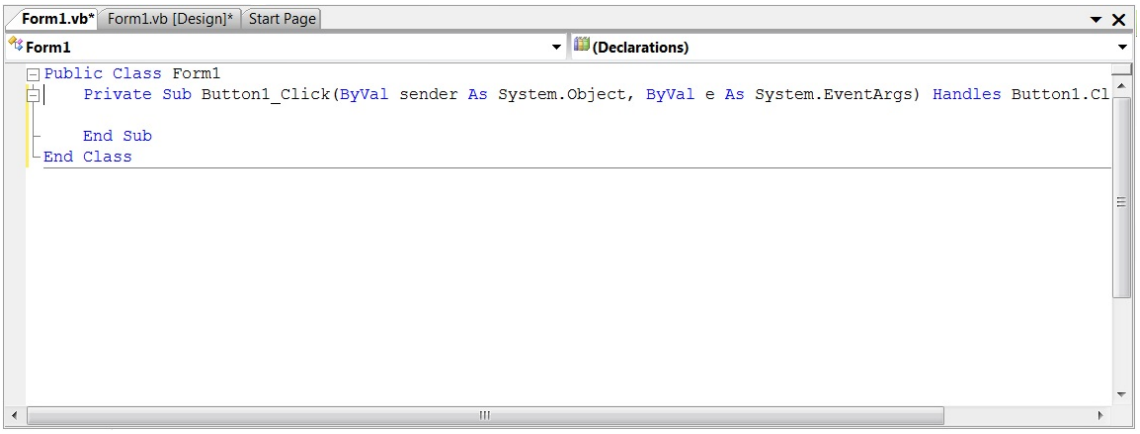


Рис. 9. Окно программного кода

Оглавление

Окно списка ошибок (рис. 10) – в этом окне отображаются сообщения о синтаксических ошибках, которые обнаружил Visual Basic 2005. Здесь же выводятся предупреждения о потенциально некорректных моментах в разрабатываемой программе. Например, сообщения о неиспользуемых переменных. Для вывода этого окна надо выбрать команду **Error List** из меню **View**.

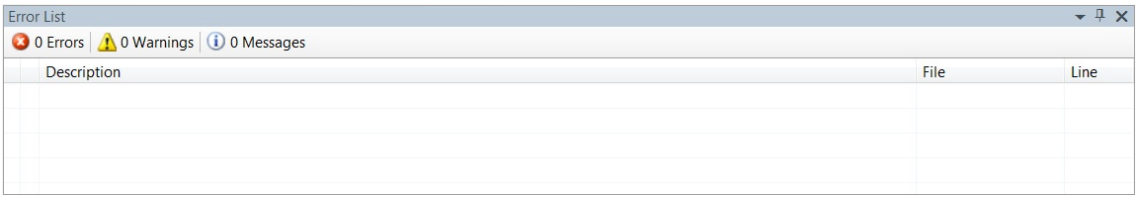


Рис. 10. Окно списка ошибок

2.2. Создание нового проекта

Создание любого приложения в Visual Basic 2005 начинается с создания проекта. ***Проектом*** называется совокупность файлов, входящих в состав приложения и хранящих информацию о его компонентах. Чтобы создать новый проект необходимо выполнить следующие действия.

1. Запустить Visual Basic 2005.
2. Выбрать команду **Create Project** из стартового окна **Start Page** или, закрыв стартовое окно, выбрать команду **New Project** из меню **File.**
3. Открывшееся окно создания нового проекта (рис. 11) состоит из двух частей. В левой части надо выбрать тип проекта (**Project Types**). Здесь определяется язык программирования, на котором будет написан программный код будущего проекта. В этом окне надо выбрать язык **Visual Basic** и указать тип приложения – **Windows**. По строке **Windows** необходимо щелкнуть левой кнопкой мыши. В правой части окна выводится список шаблонов приложений (**Templates**). В этом списке надо выбрать шаблон **Windows Application**.
4. В поле **Name** надо указать имя будущего проекта. Имя следует записывать без пробелов, скобок и знаков препинания. Другими словами, имя проекта должно подчиняться правилу имен. Как правило, проект хранится в папке, имя которой совпадает с именем проекта.
5. В поле **Location** указывается место физического расположения папок проекта на жестком диске.

Оглавление

6. Если рядом со словами **Create directory for solution** стоит галочка, ее нужно снять.
7. После заполнения всех полей надо нажать кнопку **OK**.

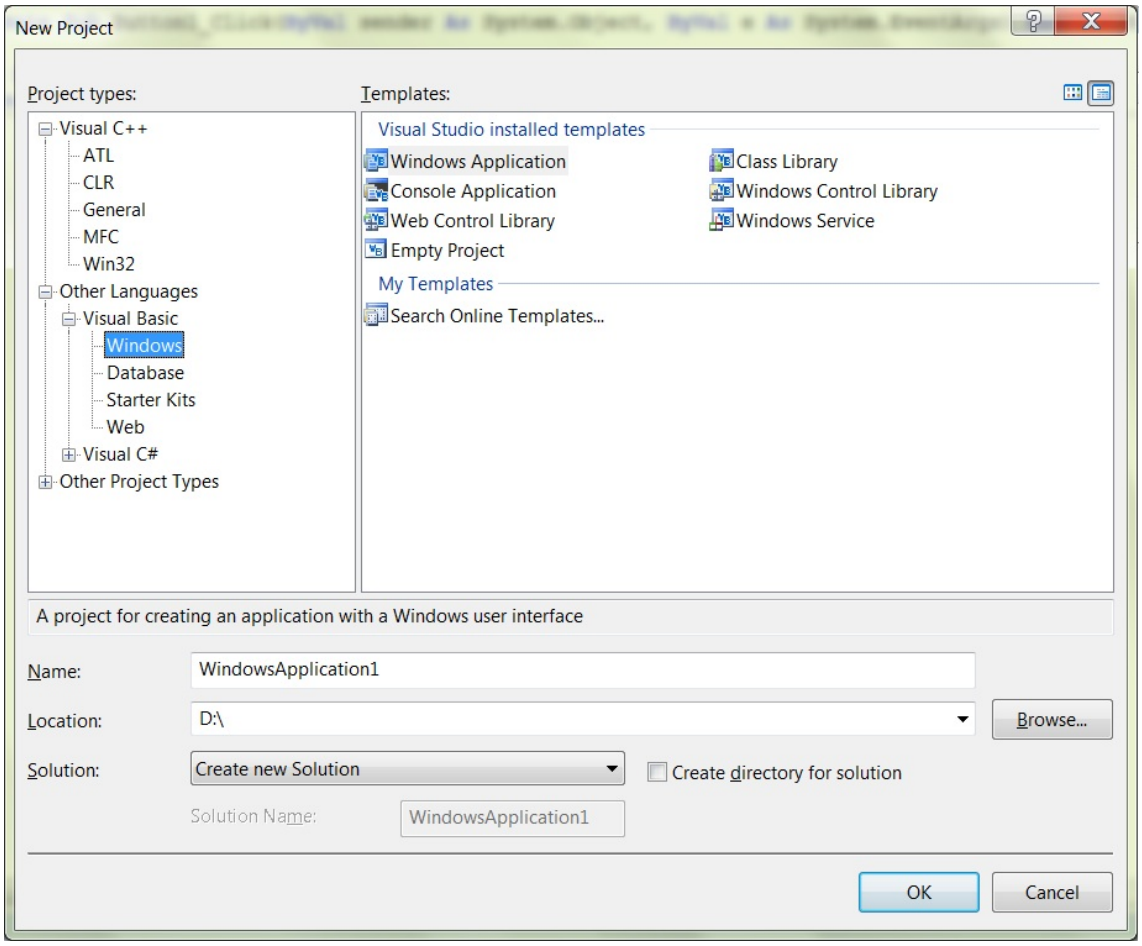


Рис. 11. Окно создания нового проекта

2.3. Сохранение проекта

Для сохранения разработанного проект используются команды **Save Project** или **Save All** из меню **File**. Команду **Save All** можно также вызвать при помощи одноименной кнопки, находящейся на стандартной панели инструментов ( ). При первом сохранении проекта может появляться специально диалоговое окно (рис. 12), в котором надо указать имя проекта (**Name**) и место его физического расположения на диске (**Location**). Поле **Solution name** надо оставить пустым и убрать галочку, стоящую рядом со словом **Create.** После заполнения всех полей надо нажать кнопку **OK.**

Каждый проект следует сохранять в отдельной папке. В результате сохранения в папке проекта создаются три новых папки (Bin, My Project, Obj) и пять отдельных файлов. Три для описания формы и два для описания проекта. При переносе проекта с

Оглавление

одного компьютера на другой необходимо скопировать папку My Project и все пять отдельных файлов. Папки Bin и Obj являются необязательными.

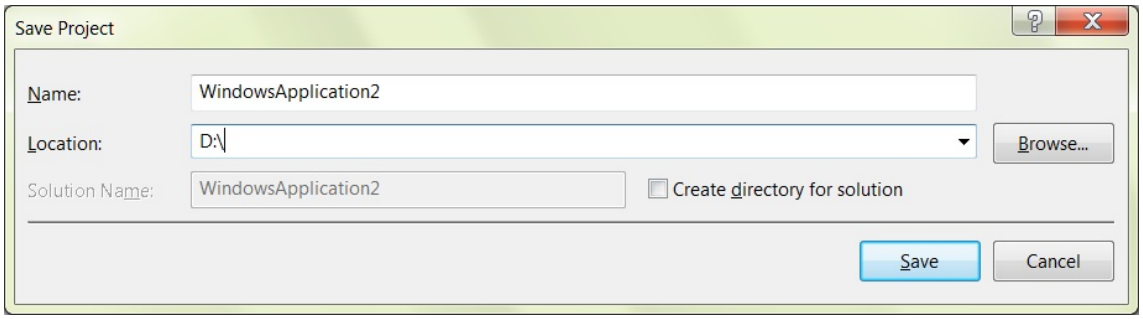


Рис. 12. Окно сохранения проекта

2.4. Выполнение приложения

Существует несколько способов запустить выполнение разработанного приложения.

- Нажать кнопку **F5**.
- Выбрать команду **Start Debugging** из меню **Debug**.
- Нажать на кнопку **Start Debugging** на стандартной панели инструментов.

Если проект не содержит ошибок, то он сразу начнет выполняться, и поверх окна Visual Basic появится окно разработанного приложения. Если в программном коде есть ошибки, то перед выполнением появится диалоговое окно (рис. 13). Проект можно попытаться выполнить и при наличии ошибок. Для этого необходимо нажать кнопку **Yes**. Если принято решение исправить ошибки до запуска, то следует нажать кнопку **No**.

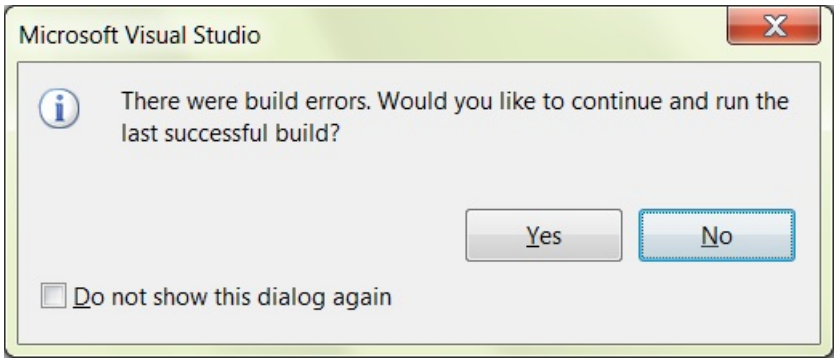


Рис. 13. Окно сообщения об ошибках, обнаруженных перед выполнением программы

Оглавление

2.5. Основные команды среды Visual Basic 2005

Рассмотрим основные команды главного меню и соответствующие им кнопки стандартной панели инструментов (табл. 1).

Таблица 1

Действие	Пункт меню → Команда	Пиктограмма	Описание
Создание нового проекта	File → New Project		Создание нового проекта, состоящего из одной формы
Открытие ранее созданного проекта	File → Open Project		Вывод универсального диалогового окна для открытия ранее созданного проекта.
Закрыть текущую страницу	File → Close	Нет	Закрывает текущую страницу. Если на ней есть несохраненная информация, то задается вопрос, надо ли ее сохранять.
Закрыть проект	File → Close Project		Закрывает проект целиком. Если в одном из файлов проекта есть несохраненная информация, то задается вопрос, надо ли ее сохранять.
Сохранить текущий файл	File → Save		Сохранение файла, из активного окна.
Сохранить все	File → Save All		Сохранение проекта и всех его элементов (форм и модулей).
Выход из среды	File → Exit	Нет	Завершение работы в среде Visual Basic 2005.
Открыть окно редактора программного кода	View → Code		Выводит на экран окно редактора программного кода.
Открыть окно конструктора форм	View → Designer		Выводит на экран окно для разработки интерфейса приложения.
Открыть окно проводника проекта (решения)	View → Solution Explorer		Выводит на экран окно проводника проекта (решения), позволяющее обращаться к отдельным частям проекта.

Оглавление

Действие	Пункт меню → Команда	Пиктограмма	Описание
Открыть окно списка ошибок	View → Other Windows → Error List		Выводит на экран окно со списком ошибок и предупреждений о возможных некорректных фрагментах программного кода
Открыть окно свойств	View → Other Windows → Properties window		Выводит на экран окно, содержащее таблицу свойств и позволяющее настраивать свойств объектов.
Открыть панель элементов управления	View → Toolbox		Выводит на экран панель элементов управления, с помощью которой проектируется интерфейс приложения.
Открыть стандартную панель инструментов	View → Toolbars → Standard	Нет	Выводит на экран стандартную панель инструментов, обеспечивающую доступ к основным командам среды Visual Basic 2005.
Добавить к проекту экранную форму	Project → Add Windows Form		Выводит на экран диалоговое окно, позволяющее выбрать тип экранной формы, которая будет добавлена к проекту.
Добавить к проекту новый модуль	Project → Add Module		Выводит на экран диалоговое окно, в котором задается имя и характеристики добавляемого модуля.
Удалить элемент из проекта	Project → Exclude From Project	Нет	Удаляет активный элемент (форму или модуль) из проекта. Активным считается элемент, отображенный в открытом окне.

Оглавление

Действие	Пункт меню → Команда	Пиктограмма	Описание
Создание исполняемого файла	Build → Build		Осуществляет проверку синтаксической правильности всех форм и модулей, входящих в проект. Если в проекте нет ошибок, то создается исполняемый файл.
Запуск программы	Debug → Start Debugging		Проводится проверка синтаксической правильности всех форм и модулей, входящих в проект. Если в проекте нет ошибок, то он запускается.
Остановка выполнения программы	Debug → Stop Debugging		Позволяет в любой момент остановить выполнение проекта, даже если программа попала в бесконечный цикл.
Пошаговое выполнение программы	Debug → Step Into		Пошаговое выполнение программы с заходом во все подпрограммы.
	Debug → Step Over		Пошаговое выполнение программы без захода в подпрограммы. Подпрограммы выполняются, но их пошаговое выполнение не демонстрируется.
Задать точку остановки при выполнении программы	Debug → Toggle Breakpoint	Нет	Программа выполняется непрерывно до заданной точки остановки. На выбранной строке выполнение программы приостанавливается. Для дальнейшего выполнения программы, надо выбрать из пункта меню **Debug** одну из следующих команд: **Continue**, **Step Into** или **Step Over**.

Оглавление

Действие	Пункт меню → Команда	Пиктограмма	Описание
Выравнивание положения элементов	Format → Align	Нет	Блок команд, позволяющих выравнивать положение элементов управления относительно друг друга. Возможные варианты: выравнивание по левому, правому, верхнему и нижнему краям, совмещение горизонтальных или вертикальных центров.
Выравнивание размеров элементов	Format → Make Same Size	Нет	Блок команд, позволяющих выравнивать размеры элементов управления относительно друг друга. Возможные варианты: выравнивание ширины, высоты или одновременно обоих размеров.
Изменение горизонтальных расстояний между элементами	Format → Horizontal Spacing	Нет	Блок команд, позволяющих изменять горизонтальные расстояния между элементами управления. Возможные варианты: увеличить, уменьшить, выровнять или удалить расстояния.
Изменение вертикальных расстояний между элементами	Format → Vertical Spacing	Нет	Блок команд, позволяющих изменять вертикальные расстояния между элементами управления. Возможные варианты: увеличить, уменьшить, выровнять или удалить расстояния.

Оглавление

Действие	Пункт меню → Команда	Пиктограмма	Описание
Центрирование элементов на форме	Format → Center in Form	Нет	Блок команд, позволяющих отцентрировать отдельный элемент управления или их группу на форме. Возможно центрирование по вертикали и по горизонтали.
Изменение настроек среды Visual Basic 2005	Tools → Options	Нет	Выводит на экран диалоговое окно изменения настроек среды Visual Basic 2005 и редактора программного кода.

2.6. Методы тестирования

Тестирование – это процесс многократного выполнения программы с целью выявления в ней ошибок. Любой тест должен быть направлен не на подтверждение работоспособности программы, а на выявление в ней наибольшего числа ошибок. Под тестом для программы понимают набор специально подобранных исходных данных и соответствующих им эталонных результатов, используемых для контроля правильности работы программы. Эталонные результаты получают либо с помощью ручного счета, либо, исходя из знаний и понимания специфики предметной области, в которой решается задача. Во всех случаях эталонные результаты вычисляются до начала тестирования.

Существует два основных метода проектирования тестов.

1. Метод «черного ящика». Программа рассматривается как «черный ящик». Важным является только соответствие входной и выходной информации. В этом случае стараются, по возможности, перебрать все варианты исходных данных.
2. Метод «белого ящика». Тесты, разработанные по этому методу, учитывают структуру алгоритма программы. То есть тестируется не только соответствие результатов исходным данным, но и процесс обработки входной информации.

На практике чаще всего используется комбинированный подход, который рекомендует сначала рассматривать программу как «черный ящик», а затем подготовить дополнительные тесты, учитывающие внутреннюю структуру программы.

Оглавление

При подготовке тестов следует пользоваться следующими рекомендациями.

- Следует готовить не только исходные данные для тестов, но и заранее вычислять эталонные результаты. Так как в противном случае можно принять неверные ответы за правильные.
- Важно, чтобы программа давала правильные результаты не только для корректных исходных данных, но и осмысленно реагировала на некорректные значения параметров.
- Составление тестов следует начинать до составления программы. В этом случае легко выявляются ситуации, подлежащие обязательной проверке.
- Составление тестов должно продолжаться параллельно с разработкой программы.

В соответствии с областью входных данных тесты разделяются на четыре класса.

1. Главные или основные тесты – проверяют основные функции программы для наиболее типичных данных. Например, для программы решения квадратного уравнения основным тестом будет уравнение, имеющее два различных корня.
2. Вырожденные тесты – проверяют работу программы в минимальном режиме. Например, для программы решения квадратного уравнения вырожденным тестом будет уравнение, у которого все коэффициенты равны нулю.
3. Тесты граничных значений или предельно допустимые тесты. Эти тесты проверяют работу программы для граничных значений параметров. Например, для программы решения квадратного уравнения это будет уравнение, имеющее два совпадающих корня.
4. Аварийные тесты проверяют реакцию программы на возникновение разного рода аварийных ситуаций, таких как попытка деления на ноль или извлечение квадратного корня из отрицательного числа. Например, для программы решения квадратного уравнения аварийными тестами будут уравнения с отрицательным дискриминантом и уравнения, у которых первый коэффициент равен нулю.

2.7. Отладка приложений в среде VB

Отладка программы – это процесс обнаружения содержащихся в ней ошибок, установления их причин и последующего исправления.

Как правило, ошибки выявляются в ходе тестирования программы, но в процессе отладки программы часто необходимо разрабатывать дополнительные тесты, направленные на локализацию обнаруженных ошибок. В результате корректировки

программы может быть изменена как ее структура, так и взаимосвязь между входными и выходными параметрами. Поэтому после исправления найденных ошибок должны быть подготовлены новые тесты и проведено дополнительное тестирование, так как в ходе исправления ошибок могли быть внесены новые ошибки. Этот процесс повторного тестирования исправленной программы называется регрессионным тестированием.

Ошибки, содержащиеся в программе, можно разделить на две основные группы: синтаксические и семантические.

Синтаксические ошибки заключаются в неправильном использовании операторов и конструкций языка программирования. Как правило, такие ошибки обнаруживаются первыми. Среда Visual Basic 2005 автоматически подчеркивает в программе места, содержащие синтаксические ошибки.

Семантические ошибки – это ошибки в смысловой части обработки данных, часто их называют логическими ошибками. Семантические ошибки нельзя обнаружить в автоматизированном режиме. Для их обнаружения применяют специальные методы и инструменты, входящие в состав среды Visual Basic 2005. К семантическим ошибкам относятся: обращение к несуществующему элементу массива, выполнение недопустимых математических операций (например, извлечение квадратного корня из отрицательного числа), выход за границы допустимых значений для переменных (например, попытка записать в переменную типа Byte число 1000) и многие другие. Семантические ошибки приводят к тому, что синтаксически правильная программа дает неправильные результаты или происходит аварийное завершение программы.

Локализация или поиск семантической ошибки выполняется в несколько этапов.

1. Определение сущности ошибки
2. Выявление фрагмента программы, содержащего ошибку.
3. Определение конкретного места ошибки.

Для выявления сущности ошибки текст программы может и не пригодиться, но необходимы хорошее знание алгоритма, наличие тестовых данных и результатов выполнения программы на этих тестовых данных. Если установить сущность ошибки не удалось, то процесс ее локализации сводится к необходимости обнаружения фрагмента программы, содержащего ошибку. Затем найденный фрагмент выполняется

в пошаговом режиме до обнаружения точного месторасположения семантической ошибки.

Среда Visual Basic 2005 предлагает программисту ряд инструментов, существенно облегающий процесс поиска семантических ошибок:

- точки останова;
- два способа пошагового выполнения программы;
- окна наблюдения за значениями переменных.

Точки останова (Breakpoints) следует располагать в тех местах программы, где необходимо временно приостановить выполнение приложения и проанализировать значения переменных с целью обнаружения ошибки. По умолчанию Visual Basic 2005 выполняет программу непрерывно. Дойдя до точки останова, выполнение останавливается, и на экран выводится окно редактора кода. В этот момент для того чтобы узнать значение любой переменной, достаточно навести на нее курсор мыши, тогда во всплывающем окне появится ее значение.

Чтобы установить или убрать точку останова, надо щелкнуть левой кнопкой мыши по серому полю рядом с нужной строкой программы, или, установив курсор в нужную строку, нажать клавишу **F9**. Выбранная строка станет красной, а в левой колонке появится красная точка – знак точки останова. Точки останова можно размещать только в тех строках, где есть операторы.

После останова можно продолжить выполнение программы. Если выбрать из пункта меню **Debug** команду **Continue** или нажать клавишу **F5**, то выполнение программы будет продолжено до следующей точки останова или до конца программы, если точки останова больше не встречаются.

Для пошагового выполнения программы (трассировки) надо из пункта меню **Debug** выбрать команду **Step Into** или нажать клавишу **F11.** Visual Basic 2005 выполнит текущую строку и остановится. Активная строка выделяется желтым цветом. Рядом с ней находится желтая стрелка. В этом режиме трассируются все подпрограммы. Если пошаговое выполнение какой-либо подпрограммы нужно пропустить, то следует из пункта меню **Debug** выбрать команду **Step Over** или нажать сочетание клавиш **F10**. В этом случае подпрограмма будет выполнена без пошаговой демонстрации. В любой момент времени трассировку можно прервать, выбрав команды **Continue** или **Stop Debugging** из меню **Debug**. Команда **Continue** продолжает выполнение программы. Команда **Stop Debugging** завершает работу программы.

Оглавление

Значение любой переменной можно узнать, наведя на нее курсор мыши. В тех случаях, когда одновременно надо контролировать значения нескольких переменных используются окна наблюдения **Watch**. Среда Visual Basic 2005 позволяет одновременно использовать четыре разных окна наблюдения для слежения за разными группами переменных. Для того чтобы вызвать эти окно надо из пункта меню **Debug** выбрать подменю **Windows**, а в нем – подменю **Watch** (обратите внимание, что этот пункт меню доступен только в процессе выполнения программы), а там нужное окно от **Watch 1** до **Watch 4**. В нижней части главного окна появится окно наблюдения (рис. 14). Оно состоит из трех колонок. В первой (**Name**) отображаются имена переменных, во второй (**Value**) – их значения, а в третьей (**Type**) – тип данных, к которому принадлежит эта переменная. Для того чтобы добавить переменную, достаточно просто набрать ее имя и нажать клавишу **Enter**. Если какую-то переменную надо удалить из списка наблюдения, то надо выделить в таблице соответствующую строку и нажать клавишу **Delete**. Когда в процессе выполнения программы некоторая переменная из списка меняет свое значение, оно выводится красным цветом. На следующем шаге оно снова восстанавливает обычный черный цвет. Таким образом можно легко следить за изменением значений переменных, что позволяет существенно повысить эффективность отладки программы.

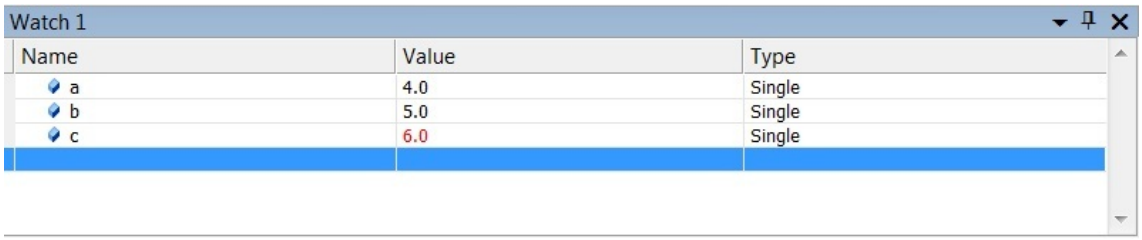


Рис. 14. Окно наблюдения за значениями переменных

3. Разработка интерфейса в среде VB. Основные элементы управления

Создание приложения начинается с разработки интерфейса. Элементы интерфейса (или элементы управления) размещаются на форме. Для этого используются панель элементов управления, конструктор форм и специальные инструменты среды Visual Basic 2005 для выравнивания размеров и положения элементов управления. Чтобы поместить любой элемент управления на форму, надо щелкнуть по кнопке с нужной пиктограммой на панели элементов управления, переместить курсор мыши в окно конструктора формы и, удерживая левую кнопку

Оглавление

мыши, растянуть прямоугольник, в которой будет вписан выбранный элемент управления.

3.1. Метка

Элемент управления Label (метка) предназначен для вывода текста, который не будет меняться в процесс работы приложения. Как правило, метки используются для вывода поясняющего текста или для подписи других элементов управления. На панели элементов управления метка обозначена пиктограммой с буквой «A» (). Рассмотрим основные свойства этого объекта.

- (Name) – имя метки. Так как обращение к метке происходит достаточно редко, то имя метки, как правило, не меняют. Если же метке необходимо задать имя, то его строят в соответствии с правилом имен и начинают с приставки *lbl*, например *lblResult*.
- AutoSize – это свойство позволяет выровнять размер метки по размеру написанного в ней текста. Если свойство AutoSize имеет значение True, то размер метки выравнивается автоматически. Если же свойство имеет значение False, то выравнивание делается вручную.
- BackColor – задает цвет фона. Задание значения происходит с помощью стандартного окна выбора цвета.
- BorderStyle – определяет тип рамки, которой обведен элемент управления. Возможны три варианта: рамка отсутствует (None), одинарная рамка (FixedSingle), трехмерная рамка (Fixed3D).
- Font – позволяет настроить шрифт, которым выводится текст метки. Для задания шрифта используется стандартное диалоговое окно выбора шрифтового оформления.
- ForeColor – задает цвет шрифта. Задание значения происходит с помощью стандартного окна выбора цвета.
- Text – основное свойство метки. Оно определяет текст, который будет отображаться на форме.

3.2. Текстовое поле

Элемент управления TextBox (текстовое поле) предназначен для ввода, редактирования и вывода информации. Оно используется для задания исходных данных и вывода полученных результатов. В текстовом поле может находиться как числовая, так и символьная информация. На панели элементов управления текстовое

поле обозначено пиктограммой с буквами «ab» (). Рассмотрим основные свойства этого объекта.

- (Name) – имя текстового поля. Имя составляют в соответствии с правилом имен и начинают с приставки *txt*, например *txtResult*.
- BackColor – задает цвет фона. Задание значения происходит с помощью стандартного окна выбора цвета.
- BorderStyle – определяет тип рамки, которой обведен элемент управления. Возможны три варианта: рамка отсутствует (None), одинарная рамка (FixedSingle), трехмерная рамка (Fixed3D).
- Font – позволяет настроить шрифт, которым выводится информация. Для задания шрифта используется стандартное диалоговое окно выбора шрифтового оформления.
- ForeColor – задает цвет шрифта. Задание значения происходит с помощью стандартного окна выбора цвета.
- Text – основное свойство текстового поля. Оно определяет текст, который ввел пользователь, или текст, который будет отображаться на форме.
- TextAlign – позволяет задать выравнивание текста в текстовом поле. Возможные варианты выравнивания: по левому краю (Left), по правому краю (Right), по центру (Center).
- MultiLine – определяет формат вывода текста в текстовое поле. Если свойство имеет значение True, то текст выводится в несколько строк (многострочный режим). Если свойство имеет значение False, то текст выводится в одну строку (однострочный режим).
- Lines – это свойство используется только в многострочном режиме. Оно представляет собой массив строк. Каждая строка хранится отдельно. Нумерация строк начинается с нуля.

При работе с текстовым полем важно помнить, что числа, введенные или выведенные в это поле, рассматриваются Visual Basic 2005 как набор символов. Поэтому при организации ввода/вывода числовой информации необходимо использовать соответствующие преобразования.

- Ввод числа в переменную *a*.

```
a = Val(txtA.Text)
```

Оглавление

Функция Val преобразует символьную информацию в числовую. Получив из текстового поля с именем txtA набор цифр, Val преобразует их в число, которое затем будет записано в переменную a.

- Вывод числа в текстовое поле.

```
txtA.Text = Str(12.34)
```

При выводе числа в текстовое поле необходимо провести обратное преобразование, то есть преобразовать информацию из числовой в символьную. Для этого используется функция `Str`. Она преобразует число, указанное в скобках, в набор символов – цифр, которые будут затем выведены в текстовое поле.

- Вывод числа из переменной *a* в текстовое поле.

```
txtA.text = Str(a)
```

Вывод числовой переменной ничем не отличается от вывода числа. Для него тоже необходимо использовать преобразование `Str`. Числовая переменная, значение которой надо вывести в текстовое поле, указывается в круглых скобках.

- Ввод строки в переменную *s*.

```
s = txtS.Text
```

При вводе символьной информации никаких дополнительных преобразований не требуется. Она просто переносится из значения свойства Text в переменную, имя которой указывается слева от оператора присваивания (знака равенства).

- Вывод текста в текстовое поле.

```
txtS.Text = "Выводимый текст"
```

При выводе текста тоже не требуется дополнительных преобразований. Достаточно указать в двойных кавычках текст, который должен отобразиться в текстовом поле.

- Вывод строки из переменной *s* в текстовое поле.

```
txtS.Text = s
```

Так же выполняется вывод значения текстовой переменной. Слева от оператора присваивания (знака равенства) указывается имя текстового поля и через точку свойство Text, а справа от оператора присваивания ставится имя переменной значение, которой надо распечатать в текстовом поле.

3.3. Кнопка

Элемент управления Button (кнопка) предназначен для запуска вычислительного процесса. На панели элементов управления кнопка обозначена пиктограммой с буквами «ab» (). Рассмотрим основные свойства этого объекта.

Оглавление

- (Name) – имя кнопки. Имя составляют в соответствии с правилом имен и начинают с приставки *bt*, например *btStart*.
- BackColor – задает цвет фона. Задание значения происходит с помощью стандартного окна выбора цвета.
- Font – позволяет настроить шрифт, которым выводится надпись на кнопке. Для задания шрифта используется стандартное диалоговое окно выбора шрифтового оформления.
- ForeColor – задает цвет шрифта. Задание значения происходит с помощью стандартного окна выбора цвета.
- Text – определяет надпись на кнопке.
- TextAlign – позволяет задать выравнивание текста на кнопке. Возможны девять различных вариантов выравнивания: текст располагается в левом верхнем углу, вверху по центру, в правом верхнем углу, в середине кнопки слева, в центре кнопки, в середине кнопки справа, в левом нижнем углу, внизу по центру, в правом нижнем углу.

Для того чтобы связать программный код с кнопкой, достаточно дважды щелкнуть левой кнопкой мыши по кнопке на разрабатываемой форме. После этого откроется окно редактора программного кода, в котором автоматически будет создана заготовка процедура обработки события нажатия на кнопку (click). Курсор находится между словам Private Sub и End Sub (рис. 15). Именно там и надо написать программный код, связанный с данной кнопкой.

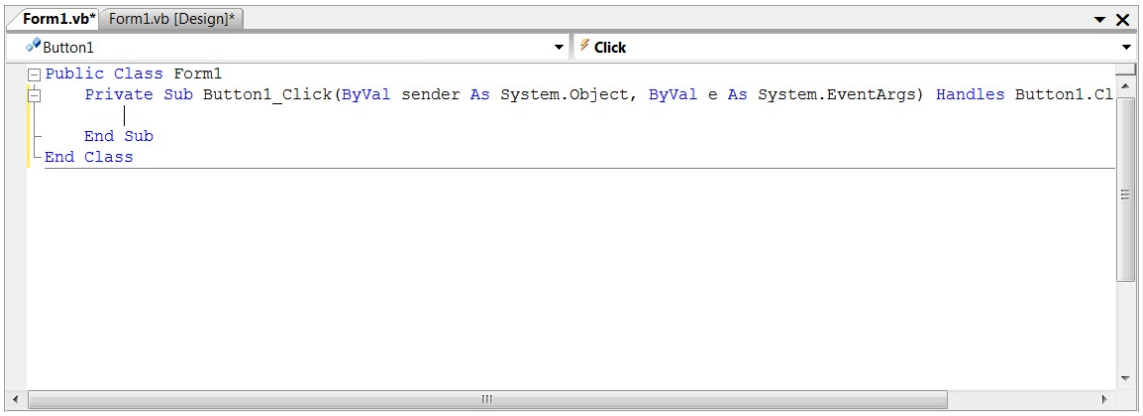


Рис. 15. Окно редактора программного кода с заготовкой процедуры обработки события

Оглавление

3.4. Окно списка

Элемент управления ListBox (окно списка) предназначен для вывода больших объемов информации, например, для распечатки массива чисел. Этот элемент управления используется только для вывода. Каждый выводимый элемент будет расположен на отдельной строке списка. На панели элементов управления окно списка обозначено следующей пиктограммой (). Рассмотрим основные свойства и методы этого объекта.

- (Name) – имя окна списка. Имя составляют в соответствии с правилом имен и начинают с приставки *lst*, например *lstA*.
- BackColor – задает цвет фона. Задание значения происходит с помощью стандартного окна выбора цвета.
- BorderStyle – определяет тип рамки, которой обведен элемент управления. Возможны три варианта: рамка отсутствует (None), одинарная рамка (FixedSingle), трехмерная рамка (Fixed3D).
- Font – позволяет настроить шрифт, которым выводится информация. Для задания шрифта используется стандартное диалоговое окно выбора шрифтового оформления.
- ForeColor – задает цвет шрифта. Задание значения происходит с помощью стандартного окна выбора цвета.
- Sorted – это свойство включает или выключает режим автоматической сортировки строк в окне списка. Строки располагаются в алфавитном порядке с учетом регистра (сначала располагаются строки, начинающиеся с прописных букв, затем – строки, начинающиеся со строчных букв). Если свойство имеет значение True, то режим сортировки включен. Если свойство имеет значение False, то режим сортировки выключен.
- Items.Count – это свойство показывает, сколько строк (элементов) содержится в окне списка. Это значение нельзя явно изменять в программе, его можно только считывать.
- Items.Item(Номер) – свойство позволяет получить доступ к любому элементу, хранящемуся в окне списка. Для этого необходимо указать номер нужной строки. Строки в окне списка нумеруются с нуля.
- SelectedIndex – номер выделенного элемента. Если в окне списка нет выделенного элемента (строки), то свойство имеет значение -1 (минус один).

Оглавление

- Items.Clear() – метод для работы с окном списка. Очищает окно списка, удаляя из него все записи. Пример: `lstA.Items.Cear()`.
- Items.Add(Элемент списка) – метод для работы с окном списка. Добавляет в окно списка новую строку (элемент списка). Новая строка всегда добавляется в конец списка. Текст, который будет отображаться в этой строке, указывается в качестве параметра метода. Выводить в окно списка можно только текстовую информацию. Для вывода числовой информации необходимо использовать функцию преобразования `Str`. Примеры.
  - вывод текстовой информации
    ```
    lstA.Items.Add("Выводимый текст")
    ```
  - вывод текстовой переменной *s*
    ```
    lstA.Items.Add(s)
    ```
  - вывод числа
    ```
    lstA.Items.Add(Str(12.34))
    ```
  - вывод числовой переменной *a*
    ```
    lstA.Items.Add(Str(a))
    ```
- Items.Insert(Номер позиции, Элемент списка) – метод для работы с окном списка. Вставляет новый элемент в список на позицию с указанным номером. Нумерация строк в окне списка идет с нуля. При добавлении числовой информации ее необходимо преобразовывать с помощью функции `Str`.
- Items.RemoveAt(Номер позиции) – метод для работы с окном списка. Удаляет из списка элемент с указанным номером. Нумерация элементов идет с нуля.

3.5. Выравнивание положения элементов управления

Для выравнивания положения и размеров элементов управления на форме используются команды, входящие в состав меню **Format**. Этот пункт меню доступен только при работе в конструкторе форм. Большинство команд из этого меню предназначены для работы с группой элементов управления. Для того чтобы выделить несколько элементов управления, необходимо последовательно щелкать левой кнопкой мыши на нужных элементах, удерживая нажатой клавишу **Ctrl** на клавиатуре. Каждый выбранный элемент выделяется специальной рамкой с темными уголками. Один из этих элементов будет использован в качестве эталонного. Его рамка выделена светлыми уголками (рис. 16). Чтобы выбрать эталонный объект, надо щелкнуть по нему левой кнопкой мыши,

Оглавление

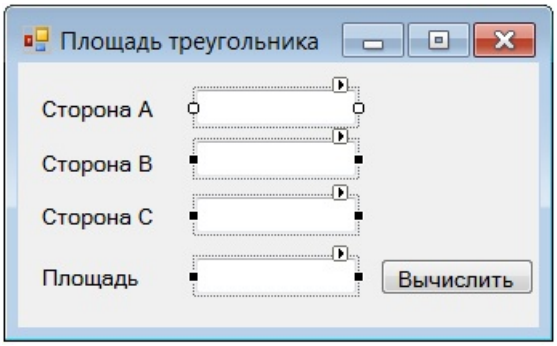


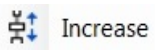
Рис. 16. Выделение группы элементов

Рассмотрим команды меню **Format**.

- Подменю **Align** объединяет команды выравнивания положения элементов без изменения их геометрических размеров. Возможные варианты выравнивания:
  - по левому краю; Lefts
  - выравнивание вертикальных центров; Centers
  - по правому краю; Rights
  - по верхнему краю; Tops
  - выравнивание горизонтальных центров; Middles
  - по нижнему краю; Bottoms
  - выравнивание по линиям сетки. to Grid
- Подменю **Make Same Size** содержит команды выравнивания геометрических размеров без изменения положения элементов. Возможные варианты:
  - выравнивание горизонтальных размеров (ширины); Width
  - выравнивание вертикальных размеров (высоты); Height
  - одновременное выравнивание обоих размеров; Both
  - выравнивание размеров по линиям сетки. Size to Grid
- Подменю **Horizontal Spacing** включает в себя команды настройки горизонтальных расстояний между элементами управления. Возможные действия:
  - увеличить расстояния; Increase
  - уменьшить расстояния; Decrease
  - сделать все расстояния одинаковыми; Make Equal

Оглавление

  - удалить все промежутки между элементами управления. 

- Подменю **Vertical Spacing** включает в себя команды настройки вертикальных расстояний между элементами управления. Возможные действия:
  - увеличить расстояния; 

  - уменьшить расстояния; 

  - сделать все расстояния одинаковыми; 

  - удалить все промежутки между элементами управления. 

- Подменю **Center in Form** объединяет команды, позволяющие отцентрировать отдельный элемент управления или их группу на форме. Команды этого подменю можно применять не только к группам элементов управления, но и к отдельным объектам. Возможные варианты центрирования:
  - по горизонтали; 

  - по вертикали. 

Список литературы

1. Волчёнков Н.Г. Программирование на Visual Basic 6: В 3-х ч. Часть 1. – М.: ИНФРА-М, 2000. – 286 с.
2. Шевякова Д.А., Степанов А.М., Карпов Р.Г. Самоучитель Visual Basic 2005 / под общ. ред. А.Ф. Тихонова. – СПб: БХВ-Петербург, 2007. – 576 с.
3. Богданов М.Р. Visual Basic 2005 на примерах. – СПб: БХВ-Петербург, 2007. – 592 с.

Оглавление