**Редактор меню Menu Editor**

Редактор меню Menu Editor Для проектирования меню всех видов используется редактор меню Menu Editor (Редактор меню) среды проектирования IDE (рис. 7.8). Редактор меню вызывается одним из следующих способов: командой Menu Editor (Редактор меню) меню Tools (Инструменты); нажатием кнопки Menu Editor на стандартной панели инструментов; нажатием комбинации клавиш +. Редактор создает меню для активного в данный момент окна, то есть, если активно MDI-OKHO, проектируется меню для него, если активна дочерняя форма, проектируется меню для дочерней формы. На рис. 7.8 показан редактор меню родительского окна, созданного автоматически мастером приложения Application Wizard. Редактор меню состоит из двух групп: элементов управления свойствами и элементов конструирования структуры меню. Управлять основными свойствами меню, о которых было сказано выше, можно с помощью следующих элементов редактора меню: поле Caption (Заголовок) — наименование пункта меню, то есть текст, появляющийся в меню; поле Name (Имя) — имя меню. Используется для идентификации объекта при написании программных кодов; раскрывающийся список Shortcut (Оперативная клавиша) — назначает комбинацию клавиш для быстрого вызова команды меню; поле HelpContextID (Идентификатор справки) — ссылка на тему в справочной системе; флажок Enabled (Доступно) — доступ к пункту меню; флажок Visible (Видимость) — определяет, будет ли виден на экране элемент меню; флажок WindowList (Список окон) — определяет наличие списка открытых окон. Рис. 7.8. Окно редактора меню Menu Editor Элементы группы конструирования структуры меню позволяют добавлять и удалять новые пункты меню, перемещать пункты по вертикали, меняя порядок их следования, и по горизонтали, меняя расположение пунктов в иерархии системы меню: кнопки с направленными вправо и влево стрелками перемещают пункты или команды меню в иерархии меню; кнопки с направленными вверх и вниз стрелками перемещают пункты или команды меню по структуре меню; Next (Следующий) — перемещает указатель к следующему пункту меню. Если указатель находится на последнем пункте меню, то создается новый пункт меню или новая команда меню такого же уровня иерархии; Insert (Вставить) — добавляет пункт меню или команду в пункт меню; Delete (Удалить) — удаляет пункт меню или команду из пункта меню. Для добавления пункта меню выполните следующие шаги: 1. Выберите место в имеющейся структуре меню. 2. Добавьте пункт меню, нажав кнопку Insert (Вставить). 3. В поле Caption (Заголовок) введите название пункта меню, которое будет отображаться в строке меню при запуске приложения на выполнение. 4. В поле Name (Имя) введите имя пункта меню, по которому он иденти фицируется в программном коде.

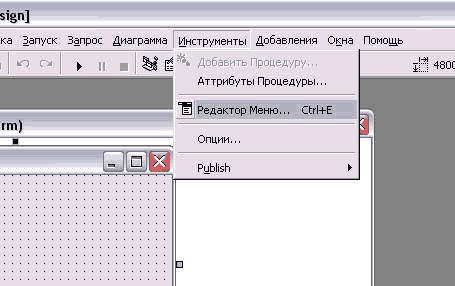
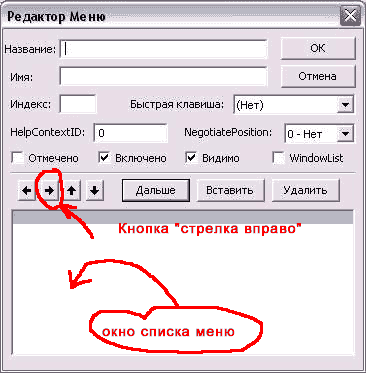
**Контекстное меню**

Контекстное меню Для приложения любого типа можно использовать удобное средство быстрого доступа к функциям — контекстное меню. Контекстное меню связано с некоторым действием (обычно это щелчок правой кнопки мыши на объекте) и вызывается в любом месте приложения. В исходном состоянии контекстное меню невидимо и визуализируется рядом с указателем мыши после вызова. Контекстным такое меню называется потому, что оно появляется рядом с выбранным объектом, и его состав зависит от содержания (контекста) этого объекта. После выбора команды из контекстного меню оно исчезает. Проектируется контекстное меню как один из пунктов верхнего уровня строки меню. Поскольку меню должно быть скрыто и вызываться определенным действием, свойство visible проектируемого пункта меню необходимо установить в значение False. Для визуализации контекстного меню на экране необходимо использовать метод popUpMenu формы. Например, для вызова меню File (Файл) при нажатии правой кнопки мыши достаточно написать такую условную конструкцию для события MouseUp формы: If Button = vbRightButton Then Form1.PopUpMenu mnuFile End If В этом примере при выполнении в форме события MouseUp (Нажатие на кнопку мыши) проверяется, какая кнопка нажата. В данном случае, при помощи внутренней константы Visual Basic 6 vbRightButton (Правая кнопка) проверяется нажатие на правую кнопку мыши и запускается метод PopUpMenu. В качестве параметра задается имя требуемого контекстного меню.

В дополнение к строке состояния и контекстному меню, в настоящее время большой популярностью пользуются панели инструментов, позволяющие ускорить доступ к функциям приложения. Обычно панель инструментов содержит наиболее часто используемые команды строки меню или контекстных меню. При разработке приложении можно применять панели двух видов: обычную панель ToolBar (рис. 7.9) и улучшенную панель CoolBar (рис. 7.10). Рис. 7.9. Обычная панель инструментов Рис. 7.10. Улучшенная панель инструментов Для проектирования панелей инструментов в Visual Basic 6 используются: мастер панелей инструментов, входящий в состав мастера приложений; элемент управления ToolBar для создания обычной панели инструментов; элемент управления CoolBar для создания улучшенной панели инструментов.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Часть 1. Основы Visual Basiс | | | |  |  |  |
|  | | | |  | 11.12.2004 |  |
|  | **Глава 9.** |  |
|  | **Создание меню с помощью Редактора Меню. Контекстное (всплывающее) меню. Метод PopupMenu.** |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Конечно, можно обойтись и без меню. Я, например считаю, что меню надо использовать в сложных программах, чтобы избежать использования множества командных кнопок. Однако меню придает программе профессиональный, "не детский" вид, в отличие от насобаченых на форму кнопок. Для добавления меню имеется редактор Menu Editor. С его помощью и можно как раз добавлять, изменять, уничтожать меню.

Создай новый exe-проект. Вверху окна VB найди кнопку инструменты, и из выпадающего меню выберем Редактор Меню (рис.18). Обратит внимание, что вызов редактора меню доступен, когда ты нахожишься в окне формы - объект. Если ты редактируешь код программы или программа запущена, вызвать редактор меню нельзя.  
  
**Рисунок 18.**У насв общем-то откроется окно меню редактора (Рис.19)  
   
**Рисунок 19.**

В первой строке этого чуда "Название" (Caption) мы пишем название меню, то, что у на отобразиться на экране при запуске программы. А во второй строке, "Имя" (Name) имя меню, которое используется в программном коде. Заполнение обоих этих полей обязательно.   
Для примера слепим меню, которое (должно же меню что-то делать), изменяет свойства графического объекта**Shape**, поскольку мы наплевали на него в прошлой главе. Поэтому, нажмем кнопку OK, отчего наше окно редактора меню закроется, положим на форму **Form1** объект **Shape1.**По умолчанию он у нас будет в виде прямоугольника. Снова откроем редактор точно также, как и в первый раз.

Итак, в названии пишем "Фигура", а в имени - **mnuShape**. Почему mnu? Потому что так принято, и в коде программы потом будет легче отличать меню от прочих переменных.

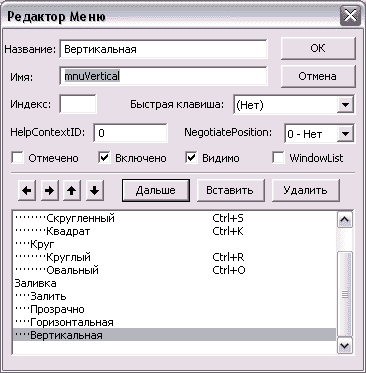
Из нашего пункта меню **Фигура** должно выскакивать, например, подменю **Прямоугольник** и подменю **Круг**. Мы можем нажать кнопку **Дальше** и курсор в нижнем большом окне (окно списка меню) перескочит на следующую строчку, сейчас пустую, а можем просто мышь кликнуть по нужной строке и таким образом перейти к следующему пункту. Перейдем на вторую строку и напишем в названии **Прямоугольник**, а в имени -**mnuPram**. Перейдем на третью строчку и напишем соответственно **Круг** и **mnuKrug**. Поскольку названия и имена я придумывал на ходу, прошу простить за их несуразность.  
Вот и здорово, но если мы запустим нашу программу, то увидим, что наши названия расположились рядком, и никаких подменю не получилось. Для того чтобы строчку **Прямоугольник** переместить в подменю, нужно щелкнуть по ней в окне списка меню (при этом она выделится) и нажать маленькую стрелочку вправо. Ты увидишь, что строка переместилась вправо, как бы стала на другой уровень. Нажимая кнопки вправо и влево мы создаем подменю второго, третьего и т.д. уровня или выводим из подменю на более высокий уровень. Теперь посмотри на эту таблицу.

|  |  |  |
| --- | --- | --- |
| Определение уровня подменю | Вид в редакторе меню | Вид в программе |
| Мы не нажимали стрелку вправо, все наши пункты меню расположились в главном меню, подменю нет. | http://vbzero.narod.ru/chapter1/graphica/page_9/menu10.gif | http://vbzero.narod.ru/chapter1/graphica/page_9/menu5.gif |
| Мы нажали на стрелку вправо на строках "Прямоугольник" и "Круг" **один раз,** отчего они сдвинулись на 4 пикселя вправо и вошли в меню "Фигура", как строки подменю. | http://vbzero.narod.ru/chapter1/graphica/page_9/menu9.gif | http://vbzero.narod.ru/chapter1/graphica/page_9/menu6.gif |
| Мы нажали на стрелку вправо на строках "Простой", "Скругленный", "Квадрат", а также "Круглый" и "Овальный" **два раза,**отчего они сдвинулись в право на 8 пикселей и вошли в подменю следующего уровня (в подменю "Прямоугольник" и подменю "Круг" соответственно). | http://vbzero.narod.ru/chapter1/graphica/page_9/menu8.gif | http://vbzero.narod.ru/chapter1/graphica/page_9/menu7.gif |

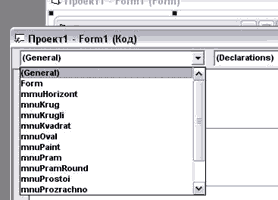
Сдвинем строчку **Круг**, как в таблице. Вот теперь у нас настоящее подменю. А кнопочками вверх и вниз мы можем изменять их положение по вертикали.

Теперь самостоятельно, так же, как в таблице, введем в подменю **Прямоугольник** подменю следующего уровня, куда войдет  
Название **Простой** Имя **mnuProstoi**  
Название **Скругленный** Имя **mnuPramRound**  
Название **Квадрат** Имя **mnuKvadrat**  
а в подменю**Круг** подменю следующего содержания (уровень тот же, что и у подменю Прямоугольник)   
Название **Круглый** Имя **mnuKrugli**  
Название **Овальный** Имя **mnuOval**  
  
Кроме того еще введем паралельно меню Фигура меню **Заливка** (Самого первого уровня) c именем **mnuPaint**и в него поместим следующие подменю (их нужно передвинуть с помощью стрелки вправо на следующий уровень)  
Название **Залить** Имя **mnuZalit**  
Название **Прозрачно** Имя **mnuProzrachno**  
Название **Горизонтальная** Имя **mmuHorizont**  
Название **Вертикальная** Имя **mnuVertical**

В поле "Быстрая клавиша" из выпадающего списка мы можем назначить клавишу или комбинацию клавиш, для каждого пункта нашего меню.

Общий вид редактора меню должен получиться, как на рисунке 20.  
  
**Рисунок 20.**  
  
Если горячие клавиши были тобой назначены, они тоже показаны в окне списка меню (Ctrl+S, Ctrl+K и т.д.)

Что еще есть в редакторе меню. Ну понятно, что с помощью кнопок "Дальше", "Вставить", "Удалить" мы можем передвигаться в окне списка меню, вставлять и удалять строки. Если поставить галочку возле "Отмечено", то в меню данная строка тоже будет помечена галочкой (Это для переключателей) только эта галка так и будет светиться для нас это пока совершенно бесполезно. Галочка возле "Видимо" указывает на то, что эта стока видима, если ее убрать, пользователь не будет видеть эту строчку меню. Галочка возле "Включено" показывает, что эта строчка доступна пользователю. Если ее убрать, то эта строка будет показана тускло и кликать по ней пользователю будет бесполезно. Вообще-то эти свойства для пользы дела надо задавать программно, а не тыкать сейчас по ним мышью.

Дальше надо заставить наше меню работать. Нажми кнопку OK и выйди из редактора меню.  
**Внимание. Если кнопка OK не будет нажата, твои изменения в меню не сохранятся!**  
Закрой это чертов редактор и перейди к окну редактирования кода. Кликни по полю (General) и опаньки, все наши строчки там есть! Можно кликать и делать процедуру для любой строчки, то бишь пункта меню (рисунок 21).  
  
**Рисунок 21.**  
Но нам нужны только пунты самого последнего уровня, потому, что предыдущих уровней мы используем только для выбора.  
Пишем процедуры, которые включим изменения свойств объекта **Shape** ( ну там заливку, и вид). Свойств объекта **Shape** на самом деле больше, но мы используем некоторые из них:

**Private Sub mmuHorizont\_Click()  
Shape1.FillStyle = 2  
End Sub**

**Private Sub mnuKrugli\_Click()  
Shape1.Shape = 3  
End Sub**

**Private Sub mnuKvadrat\_Click()  
Shape1.Shape = 1  
End Sub**

**Private Sub mnuOval\_Click()  
Shape1.Shape = 2  
End Sub**

**Private Sub mnuPramRound\_Click()  
Shape1.Shape = 4  
End Sub**

**Private Sub mnuProstoi\_Click()  
Shape1.Shape = 0  
End Sub**

**Private Sub mnuProzrachno\_Click()  
Shape1.FillStyle = 1  
End Sub**

**Private Sub mnuVertical\_Click()  
Shape1.FillStyle = 3  
End Sub**

**Private Sub mnuZalit\_Click()  
Shape1.FillStyle = 0  
End Sub**

Можно запустить программу, и с помощью меню изменять вид объекта **Shape**.

Но мы еще с меню не закончили. Допусти нам захотелось, чтобы меню появлялось не в стандарном месте в верхней строке нашего окна, а допустим, где угодно, например около курсоры мыши. Давайте "Фигура" оставим как есть, а меню "Заливка" превратим в контекстное, т. е. всплывающее. Но для этого сначала познакомимся с методом **PopupMenu**.

|  |
| --- |
| **Метод PopupMenu.** |

С помощью этого метода мы можем создавать всплывающие меню в любом месте нашей формы (одновременно отображаться может только одно).   
Синтаксис его таков

***объект.PopupMenu имяменю, флаг, позицияX, позицияY, выделеннаястрока***

***Объект*** - необязательный параметр, если опущен, то принимается форма, на которой фокус  
***Имяменю***- имя всплявающего меню. Обязательный параметр.  
***Флаг*** - константа или значение, указывающее расположение и поведение меню. Список констант приведен в таблице ниже. При использовании двух констант (по одной из каждой группы) необходимо между ними ставить логический оператор **Or**.  
***ПозицияX, позицияY***- координаты X и Y, указывающие, где отобразиться окно. Если они опущены, то принимаются координаты мыши.  
***Выделеннаястрока*** - название элемента всплывающего меню, которое отобразится жирным шрифтом.

*Таблица флагов, указывающих позицию и поведение контекстного меню.*

|  |  |  |
| --- | --- | --- |
| **Константа** | **Числовое значение константы** | **Описание** |
| **Размещение** | | |
| **vbPopupMenuLeftAlign** | **0** | По умолчанию. Размещение по x левого края всплывающего меню |
| **VbPopupMenuCenterAlign** | **4** | Центрирование всплывающего меню по x. |
| **VbPopupMenuRightAlign** | **8** | Размещение по x правого края всплывающего меню. |
| **Поведение** | | |
| **vbPopupMenuLeftButton** | **0** | По умолчанию. Элемент на всплывающем меню реагирует только на щелчок левой кнопки мыши. |
| **VbPopupMenuRightButton** | **8** | Элемент на всплывающем меню реагирует на щелчки и левой и правой кнопки мыши. |

|  |
| --- |
| **Контекстное меню.** |

Теперь, опираясь на метод**PopupMenu** сделать контекстное меню раз плюнуть. Создадим процедуру, которая срабатывает на щелчок мыши по форме **Form\_Click** и вставим в нее наш новый метод:

**Private Sub Form\_Click()  
Form1.PopupMenu mnuPaint, , , , mnuZalit  
End Sub**  
  
Как видишь, **mnuPaint** - это название нашего меню "Заливка", а **mnuZalit** - имя строки "Залить", оно будет выделяться жирным цветом. Остальные опции мы опускаем, хочешь, пробуй сам. Поскольку теперь меню "Заливка" в старом месте, в верхней строке формы нам не нужна, сделаем ее невидимой. Зайдем в Редактор Меню выделим в окне списка меню строку "Заливка" и поставим галочку у "Видимо". Ну, пока хватит. Если понадобиться исходник, то как всегда, его можно скачать вверху страницы.

**Главное меню редактора Visual Basic**

В таблице 9.1 приведено описание некоторых характерных пунктов главного меню редактора Visual Basic Большая часть описываемых пунктов меню предназначена для работы с окном модуля, и смысл их будет более понятен при изучении соответствующего материала

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица 9.1.** Некоторые характерные пункты главного меню приложения Visual Basic | | | |
| **Меню** | **Пункт** | **Кнопка панели инструментов** | **Описание** |
| **Файл (File)** | **Импортировать файл** (**Import File...**) | http://matveev.kiev.ua/macceass/ch1/gl9/images/t1.gif | Позволяет импортировать модули и формы из внешнего файла Visual Basic в формате *\*.frm* (файлы форм), *\*. has* (файлы Basic) и *\*.cls* (файлы классов). Клавиатурная комбинация — **<Ctrl>+<M>**. |
| **Экспортировать файл (Export File...)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t2.gif | Позволяет экспортировать существующий модуль или форму во внешний файл в формате *\*.cls* (файл класса). Клавиатурная комбинация —**<Ctrl>+<E>.** |
| **Удалить*ИмяМодуля* (Remove*ИмяМодуля)*** |  | Удаляет из проекта модуль активной формы или модуль с вкладки *Модули* (*Modules*). Выполненное действие отменить невозможно. |
| **Правка (Edit)** | **Отступ вправо (Indent)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t3.gif | Сдвигает все выбранные строки текста в следующую позицию табуляции. Эта операция позволяет улучшить читабельность программы. Клавиатурный эквивалент — **<Tab>**. |
| **Отступ влево (Outdent)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t4.gif | Сдвигает все выбранные строки текста в предыдущую позицию табуляции. Эта операция также позволяет улучшить читабельность программы. Клавиатурный эквивалент — **<Shift>+<Tab>**. |
| **Список свойств/методов (List Properties / Methods)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t5.gif | Открывает в окне модуля поле со списком, содержащим наименования всех свойств и методов объекта в позиции курсора. Если такого объекта нет, список содержит глобальные методы. Клавиатурная комбинация — **<Ctrl>+<J>**. |
| **Список констант (List Constants)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t6.gif | В окне модуля открывает поле со списком, в котором содержатся доступные константы. Клавиатурная комбинация — **<Ctrl>+<Shift>+<J>**. |
| **Подсказка (Quick Info)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t7.gif | Отображает информацию о синтаксисе переменных, функций, операторов, методов или процедур, выбранных в окне модуля. Клавиатурная комбинация — **<Ctrl>+<I>**. |
| **Информация о параметрах (Parameter Info)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t8.gif | Отображает всплывающее окошко с информацией о параметрах выбранной процедуры или оператора. Клавиатурная комбинация — **<Ctrl>+<Shift>+<I>**. Чтобы убрать окошко, нужно нажать клавишу **<Esc>**. |
| **Завершить слово (Complete Word)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t9.gif | Завершает слово (ключевое слово или название процедуры), которое вы начали набирать, если редактор Visual Basic в состоянии его распознать. Клавиатурная комбинация — **<Ctrl>+<Space>**. |
| **Закладки (Bookmarks)** |  | Подменю, содержащее четыре пункта, которые позволяют добавить/убрать закладку, очистить все закладки, перейти к следующей или предыдущей закладке. |
| **Вид (View)** | **Код (Code)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t10.gif | Отображает или активизирует окно модуля для выбранного объекта. Клавиатурный эквивалент — **<F7>**. |
| **Объект (Object)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t11.gif | Отображает активный объект (например, форму в режиме конструктора, для которой открыт модуль). Клавиатурная комбинация — **<Shift>+<F7>**. |
| **Определение (Definition)** |  | Переводит в окне модуля к определению переменной или процедуры, находящейся в позиции курсора. Клавиатурная комбинация — **<Shift>+<F2>**. |
| **Последняя позиция (Last Position)** |  | Возвращает к последней позиции курсора, например, если перед этим вы воспользовались пунктом *Определение* (*Definition*). Клавиатурная комбинация — **<Ctrl>+<Shift>+<F2>**. |
| **Просмотр объектов (Object Browser)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t12.gif | Отображает окно *Просмотр объектов* (*Object Browser Window*). Клавиатурный эквивалент — **<F2>**. |
| **Окно непосредственной отладки (Immediate Window)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t13.gif | Отображает окно *Непосредственной отладки* (*Immediate Window*). Клавиатурная комбинация — **<Ctrl>+<G>**. |
| **Окно локальных переменных (Locals Window)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t14.gif | Отображает окно *локальных переменных* (*Locals Window*). |
| **Окно контрольных выражений (Watch Window)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t15.gif | Отображает окно *контрольных выражений* (*Watch Window*). |
| **Окно стека вызовов (Call Stack)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t16.gif | Отображает окно *стека вызовов* (*Call Stack Windows*). Клавиатурная комбинация — **<Ctrl>+<L>**. |
| **Окно проекта (Project Explorer)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t17.gif | Отображает окно *проекта* (*Project Explorer*).  Клавиатурная комбинация — **<Ctrl>+<R>**. |
| **Окно свойств (Properties Window)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t18.gif | Отображает окно *свойств* (*Properties Window*). Клавиатурный эквивалент — **<F4>**. |
| **Вставка (Insert)** | **Процедура (Procedure)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t19.gif | Вставляет в модуль новую процедуру, *подпрограмму* (*Sub*), *функцию* (*Function*) или *свойство объекта* (*Property*). |
| **Модуль (Module)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t20.gif | Вставляет новый стандартный модуль |
| **Модуль класса (Class Module)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t21.gif | Вставляет новый модуль класса |
| **Файл... (File...)** |  | Позволяет вставить из внешнего файла модуль, форму или текст процедуры |
| **Отладка (Debug)** | **Компилировать*ИмяБазы* (Compile*ИмяБазы)*** |  | Компилирует базу данных |
| **Шаг с заходом в процедуру (Step Into)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t22.gif | Выполняет текущий оператор и останавливается на следующем. Если текущий оператор является вызовом процедуры, то выполнение программы прервется на первом операторе этой процедуры. Клавиатурный эквивалент — **<F8>**. |
| **Шаг без захода в процедуру (Step Over)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t23.gif | Выполняет те же действия, что и в предыдущем пункте, но без захода в вызываемую процедуру. Клавиатурный эквивалент — **<Shift>+<F8>** |
| **Шаг с возвратом в  вызвавшую процедуру (Step Out)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t24.gif | Выполняет все операторы, расположенные за текущим оператором в процедуре. Выполнение программы останавливается на операторе, следующим за вызовом текущей процедуры. Клавиатурная комбинация — **<Ctrl>+<Shift>+<F8>**. |
| **Выполнять до позиции курсора (Run То Cursor)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t25.gif | Выполняет программу вплоть до оператора в позиции курсора. Клавиатурная комбинация — **<Ctrl>+<F8>**. |
| **Добавить контрольное выражение (Add Watch)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t26.gif | Вызывает диалоговое окно для добавления нового контрольного выражения *Add Watch*. |
| **Изменить контрольное выражение (Edit Watch)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t27.gif | Вызывает диалоговое окно для редактирования добавленного ранее контрольного выражения *Edit Watch*. Клавиатурная комбинация — **<Ctrl>+<W>**. |
| **Быстрый контроль выражения (Quick Watch)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t28.gif | Отображает диалоговое окно *Quick Watch*, содержащее текущее значение выбранного выражения. Клавиатурная комбинация — **<Shift>+<F9>**. |
| **Установить/снять точку прерывания (Toggle Breakpoint)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t29.gif | Устанавливает/снимает точку прерывания в позиции курсора. Клавиатурная комбинация — **<F9>**. |
| **Снять все точки прерывания (Clear All Breakpoint)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t30.gif | Снимает все точки прерывания. Клавиатурная комбинация — **<Ctrl>+<Shift>+<F9>**. |
| **Установить следующий оператор (Set Next Statement)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t31.gif | Устанавливает точку выполнения на указанную строчку кода. Клавиатурная комбинация — **<Ctrl>+<F9>**. |
| **Показать следующий оператор (Show Next Statement)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t32.gif | Указывает оператор кода, который будет выполняться следующим. |
| **Запуск (Run)** | **Запуск процедуры/формы или Запуск макроса или Продолжить (Run Sub/UserForm или Run Macro или Continue)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t33.gif | Запускает текущую открытую процедуру, если такой нет — запускает активную форму. Если оба условия не выполняются, пункт меню выглядит как *Запуск макроса* (*Run Macro*). В режиме прерывания выполнения (*Break Mode*) программы этот пункт выглядит как *Продолжить* (*Continue*) и позволяет продолжить выполнение программы. Клавиатурный эквивалент — **<F5>**. |
| **Прервать (Break)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t34.gif | Останавливает выполнение программы и переводит ее в режим прерывания. Клавиатурная комбинация — **<Ctrl>+<Break>**. |
| **Сброс (Reset)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t35.gif | Прекращает выполнение программы, возвращая ее в первоначальное состояние. При этом очищается стек вызовов и переменные уровня модуля. |
| **Включить/Выключить режим конструктора (Design Mode/Exit Design Mode)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t36.gif | Включает/выключает режим конструктора. В режиме конструктора не выполняется код модуля и не происходят события. |
| **Сервис (Tools)** | **Ссылки (References)** | http://matveev.kiev.ua/macceass/ch1/gl9/images/t37.gif | Отображает диалоговое окно *Ссылки* (*References*). В этом окне можно подключить или отключить библиотеки объектов или типов, делая доступным код других приложений в вашей базе данных. Часть библиотек подключается автоматически при инсталляции Microsoft Office. |
| Параметры (Options) |  | Отображает диалоговое окно *Параметры* (*Options*), при помощи которого можно изменить настройки редактора Visual Basic. |
| Свойства (Properties) |  | Открывает, диалоговое окно *Свойства* (*Properties*), содержащее некоторые свойства проекта, относящиеся в основном к его модулям. |

**ЗАМЕЧАНИЕ**

Не все кнопки, приведенные в таблице 9.1, по умолчанию присутствуют на панели инструментов редактора Visual Basic. Однако их можно туда добавить при помощи диалогового окна Настройка (Customize). Это окно можно отобразить на экране, выбрав одноименный пункт из контекстного меню любой панели инструментов Visual Basic.

**9.1.2. Настройка редактора Visual Basic**

Настроить поведение редактора Visual Basic можно при помощи диалогового окна *Параметры* (*Options*) (см. рис. 9.2), доступного через одноименный пункт меню *Сервис* (*Tools*). Это окно содержит четыре вкладки — *Редактор* (*Editor*), *Формат* (*Editor Format*), *Общие* (*General*) и *Закрепление* (*Docking*). Ниже кратко описано назначение всех элементов управления, расположенных на этих вкладках.

|  |
| --- |
| http://matveev.kiev.ua/macceass/ch1/gl9/images/m2.gif |
| **Рисунок 9.2.** Диалоговое окно **Параметры (Options)** редактора Visual Basic. |

**Редактор (Editor)**

На этой вкладке собраны элементы управления, при помощи которых можно настроить поведение редактора в окне модуля (*Code Window*), а также некоторые свойства самого окна кода.

• Флажок *Автоматическая проверка синтаксиса (Auto Syntax Check).*Указывает, должен ли редактор Visual Basic проверять синтаксическую корректность введенной строки программы. По умолчанию флажок установлен.

* Флажок ***Обязательное объявление переменных*** (***Require Variable Declaration***). Определяет, обязательно ли описание всех используемых переменных или нет. По умолчанию флажок снят. Если флажок установлен, то в каждый новый модуль в раздел общих описаний будет добавляться строка *Option Explicit*, в результате чего попытка использования любой необъявленной переменной вызовет ошибку.
* Флажок ***Автоматическое отображение списка элементов*** (***Auto List Members***). Определяет, будет ли выводиться список с информацией, подсказывающей логическое завершение вводимого оператора. По умолчанию флажок установлен.
* Флажок ***Автоматическое отображение подсказки*** (***Auto Quick Info***). Определяет, будет ли отображаться подсказка о процедуре и ее параметрах. По умолчанию флажок установлен.
* Флажок ***Автоматическое отображение значения выражения*** (***Auto Data Tips***). Разрешает или запрещает отображение значения переменной (или выделенного выражения), над которой расположен курсор. Это может происходить только в режиме прерывания выполнения (*Break Mode*). По умолчанию флажок установлен.
* Флажок ***Автоотступ*** (***Auto Indent***). Разрешает или запрещает отступ на величину ширины табуляции для первой строки кода. По умолчанию флажок установлен.
* Поле ***Ширина табуляции*** (***Tab Width***). Задает ширину табуляции в символах. Возможные значения от 1 до 32 символов. Значение по умолчанию — 4 символа.
* Флажок ***Разрешить перетаскивание*** (***Drag-and-Drop Text Editing***). Разрешает или запрещает перетаскивание элементов программного кода в окна *непосредственной отладки* (*Immediate Window*) и *контрольных выражений* (***Watch Window***). По умолчанию флажок установлен.
* Флажок ***Режим полного модуля*** (***Default to Full Module View***). Указывает режим открытия по умолчанию для окна модуля — *режим процедуры* (*Procedure View*) или *полного модуля* (*Full Module View*). По умолчанию флажок установлен — окно модуля открывается в режиме *полного модуля* (*Full Module View*).
* Флажок ***Отображать разделитель*** (***Procedure Separator***). Задает отображение разделителя между процедурами модуля. В качестве разделителя процедур используется горизонтальная линия. По умолчанию флажок установлен.

**Формат (Editor Format)**

На этой вкладке собраны параметры, управляющие отображением текста в окне модуля.

* Список ***Цвета кода*** (***Code Colors***). В этом списке выбираются синтаксические конструкции кода, для которых задается оформление (например, *обычный текст* (*Normal Text*), *выделенный текст* (*Selected Text*), *ключевые слова* (*Keyword Text*) и т.д.).
* Поля со списком ***Текст*** (***Foreground***), ***Фон*** (***Background***) и ***Индикатор*** (***Indicator***). Эти элементы задают цвет текста, фона и панели индикатора для выбранной выше конструкции. Значения по умолчанию для большинства текстовых конструкций в окне модуля — *Auto*.
* Поле со списком ***Шрифт*** (***Font***). Указывает шрифт для всех текстовых элементов окна модуля (*Code Window*).
* Поле со списком ***Размер*** (***Size***). Указывает размер шрифта, используемого в окне модуля (*Code Window*).
* Флажок ***Панель индикатора*** (***Margin Indicator Bar***). Указывает, отображается ли *панель индикатора* (серая полоса в левой части окна модуля). Флажок по умолчанию установлен.

В правой нижней части окна расположен элемент *Образец* (*Sample*), в котором отображаются все изменения, сделанные на вкладке *Формат* (*Editor Format).*

**Общие (General)**

На этой вкладке расположены параметры, управляющие некоторыми аспектами обработки ошибок, компиляции и другие общие параметры.

* Группа параметров, управляющих отображением координатной сетки в форме (в режиме конструктора) — *Отображать сетку* (*Show Grid*), *Ширина* сетки (*Width*), *Высота* сетки*(Height*) и *Выровнять по узлам сетки* (*Align Controls to Grid*). Эти параметры используются при конструировании пользовательских форм, которые в Microsoft Access не используются.
* Флажок *Отображать подсказки* (*Show ToolTips*). Управляет отображением всплывающих подсказок для кнопок панели инструментов. Флажок по умолчанию установлен.
* Флажок *Скрывать окна при сворачивании проекта* (*Collapse Proj.Hide Windows*). Определяет, закрываются ли автоматически объекты, окна модуля и т.д. когда проект сворачивается в окне проектов*(Project Explorer Window*). Флажок по умолчанию установлен.
* Флажок *Уведомлять перед сбросом* (*Notify Before State Loss*). Определяет, будет ли отображаться предупреждающее сообщение о том, что требуемое действие приведет к сбросу всех переменных уровня модуля для текущего проекта. Флажок по умолчанию сброшен.
* Группа параметров *Обработка Ошибок* (*Error Trapping*). Определяет, как редактор Visual Basic обрабатывает ошибки. Возможные значения: *Останов при любой ошибке* (*Break on All Errors*) (любая ошибка приводит к остановке выполнения программы), *Останов в классе модуля* (*Break in Class Module*) (любая необработанная ошибка в модуле класса приводит к остановке выполнения программы на операторе, вызвавшем ошибку) и *Останов при необработанной ошибке* (*Break on Unhandled Errors*) (любая необработанная ошибка приводит к остановке программы). Значение по умолчанию — *Останов при необработанной ошибке* (*Break on Unhandled Errors*).
* Флажок *Компиляция по требованию* (*Compile on Demand*). Определяет, компилируется ли проект весь сразу, или модули компилируются по мере необходимости, что ускоряет запуск приложения. Флажок по умолчанию установлен, что приводит к полной компиляции проекта перед запуском.
* Флажок *Фоновая компиляция* (*Background Compile*). Доступен, если установлен предыдущий флажок. Определяет, будет ли проект компилироваться по возможности в фоновом режиме. По умолчанию флажок установлен.

**Закрепление (Docking)**

На этой вкладке расположено шесть флажков, по одному на каждое из вспомогательных окон (за исключением окна *стека вызовов* (*Call Stack Window*)). Установленный слева от названия окна флажок сигнализирует о том, что соответствующее вспомогательное окно будет *прикрепляемым* (*dockable*), иначе — вспомогательное окно будет обычным дочерним окном.

**9.1.3. Диалоговое окно Ссылки (References)**

При помощи диалогового окна *Ссылки* (*References*) можно подключить к разрабатываемому проекту код других объектов. Эти объекты располагаются во внешних файлах, которые могут быть файлами приложений, библиотек типов или объектов. Они могут быть разработаны как сторонними' производителями, так и поставляться с Microsoft Office. В частности, к проекту можно подключить другую базу данных Microsoft Access (файлы *\*.mdb* или *\*.mde*), проект Microsoft Access (файлы *\*.adp* или *\*.ade*), надстройки или инструментальные средства (файлы *\*.mda*). Окно*Ссылки* (*References*) доступно через пункт *Ссылки* (*References*) меню **Сервис** (**Tools**) (см. рис. 9.3). Добавить к проекту надстройку можно также при помощи мастера *Менеджер надстроек* (*Add-Ins Manager*), который доступен через одноименный пункт меню **Надстройки** (**Add-Ins**).

|  |
| --- |
| http://matveev.kiev.ua/macceass/ch1/gl9/images/m3.gif |
| **Рисунок 9.3.** В диалоговом окне **Ссылки** (**References)**подключенные библиотеки отмечены галочками. |

В центре диалогового окна *Ссылки* (*References*) находится список, содержащий названия всех доступных для подключения объектов. Слева от названия каждого объекта в этом списке расположен флажок, который служит для подключения или отключения соответствующего объекта. Подключенные библиотеки перечислены в списке по их приоритету. Наивысший приоритет имеет объект вверху списка. Изменить приоритет можно при помощи группы кнопок *Приоритет* (*Priority*), размещенных в окне *Ссылки* (*References*) справа от списка. Приоритет ссылок влияет на очередность загрузки библиотек кода и их компиляции. Неподключенные объекты располагаются ниже подключенных в алфавитном порядке. Ниже списка объектов в окне выводится название текущего выбранного объекта, путь к файлу, в котором находится код объекта, а также его языковая версия.

Некоторые ссылки должны быть обязательно включенными в любом случае (например, *Visual Basic for Application* — без этого объекта не будет работать Visual Basic). Другие ссылки должны быть включены для конкретной базы данных, в другом случае они могут быть выключены. Если вы создаете новую базу данных или проект Microsoft Access, то устанавливается ссылка на библиотеку *Microsoft ActiveX Data Objects 2.5 Library* (*ADO 2.5*); если же Вы открываете базу данных, сделанную в версиях Microsoft Access до 2000 года, то ссылка устанавливается на библиотеку *Microsoft DAO 3.6 Library* (*DAO 3.6*).

Кроме стандартных библиотек, установленных в системе при инсталляции Microsoft Office или Microsoft Windows, существует огромное количество библиотек объектов, разработанных сторонними производителями. Они могут попасть в систему либо как самостоятельные объекты, которые специально устанавливаются для использования их кода, либо как побочный эффект от инсталляции каких либо программных продуктов. В любом случае при использовании нестандартных объектов в проекте следует позаботиться о его переносимости. Программа, написанная с использованием нестандартных объектов, не будет работать в среде Microsoft Access, в которой отсутствует установленная ссылка на этот объект. Чаще всего отсутствие нужных ссылок проявляется в виде появления сообщений о неверном имени функции, которая совсем недавно еще работала безошибочно. В этом случае следует проверить — все ли необходимые для работы приложения ссылки установлены.

Чтобы подключить библиотеку объекта к базе данных Microsoft Access, откройте диалоговое окно *Ссылки* (*References*) и нажмите кнопку *Обзор* (*Browse*). Открывшееся диалоговое окно*Добавить ссылку* (*Add Reference*) является обычным стандартным окном для открытия файла. В этом окне можно выбрать файл одного из типов: *Базы данных Microsoft Access* (*\*.mdb*) (*MicrosoftAccess Databases* (*\*.mdb*)), *Надстройки* (*\*.mda*) (*Add-ins (\*.mda)*), *Файлы MDE* (*\*.mde*) (*MDE Files (\*.mde)*), *Проект Access* (*\*.adp*) (*Access Project (\*.adp)*), *Файлы ADE(\*.ade)* (*ADEFiles (\*.ade)*), *Библиотеки типов* (*\*.olb, \*.tlb, \*.dll*) (*Type Libraries (\*.olb; \*.tlb; .dll)*), *Исполняемые файлы* (*\*.ехе, \*.dtt*) (*Executable Files (\*.exe; \*.dlt)*) и *Элементы управления ActiveX (\*.ocx)* (*ActiveX Controls (\*.ocx)*). Выберите нужный файл и нажмите кнопку **ОК**. Выбранная ссылка появится в списке доступных объектов. При помощи группы кнопок *Приоритет* (*Priority*) задайте для ссылки подходящий приоритет и закройте диалоговое окно *Ссылки* (*References*). После этого код из подключенной библиотеки станет доступным в текущем проекте.

**9.2. Окно модуля (Code Window)**

Окно модуля предназначено для отображения содержащихся в нем программ на языке Visual Basic. В этом окне вы можете создавать новые программы и редактировать созданные ранее. Кроме того, окно модуля может использоваться для отладки (поиска и устранения ошибок). В редакторе Visual Basic может быть открыто одновременно неограниченное количество *окон модуля* (*Code Window*), каждое из которых содержит свой модуль — модуль формы, отчета или стандартный модуль. Это позволяет легко перемещаться между различными модулями (например, при помощи меню **Окно** (**Window**), в котором содержится список всех открытых в данный момент окон модулей), копировать участки программного кода из одного модуля в другой и т.д.

На рис. 9.4 изображено окно нового модуля и обозначены некоторые наиболее важные его элементы. В левом верхнем углу окна находится поле со списком доступных в модуле объектов (*Object Box*). В модулях всех типов присутствует один стандартный объект — *Общий* (*General*). Точнее его можно *назвать разделом общих описаний*. Этот раздел содержит описание общих переменных и процедур, доступных из других процедур модуля. В модулях форм и отчетов кроме объекта *Общий* (*General*) доступны объекты, представляющие собой форму (или отчет), ее разделы, а также все содержащиеся в ней элементы управления.

|  |
| --- |
| http://matveev.kiev.ua/macceass/ch1/gl9/images/m4.gif |
| **Рисунок 9.4.** Окно нового модуля в редакторе Visual Basic. |

В правом верхнем углу *окна модуля* (*Code Window*) расположено еще одно поле со списком — *список процедур/событий* (*Procedures/Event Box*). В этом списке содержатся процедуры и/или события, соответствующие объекту, выбранному в списке объектов (*Object Box*). Для объекта *Общий* (*General*) в списке процедур и событий присутствует обязательный элемент *Объявления*(*Declaration*). Этот элемент является разделом, в котором обычно располагаются объявления общих переменных, доступных для использования в любой точке модуля.

В левой части окна модуля расположена серая вертикальная полоска — *Панель индикатора* (*Margin Indicator Bar*). На панели индикатора могут располагаться различные значки, обозначающие состояние строк кода. Смысл этих значков приведен ниже:

http://matveev.kiev.ua/macceass/ch1/gl9/images/p4.gif — *Точка прерывания* (*Breakpoint*). Значок с изображением красной жирной точки означает, что в этой строке программы установлена точка прерывания. Точку прерывания (*Breakpoint*) можно установить или снять, щелкая левой клавишей мыши на *Панели индикатора* (*Margin Indicator Bar*). Фон строки, в которой установлена точка прерывания, окрашивается в красный цвет.

http://matveev.kiev.ua/macceass/ch1/gl9/images/p5.gif — *Закладка* (*Bookmark*). Этот значок указывает месторасположение закладок в тексте программы.

http://matveev.kiev.ua/macceass/ch1/gl9/images/p6.gif — *Маркер следующего выполняемого оператора* (*Call Stack Marker*). Значок указывает на следующий выполняемый оператор. Фон этого оператора окрашивается в желтый цвет. Это изображение на *Панели индикатора* (*Margin Indicator Bar*) может появиться только в режиме прерывания выполнения программы (*Break Mode*), например, в точке прерывания. *Маркер следующего выполняемого оператора* (*Call Stack Marker*) можно перетаскивать при помощи мыши вверх или вниз, тем самым перемещая точку останова программы. Обратите внимание, что маркер можно перетащить и назад, вернувшись к уже выполненным операторам. Иногда это бывает очень удобно при отладке. При перемещении таким образом маркера вдоль панели индикатора напротив участков кода, на которые нельзя переместить точку останова программы, указатель мыши приобретает вид дорожного знака *Остановка запрещена.*

В левой нижней части окна модуля размещены две кнопки — *Режим процедуры* (*Procedure View*) и *Режим полного модуля* (*Full Module View*). Первая из этих кнопок передает окно модуля в режим представления процедуры. В этом режиме в окне отобра-тся только одна процедура, выбранная в поле *Список процедур/событий* (*Procedures/Events Box*). Вторая кнопка переключает окно модуля в режим полного модуля. В этом име окно отображает текст всего модуля, и отсюда, воспользовавшись прокруткой ж или вниз, можно перейти к описанию любой процедуры или переменной модуля. Над вертикальной полосой прокрутки расположена *панель разделения окна* (*Split Bar*) рис. 9.4). Указатель мыши над этой панелью принимает вид двух горизонтальных чер-ж со стрелками вверх и вниз. Такая форма курсора называется *формой вертикалъного разделения* (*Vertical Split*). Перетаскивая панель при помощи мыши вниз вдоль вертильной полосы прокрутки, можно добиться разделения окна модуля на две независимые части (см. рис. 9.5). В каждой из частей могут отображаться различные участки програм-одного модуля. Например, в верхней части разделенного окна можно вывести раздел описаний переменных, а в нижней части — редактировать код одной из процедур модуля. В этом случае при необходимости можно легко добавить описание новой переменной вернуться к редактируемому участку программы, имея перед глазами ее название и тип.

|  |
| --- |
| http://matveev.kiev.ua/macceass/ch1/gl9/images/m5.gif |
| **Рисунок 9.5.**  Окно модуля, разделённое на две части при помощи **Панели разделения** (**Split Bar**). |

 Как ни странно, эта мелочь значительно облегчает жизнь программисту. Зкна модуля предоставляют достаточно мощные возможности, в том числе интеллек-ьные, для ввода, редактирования и отображения текста программ. По умолчанию при отображении текста программ используется цветовое выделение различных синтаксических конструкций. Например, символы ключевых слов Visual Basic отображаются синим цветом, текст комментария — зеленым и т.д. Как уже упоминалось чуть выше, эту цветовую схему можно изменить при помощи диалогового окна *Параметры*(*Options*) на вкладке *Формат* (*Editor Format*).

Существует несколько возможностей, облегчающих ввод текста программы. Все они сводятся к двум основным группам: одни завершают начатое вами слово, другие отображает всплывающую контекстную подсказку. К первой группе относятся: команды подстановки подходящего значения из *Списка свойств/методов* (*List Properties/Methods*), *Списка констант* (*List Constants*), а также команда *Закончить слово* (*Complete Word*). Ко второй те можно отнести команды *Подсказка* (*Quick Info*) и *Информация о параметрах* (*Parameter Info*).

Все эти команды доступны либо в контекстном меню окна модуля, либо в главном меню редактора Visual Basic. Однако главное удобство их применения — в автоматизации. То есть, подходящее по контексту значение и список допустимых вариантов предлагается редактором автоматически. Например, как только вы введете в текст программы имя специального объекта*DoCmd* и поставите точку, на экране отобразится список, содержащий все его методы. После этого переместиться в списке к нужному значению можно несколькими способами. Можно просто перемещаться по списку при помощи мыши и вертикальной полосы прокрутки либо использовать клавиши управления курсором (*стрелка вверх, стрелка вниз, PageUp* и *PageDown*). Другой способ — вводить с клавиатуры первые символы искомого значения. Редактор будет автоматически подставлять в текст значение из списка, первые символы которого совпадают с уже введенными. При этом можно не заботиться о регистре вводимых символов — впоследствии регистр символов будет скорректирован автоматически. Например, если для объекта *DoCmd* после точки ввести символы *Ru*, активным в списке методов станет значение *RunCommand*. Если добавить еще два символа — *п* и *s*, активным в списке станет значение *RunSQL*.

Когда необходимое значение найдено в списке, его можно вставить в текст программы. Сделать это можно двумя способами — при помощи двойного щелчка мышью или нажатием на одну из клавиш — **<Tab>**, **<Space>** или **<Enter>**. При нажатии на клавишу **<Tab>** выбранное значение вставляется в текст программы. При нажатии на клавишу **<Space>** в текст программы, кроме выбранного значения, вставляется символ пробела, а нажатие на клавишу **<Enter>** приводит еще и к переводу строки. Двойной щелчок левой клавишей мыши соответствует нажатию на клавишу **<Tab>**.

Как видно из названия, *Список свойств/методов* (*List Properties/Methods*) содержит список подходящих по контексту свойств и методов объекта. Этими объектами могут быть формы, отчеты, элементы управления, а также переменные типа объектов. Например, если в разделе описаний процедуры указана переменная *MyBase* типа *Database*, то в списке для нее будут представлены свойства и методы, характерные только для объектов *Database*.

В *списке констант* (*List Constant*) содержатся подходящие в текущем контексте программы константы. Чаще всего список констант отображается при вводе параметров функций или процедур. Например, при вводе значения параметра *Buttons* оператора (или функции) сообщения *MsgBox*, редактор Visual Basic предлагает набор констант, определяющих состав кнопок и тип значка окна сообщения. Зачастую можно указать не одну константу, а их комбинацию. В приведенном примере после ввода из списка константы, определяющей состав кнопок окна сообщения, можно ввести символ **+** и список констант снова отобразится на экране, позволяя указать тип значка для окна.

Команда *Завершить слово* (*Complete Word*) является комбинацией двух вышеописанных команд. Она доступна из контекстного меню окна модуля или главного меню редактора Visual Basic (одноименный пункт меню **Правка** (**Edit**)). Соответствующая клавиатурная комбинация — **<Ctrl>+<Space>**. Использование этой команды приводит, в зависимости от контекста, к отображению либо списка подходящих констант, либо списка подходящих свойств или методов.

Отображение на экране всплывающей подсказки об именах, типах, значениях по умолчанию параметров функций или процедур (*Подсказка* (*Quick Info*) и *Информация о параметрах* (*Parameter Info*)) также может использоваться либо автоматически, либо посредством выбора соответствующей команды контекстного меню окна модуля или главного меню (см. рис.9.6).

|  |
| --- |
| http://matveev.kiev.ua/macceass/ch1/gl9/images/m6.gif |
| **Рисунок 9.6.** Всплывающая подсказка с параметрами функции ***MsgBox***. |

В этой всплывающей подсказке жирным шрифтом выделяется наименование параметра, значение которого необходимо ввести в данный момент. Как только вы указали значение параметра и поставили запятую, жирным шрифтом выделяется следующий параметр и т.д. Разница между обеими командами заключается в том, что если одним из параметров является функция, команда*Подсказка* (*Quick Info*) включает в подсказку информацию о параметрах и этой функции. Команда *Информация о параметрах* (*Parameter Info*) этого не делает. Кроме того, команда *Подсказка*(*Quick Info*) может отображать информацию о переменных, объектах, операторах и методах.

Достаточно часто всплывающие подсказки очень громоздки и занимают много места на экране. Если это раздражает, можно отключить их автоматическое использование. Для этого откройте диалоговое окно *Параметры* (*Options*), перейдите на вкладку *Редактор* (*Editor*) и снимите пометку с флажка *Автоматическое отображение подсказки* (*Auto Quick Info*).

Еще одним из методов, используемых редактором Visual Basic в окне модуля для облегчения написания программ, является автоматическая проверка синтаксиса оператора. Как только текстовый курсор покидает строку программы, которая была отредактирована или написана, редактор проверяет ее синтаксическую корректность. В случае ошибки строка будет выделена красным цветом и на экране будет отображено сообщение об ошибке компиляции. Отключить автоматическую проверку синтаксиса можно в диалоговом окне *Параметры* (*Options*), сняв пометку с флажка *Автоматическая проверка синтаксиса* (*Auto Syntax Check*), расположенного на вкладке *Редактор* (*Editor*).

Кроме того, окно модуля используется и для отладки программ. Именно в окне модуля устанавливаются или снимаются точки прерывания, отображается маркер следующего выполняемого оператора (*Call Stack Marker*) в режиме прерывания выполнения (*Break Mode*). Используя возможность пошагового выполнения программы, можно отслеживать последовательность выполнения ее операторов, проверять промежуточные значения переменных или выражений. Большинство средств для вычисления значений переменных и выражений в точке останова программы размещено в окнах *непосредственной отладки* (*Immediate Window*), *локальных переменных* (*Locals Window*) и *контрольных выражений* (*Watch Window*). Можно, однако, обойтись и без этих окон. Для того чтобы проверить значение переменной или выражения на момент останова программы, подведите к имени переменной или выделенному выражению указатель мыши. Через мгновение на экране отобразится всплывающее окно, содержащее требуемое значение. Если этого не происходит, проверьте, установлен ли флажок *Автоматическое отображение значения выражения* (*Auto Data Tips*), расположенный на вкладке *Редактор* (*Editor*) диалогового окна *Параметры* (*Options*).

Как и в любом другом текстовом редакторе, все операции в окне модуля удобнее производить при помощи клавиатуры, не хватаясь через каждые несколько секунд за мышь. Зная клавиатурные эквиваленты основных операций, выполняемых при помощи мыши, можно значительно увеличить эффективность своей работы, не отвлекаясь на лишние движения и мысли. В таблице 9.2 приведены основные клавиатурные комбинации, используемые в окне модуля (*Code Window*), большинство из которых применимы и в других текстовых редакторах.

|  |  |
| --- | --- |
| **Таблица 9.2.** Комбинации клавиш, наиболее часто используемые в окне модуля. | |
| **Комбинация клавиш** | **Команда** |
| Ctrl+F | Найти (Find) |
| Ctrl+H | Заменить (Replace) |
| F3 | Найти следующее вхождение (Find Next) |
| Shift+F3 | Найти предыдущее вхождение (Find Previous) |
| Ctrl+<стрелка вниз> | Следующая процедура (Next Procedure) |
| Ctrl+<стрелка вверх> | Предыдущая процедура (Previous Procedure) |
| Shift+F2 | Переход к определению (View Definition) |
| Ctrl+PageDown | Сдвиг на один экран вниз (Shift one screen down) |
| Ctrl+PageUp | Сдвиг на один экран вверх (Shift one screen up) |
| Ctrl+Shift+F2 | Переход к последней позиции (Go to last position) |
| Ctrl+Home | Переход к началу модуля (Beginning of module) |
| Ctrl+End | Переход к концу модуля (End of module) |
| Ctrl+<стрелка вправо> | Переход на одно слово вправо (Move one word to right) |
| Ctrl+<стрелка влево> | Переход на одно слово влево (Move one word to left) |
| Home | Переход в начало строки (Move to beginning of line) |
| End | Переход в конец строки (Move to end of line) |
| Ctrl+Z | Отмена (Undo) |
| Ctrl+Y | Удалить текущую строку (Delete current line) |
| Ctrl+Del | Удалить символы от текущей позиции курсора до конца слова (Delete to end of word) |
| Ctrl-Shift-F9 | Убрать все точки прерывания (Clear all breakpoints) |
| Shift-F10 | Отобразить контекстное меню |

**9.3. Окно просмотра объектов (Object Browser Window)**

*Окно просмотра объектов* (*Object Browser Window*) позволяет в удобной форме просматривать все доступные в проекте объекты, их свойства и методы. Отобразить на экране окно просмотра объектов можно несколькими способами: нажав кнопку **Просмотр Объектов** (**Object Browser**) панели инструментов, выбрав одноименный пункт меню **Вид** (**View**) или нажав клавишу **<F2>**. Окно просмотра объектов содержит одиннадцать основных элементов (см. рис. 9.7) — два поля со списком, два списка, семь кнопок и поле, называемое *область описания*.

|  |
| --- |
| http://matveev.kiev.ua/macceass/ch1/gl9/images/m7.gif |
| **Рисунок 9.7.** Окно *просмотра объектов* (*Object Browser*). |

В левой верхней части окна расположено поле со списком *Библиотеки* (*Library Box*), содержащее наименования всех подключенных к проекту библиотек объектов. Среди этих библиотек могут быть стандартные библиотеки Microsoft Access, библиотека объектов, созданных в текущем проекте, а также другие библиотеки, подключенные в диалоговом окне *Ссылки* (*References*). В этом поле со списком можно выбрать наименование конкретной интересующей вас библиотеки, а можно выбрать значение *<Все библиотеки>* (*<All Libranes>*). Во втором случае в списке будут отображаться все доступные в проекте объекты.

В левой части окна расположен список *Объекты* (*Classes*), содержащий все объекты (классы) выбранной библиотеки. После выбора конкретного объекта, в списке *Свойства/Методы* (*Members of...*), расположенном в правой части окна, будут отображены все его свойства и методы (члены класса). Таким образом организована трехуровневая иерархическая система представления*библиотека — объект — свойство/метод*.

Окно просмотра объектов содержит средства поиска элемента по заданному тексту. Критерий поиска задается в поле со списком *Текст поиска* (*Search Text*). После нажатия кнопки **Поиск**(**Search**) с изображением бинокля производится попытка отыскать в указанной библиотеке объектов элемент, в наименование которого входит указанная в качестве критерия поиска строка. После завершения поиска в окне просмотра объектов отображается новый элемент — таблица с *результатами поиска* (*Search Results*). Эта таблица содержит три столбца — *Библиотека*(*Library*), *Объект* (*Class*) и *Методы/свойства* (*Members*). На рис. 9.8 приведены результаты поиска по критерию *«Form\_Current»*. Скрыть таблицу с результатами поиска можно, нажав кнопку с изображением двойной стрелки вверх. Нажав эту же кнопку еще раз, можно отобразить ее снова (правда, изображение на ней будет уже в виде двойной стрелки вниз).

|  |
| --- |
| http://matveev.kiev.ua/macceass/ch1/gl9/images/m8.gif |
| **Рисунок 9.8.** Окно *просмотра объектов* (*Object Browser*) с раскрытым списком результатов поиска. |

Из окна просмотра объектов можно перейти к описанию выбранного элемента — объекта (класса) или метода в окне модуля. Для этого нужно нажать кнопку *Просмотр определения* (http://matveev.kiev.ua/macceass/ch1/gl9/images/p7.gif —*View Definition*). Однако эта кнопка доступна не для всякого элемента, а только для выделенного жирным шрифтом. Жирным шрифтом выделяются элементы, чьи описания присутствуют в проекте. Например, это относится к процедурам модулей форм и отчетов, созданных разработчиком.

Использование окна просмотра объектов имеет практический смысл в большом проекте, имеющем много форм, отчетов, классов. Тогда, пожалуй, это единственное средство, позволяющее не заблудиться в дебрях окон модулей и программного кода. Хотя, с другой стороны, окно просмотра объектов может с успехом использоваться и для изучения стандартных объектов, их свойств и методов.