ЧАСТЬ II

Устранение нарушений в работе

В ЭТОЙ ЧАСТИ...

ГЛАВА 16

Отладка — ключ к успешной разработке

В ЭТОЙ ГЛАВЕ...

- Почему важна эта глава
- Предотвращение ошибок
- Использование возможностей окна Просмотр значений переменных (Immediate)
- Вызов отладчика
- Использование точек останова для поиска причин нарушений в работе
- Пошаговое выполнение кода
- Определение следующей выполняемой инструкции
- Использование окна Вызовы (Calls)
- Работа с окном Локальные (Locals)
- Работа с контрольными выражениями
- Продолжение работы после возникновения ошибки этапа прогона
- Нюансы, связанные с использованием окна Просмотр значений переменных (Immediate)
- Применение утверждений
- Рекомендации по отладке
- Практические примеры: отладка реальных приложений

Почему важна эта глава

Хорошим программистом не считается именно тот, кто может добиться требуемого результата с первого раза. В процессе работы часто приходится постепенно приближаться к намеченной цели, поэтому, чтобы стать полностью освоившим свою специальность программистом на VBA (Visual Basic for Applications), необходимо овладеть искусством *отладки*, которая представляет собой процесс устранения нарушений в работе приложения. Отладка требует поиска и выявления проблемных областей в коде и является обязательным шагом в процессе разработки приложений. К счастью, в редакторе VBE (Visual Basic Editor) программы Access 2007 предусмотрены великолепные средства, позволяющие упростить процесс отладки. С помощью средств отладки Access 2007 можно выполнять код в пошаговом режиме, устанавливая по мере необходимости контрольные точки и точки останова.

Отладка с помощью средств VBA осуществляется намного более эффективно по сравнению с тем подходом к отладке приложения, когда разработчик пытается вносить исправления бессистемно. Надежное овладение средствами отладки Access 2007 позволяет разработчику сберечь долгие часы, которые в про-

тивном случае были бы потрачены на проведение проб и исправление все новых и новых ошибок. Фактически именно в этом может состоять различие между успешно проводимым процессом разработки приложения и таким подходом, в котором борьба с ошибками продолжается неопределенно долго, а проблемы так и остаются нерешенными.

Предотвращение ошибок

Наилучший способ борьбы с ошибками состоит в том, чтобы с самого начала избежать их появления. В этом отношении могут оказать реальную помощь надлежащие методы разработки кода. Исключению ошибок в коде способствует применение инструкции `Option Explicit`, строгий контроль типов, соблюдение стандартов именования и продуманное распределение переменных с учетом областей определения.

Инструкция `Option Explicit`

После ввода в действие инструкции `Option Explicit` разработчик вынужден объявлять все предусмотренные в коде переменные до их использования. Включение инструкции `Option Explicit` в каждый модуль формы, кода и отчета приводит к тому, что опечатки в именах переменных обнаруживает сам компилятор VBA.

Как было подробно описано в главе 8, инструкция `Option Explicit` представляет собой команду, которую разработчик может вручную вставить в раздел Общие объявления (General Declarations) любого модуля кода, формы или отчета. Но при желании можно предусмотреть автоматическую вставку инструкции `Option Explicit` программой Access. Чтобы добиться этого, выберите элемент Требовать объявления переменных (Require Variable Declaration) на вкладке Редактор (Editor) после выбора команды Инструменты⇨Параметры (Tools⇨Options) в редакторе Visual Basic Editor. После того как будет выбрано это значение, программа Access вставляет инструкцию `Option Explicit` в раздел Общие объявления (General Declarations) всех новых модулей. Но эта настройка не затрагивает существующие модули.

Строгий контроль типов

Процесс строгого контроля типов переменных рассматривается в главе 8. Под строгим контролем типов подразумевается объявление каждой переменной с указанием на то, данные какого типа должны храниться в этой переменной. Например, объявление `Dim intCounter As Integer` указывает, что должна быть инициализирована переменная, предназначенная для хранения целых чисел. Если где-либо в коде будет предпринята попытка присвоить переменной `intCounter` символьную строку, то компилятор обнаружит эту ошибку.

Соблюдение стандартов именования

Значительную помощь в создании приложений, не содержащих ошибок, может также оказать применение стандартов именования. Применение тщательно продуманных имен переменных позволяет упростить чтение кода и сделать более понятным предусматриваемое назначение каждой переменной. Если в программе неуклонно соблюдаются соглашения об именовании, то становится понятней, на каком участке кода, скорее всего, может иметь место ошибка. Стандарты именования рассматриваются в главе 1. Но более подробные сведения о стандартах именования приведены в приложении А.

Область определения переменной

Наконец, следует отметить, что если для переменных отводится максимально узкая область определения, то снижается вероятность возникновения такой ситуации, когда в одной части кода непреднамеренно перезаписывается значение переменной, применяемой в другой части кода. Прежде всего, при любой возможности следует использовать только локальные переменные. Переменные, определяемые на уровне модуля, и глобальные переменные следует использовать только тогда, когда значение одной и той же переменной должно быть доступно из нескольких подпрограмм или модулей. Дополнительные сведения о том, какие вопросы связаны с применением областей определения переменных, приведены в главе 8.

Неизбежность возникновения ошибок

К сожалению, какие бы способы предотвращения проблем и ошибок не применялись, все равно могут возникнуть причины нарушений в работе кода. По-видимому, ошибками, сложнее всего поддающимися обнаружению, являются логические ошибки. Логические ошибки трудно уловимы, поскольку их не обнаруживает компилятор; компиляция кода проходит успешно, но после вызова кода на выполнение требуемый результат не достигается. Ошибки такого типа могут становиться очевидными при возникновении сбоя на этапе прогона или при обнаружении того, что не достигнуты ожидаемые результаты. В подобных случаях можно прибегнуть к помощи отладчика.

Использование возможностей окна Просмотр значений переменных (Immediate)

Окно Просмотр значений переменных (Immediate) имеет несколько назначений. Оно предоставляет превосходный способ проверки работы функций VBA и пользовательских функций, позволяет запрашивать и изменять значения переменных в процессе работы кода, а также дает возможность просматривать результаты выполнения инструкций `Debug.Print`. Чтобы открыть окно Просмотр значений переменных (Immediate) во время работы в программе Visual Basic Editor, воспользуйтесь одним из трех описанных ниже вариантов.

- Щелкните на инструменте Просмотр значений переменных (Immediate) на панели инструментов Отладка (Debug).
- Выберите команду Вид⇨Окно просмотра значений переменных (View⇨Immediate window).
- Нажмите комбинацию клавиш <Ctrl+G>.

НА ЗАМЕТКУ

Предпочтительным является вариант, предусматривающий нажатие комбинации клавиш <Ctrl+G>, поскольку он приводит к открытию окна Просмотр значений переменных (Immediate), притом что окно Программный код (Code) не становится активным. Возможность щелкнуть на кнопке панели инструментов окна Просмотр значений переменных (Immediate) или выбрать команду Вид⇨Окно просмотра значений переменных (View⇨Immediate window) предоставляется только во время работы с редактором VBE.

Окно Просмотр значений переменных (Immediate) показано на рис. 16.1.

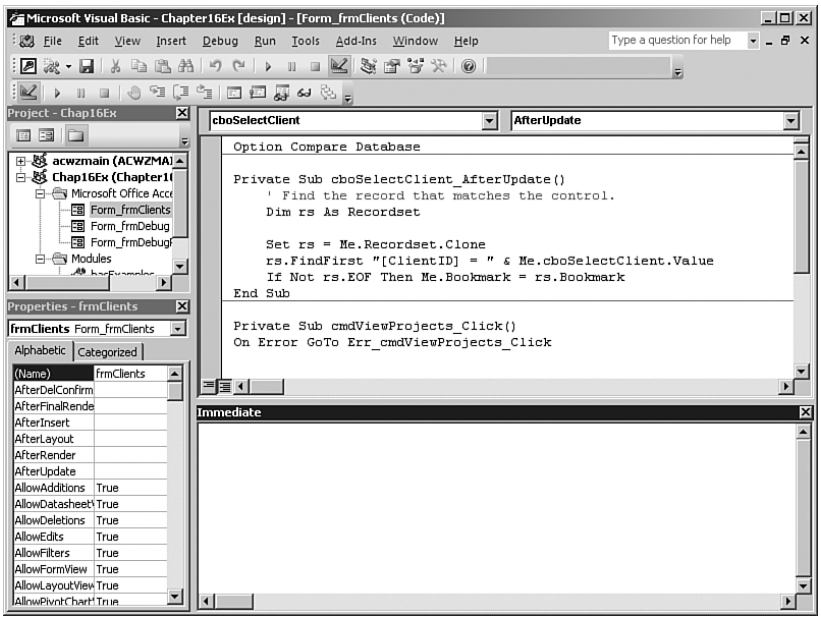

Рис. 16.1. *Окно* Просмотр значений переменных *(Immediate) позволяет проверять работу функций, а также запрашивать и изменять значения переменных*

НА ЗАМЕТКУ

На отдельной панели инструментов представлены инструменты Отладка (Debug). Чтобы вывести на экран панель инструментов Отладка (Debug), щелкните правой кнопкой мыши на любой панели инструментов или строке меню и выберите Отладка (Debug) в списке доступных панелей инструментов.

Проверка значений переменных и свойств

Окно Просмотр значений переменных (Immediate) позволяет проверять значения переменных и свойств во время выполнения кода. Такая возможность позволяет наглядно представить себе, что происходит при выполнении кода.

Чтобы освоить работу с окном Просмотр значений переменных (Immediate), даже не требуется вызывать на выполнение код. Для вызова окна Просмотр значений переменных (Immediate) в ходе применения формы, отчета или модуля нажмите комбинацию клавиш <Ctrl+G>. Чтобы ознакомиться с тем, как действует это окно, выполните приведенные ниже действия.

1. Вызовите на выполнение форму `frmClients` из базы данных `CHAP16EX.ACCDB`, которую можно найти на веб-узле, сопровождающем данную книгу.
2. Нажмите комбинацию клавиш <Ctrl+G>, чтобы открыть и активизировать окно Просмотр значений переменных (Immediate). Программа Access откроет окно Просмотр значений переменных (Immediate) в редакторе VBE.

3. Введите `?Forms!frmClients.txtClientID.Value` и нажмите клавишу <Ввод> (Enter). На следующей строке появится идентификатор текущего клиента.
4. Введите `?Forms!frmClients.txtCompanyName.Visible` и нажмите клавишу <Ввод> (Enter). На следующей строке появится слово `True`, которое указывает, что данный элемент управления является видимым.
5. Введите `?Forms!frmClients.txtContactTitle.BackColor` и нажмите клавишу <Ввод> (Enter). На следующей строке появится значение, связанное со свойством `BackColor` текстового поля `Contact Title`.

Откроется окно, показанное на рис. 16.2. Продолжите эксперименты по получению значений свойств или переменных в используемом коде VBA.

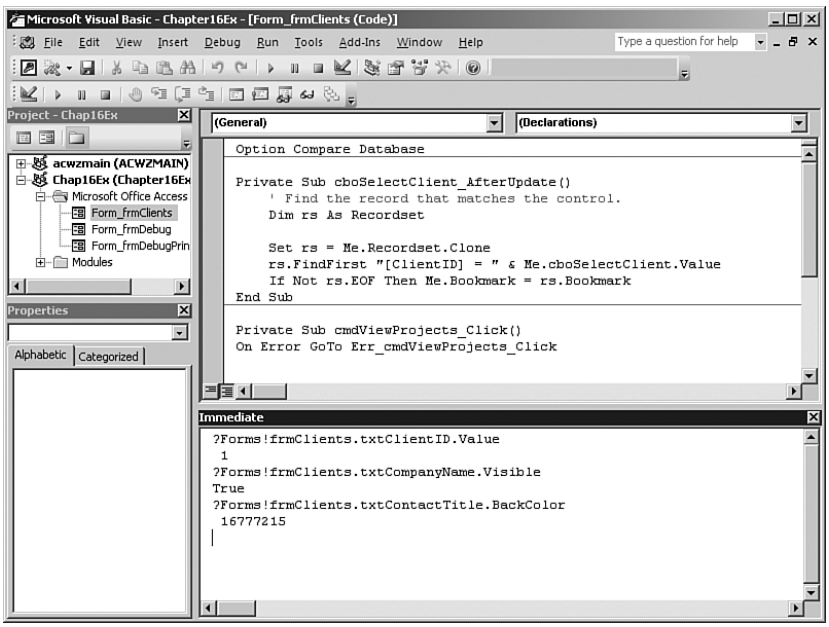

Рис. 16.2. *Применение окна* Просмотр значений переменных *(Immediate) для проверки значений свойств*

Установка значений переменных и свойств

Окно Просмотр значений переменных (Immediate) позволяет не только определять, чему равны те или иные значения, но и модифицировать значения переменных и элементов управления во время выполнения кода. Такая возможность становится еще более ценной, если возникает необходимость повторного выполнения кода процедуры после изменения значения переменной. Ниже описано, как можно осуществить такой процесс.

1. В случае необходимости откройте окно Просмотр значений переменных (Immediate). Напомним, что это можно сделать, нажав комбинацию клавиш <Ctrl+G>.
2. Введите `Forms!frmClients.txtContactTitle.Value = "Hello"` в окне Просмотр значений переменных (Immediate). Нажмите клавишу <Ввод> (Enter). Заголовок, обозначающий текущую строку, изменится и в нем появится слово `Hello`.

3. Введите выражение `Forms!frmClients.txtIntroDate.Visible = False`. Нажмите клавишу <Ввод> (Enter). Программа Access сделает невидимым (скрытым) элемент управления `txtIntroDate` в форме `frmClients`.
4. Введите выражение `Forms!frmClients.txtClientID.BackColor = 123456`. Нажмите клавишу <Ввод> (Enter). Цвет фона элемента управления `txtClientID` в форме `frmClients` станет зеленым. После этого окно Просмотр значений переменных (Immediate) и форма должны принять такой вид, как показано на рис. 16.3 и 16.4 соответственно.

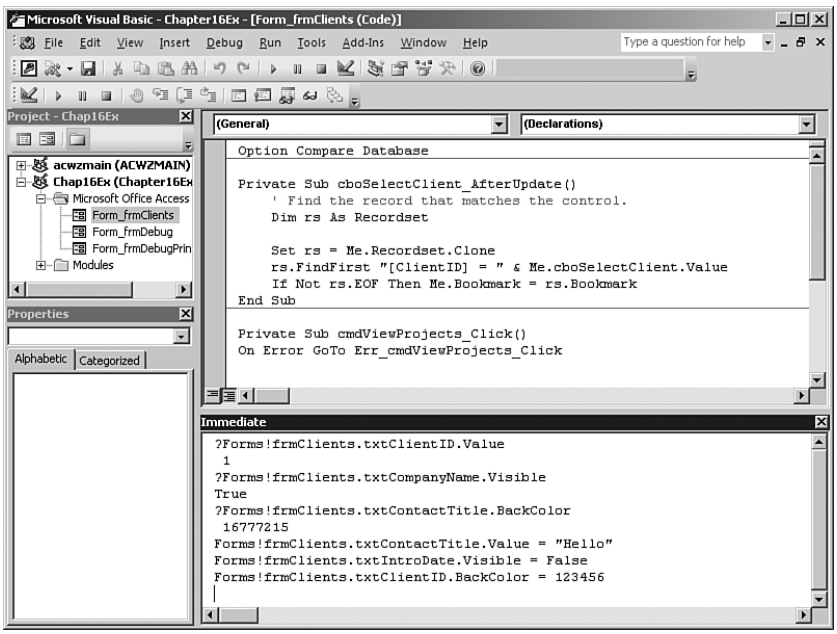

Рис. 16.3. *Определение значений свойств с помощью окна* Просмотр значений переменных *(Immediate)*

Окно Просмотр значений переменных (Immediate) представляет собой чрезвычайно ценное средство проверки и отладки приложений. Примеры, приведенные в настоящей главе, можно считать лишь слабым отражением его возможностей и разнообразия применения.

ВНИМАНИЕ!

Изменения, внесенные в данные во время работы в окне Просмотр значений переменных (Immediate), являются постоянными. С другой стороны, программа Access не сохраняет изменения, внесенные в свойства элементов управления или значения переменных, применяемых в форме или отчете.

Некоторые пользователи считают, что изменения в данных, внесенные в окне Просмотр значений переменных (Immediate), не являются постоянными. Иными словами, некоторые разработчики считают, что после изменения, допустим, фамилии клиента это изменение остается временным (но в действительности оно является постоянным). Другие

разработчики полагают, например, что после изменения значения свойства `BackColor` элемента управления это изменение сохранится в среде проектирования (но, безусловно, этого не происходит).

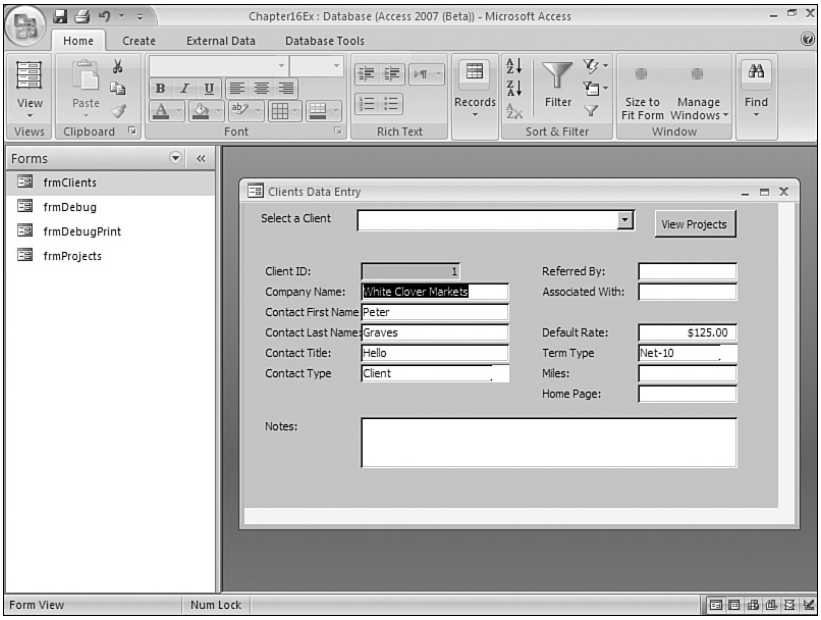

Рис. 16.4. *Окно, в котором показаны результаты использования окна* Просмотр значений переменных *(Immediate) для задания значений свойств*

Очистка окна Просмотр значений переменных (Immediate)

В окне Просмотр значений переменных (Immediate) отображаются последние 200 строк вывода. По мере ввода дополнительных строк кода в окно Просмотр значений переменных (Immediate) те строки, которые были введены ранее, исчезают. А после того как пользователь выйдет из программы Access, окно Просмотр значений переменных (Immediate) очищается. Но если потребуется очистить окно Просмотр значений переменных (Immediate) в любое другое время, выполните описанные ниже три действия.

1. Когда будет активизировано окно Просмотр значений переменных (Immediate), нажмите комбинацию клавиш <Ctrl+ Home>, чтобы перейти в верхнюю часть окна Просмотр значений переменных (Immediate).
2. Удерживая нажатой клавишу <Shift>, нажмите комбинацию клавиш <Ctrl+End>, чтобы перейти на последнюю инструкцию в окне Просмотр значений переменных (Immediate).
3. Нажмите клавишу <Удалить> (Delete).

Проведение экспериментов со встроенными функциями

Окно Просмотр значений переменных (Immediate) позволяет не только проверять и задавать значения свойств и переменных, но и проверять работу любых функций VBA.

Чтобы провести такой эксперимент, введите имя функции с указанием параметров в окне Просмотр значений переменных (Immediate), поставив перед именем вопросительный знак. Например, следующий код возвращает месяц, относящийся к текущей дате:

```
?datepart("m",date)
```

В следующем коде формируется дата, отстоящая на один месяц от сегодняшней даты:

```
?dateadd("m",1,date)
```

Этот код позволяет узнать, сколько суток должно пройти от текущей даты до конца тысячелетия:

```
?datediff("d",date(),#12/31/2999#)
```

Вызов на выполнение подпрограмм, функций и методов

Окно Просмотр значений переменных (Immediate) позволяет не только проверить работу любой функции VBA, но и выполнить проверку любой определяемой пользователем подпрограммы, функции или метода. В этом состоит превосходный способ отладки процедур, определяемых пользователем. Для ознакомления с тем, как применяется окно Просмотр значений переменных (Immediate) в указанных целях, выполните описанные ниже действия.

1. Откройте модуль `basExamples`, находящийся в базе данных `CHAP16EX.ACCDB` на веб-узле, сопровождающем данную книгу.
2. Откройте окно Просмотр значений переменных (Immediate), если оно еще не является видимым.
3. Введите выражение `?ReturnInitsFunc("Bill","Gates")`. Это приведет к вызову пользовательской функции `ReturnInitsFunc` и передаче ей `"Bill"` в качестве первого параметра и `"Gates"` — в качестве второго. В окне Просмотр значений переменных (Immediate) появится значение `B.G.` Таковым является значение, возвращаемое функцией.
4. Введите выражение `Call ReturnInitsSub("Bill","Gates")`. Это приведет к вызову определяемой пользователем подпрограммы `ReturnInitsSub` и передаче значения `"Bill"` в качестве первого параметра и значения `"Gates"` — в качестве второго. Значение `B.G.` появится в окне сообщения.

Обратите внимание на то, что вызовы функции и подпрограммы осуществляются по-разному. Функция возвращает значение, поэтому ее следует вызывать с использованием вопросительного знака. При вызове подпрограммы применяется ключевое слово `Call`.

НА ЗАМЕТКУ

Можно также вызвать подпрограмму в окне Просмотр значений переменных (Immediate) с помощью синтаксиса

```
RoutineName Parameter1, Parameter2, ...
```

Если ключевое слово `Call` опущено, то нет необходимости заключать параметры в круглые скобки.

Вывод данных в окно Просмотр значений переменных **(Immediate) на этапе прогона**

Возможность вывода данных в окно Просмотр значений переменных (Immediate) является очень удобной, поскольку позволяет следить за тем, что происходит в ходе работы кода, не приостанавливая выполнение кода. Является также ценной возможность выводить требуемые данные в окно в процессе проверки программы, не нарушая работу той части кода, которая относится к эксплуатации пользовательского интерфейса. В частности, можно проверить работу формы, не прерывая ее эксплуатацию, а затем возвратиться к форме и просмотреть значения переменных, и т.д. Ниже описано, как можно осуществить такой процесс.

1. Введите выражение `Call LoopThroughCollection` в окне Просмотр значений переменных (Immediate), чтобы вызвать определяемую пользователем подпрограмму `LoopThroughCollection`. В окне отобразятся значения переменных `Skating`, `Basketball`, `Hockey` и `Skiing`. Эти значения в окно Просмотр значений переменных (Immediate) выводит процедура.

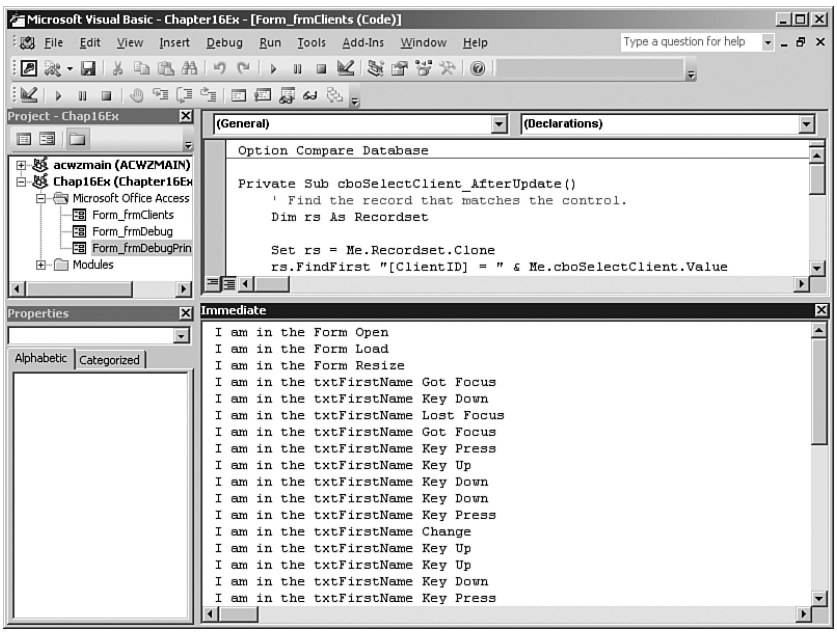

Рис. 16.5. *Инструкция* `Debug.Print`*, предназначенная для вывода значений в окно* Просмотр значений переменных *(Immediate)*

2. Откройте форму `frmDebugPrint` в представлении Форма (Form).
3. Нажмите клавишу <Tab>, чтобы перейти от поля Имя (First Name) к полю Фамилия (Last Name).
4. Нажмите клавишу <Tab>, чтобы вернуться к полю Имя (First Name).
5. Введите свое имя.

6. Откройте окно Просмотр значений переменных (Immediate). Обратите внимание на то, что данная процедура отправила результаты выполнения всех инструкций в окно Просмотр значений переменных (Immediate) (рис. 16.5). Сам автор применяет подобные инструкции `Debug.Print` в обработчиках событий форм и элементов управления во всех случаях, когда в этом возникает необходимость.

НА ЗАМЕТКУ

Безусловно, рекомендуется удалять инструкции `Debug.Print` после завершения процесса отладки, но не наступит никаких отрицательных последствий, если приложение будет развернуто без их удаления. Пользователи смогут узнать о существовании таких инструкций в коде, только если откроют окно Просмотр значений переменных (Immediate). Применение инструкций `Debug.Print` приводит лишь к незначительному снижению производительности.

Вызов отладчика

Для вызова отладчика Access можно применять несколько способов, которые описаны ниже.

- Введение в код точки останова.
- Введение в код контрольного значения.
- Нажатие комбинации клавиш <Ctrl+Break> во время выполнения кода.
- Вставка в код инструкции `Stop`.

Точка останова представляет собой такую позицию в коде, в которой выполнение кода незамедлительно приостанавливается. Сами точки останова являются временными, поскольку они остаются в силе только на то время, пока открыта база данных. Иными словами, программа Access не сохраняет точки останова в базе данных.

Контрольное значение представляет собой условие, при выполнении которого должно быть приостановление выполнение кода. Необходимость в приостановке выполнения кода может, например, возникнуть, если переменная счетчика достигает определенного значения. Контрольное значение также является временным; программа Access удаляет его после закрытия базы данных.

А инструкция `Stop` является постоянной. В действительности, если разработчик забывает удалить инструкции `Stop` из кода перед передачей в эксплуатацию, то приложение останавливается и в то время, как с ним работает пользователь.

Использование точек останова для поиска причин нарушений в работе

Как уже было сказано, точка останова представляет собой такую позицию в коде, в которой программа Access останавливает выполнение кода, какими бы ни были сложившиеся обстоятельства. Предусмотрена возможность задавать в коде произвольное количество точек останова. Кроме того, можно добавлять и удалять точки останова в процессе выполнения кода.

Точка останова позволяет прекратить выполнение кода в той его области, которая внушает подозрение. Благодаря этому появляется возможность проверить все, что происходит на этом этапе выполнения кода. Продуманное размещение точек останова в коде позволяет быстро выполнять уже отлаженные разделы кода, останавливаясь только в проблемных областях.

Чтобы задать точку останова, выполните описанные ниже действия.

1. Поместите курсор на строке программы, в которой должен быть вызван отладчик.
2. Можно вставить точку останова с помощью одного из четырех приведенных ниже способов.
   - Нажмите функциональную клавишу <F9>.
   - Щелкните на серой области поля слева от строки кода, в которой должна находиться точка останова.
   - Щелкните на кнопке Добавить/удалить точку останова (Toggle Breakpoint) панели инструментов Отладка (Debug).
   - Выберите элемент Отладка⇨Добавить/удалить точку останова (Debug⇨Toggle Breakpoint).

   Строка кода, содержащая точку останова, отображается шрифтом другого цвета, и появляется точка, обозначающая точку останова.
3. Вызовите на выполнение форму, отчет или модуль, содержащие точку останова. В коде VBA выполнение приостанавливается непосредственно перед строкой кода, на которой находится точка останова. Инструкция, которая должна быть выполнена, отображается шрифтом контрастного цвета. (По умолчанию применяется желтый цвет.)

После приостановки работы кода можно переходить к режиму пошагового выполнения, в котором одновременно обрабатывается только одна строка, изменять значения переменных, просматривать стек вызовов и осуществлять многие другие действия по отладке.

Следует учитывать, что точка останова фактически действует как выключатель. Если необходимо удалить точку останова, щелкните на серой области поля, нажмите клавишу <F9> или щелкните на элементе Добавить/удалить точку останова (Toggle Breakpoint) панели инструментов Отладка (Debug). Программа Access удаляет точки останова после закрытия базы данных, открытия другой базы данных или выхода из Access.

Самый легкий способ изучения работы отладчика состоит в его использовании на практике. Следующий пример позволяет получить реальный опыт в настройке точки останова и прекращении выполнения кода в этой точке. Разработка кода, рассматриваемого в этом примере, будет продолжена ниже в этой главе.

Начнем с создания формы `frmDebug`, которая содержит управляющую кнопку `cmdDebug`. Определите для кнопки заголовок Начать процесс отладки (Start Debug Process). Поместите следующий код в обработчик событий Щелчок (Click) кнопки:

```
Sub cmdDebug_Click ()
    Call Func1
End Sub
```

Создайте новый модуль и присвойте ему имя `basFuncs`. Введите в модуль следующие три функции:

```
Sub Func1 ()
   Dim intTemp As Integer
   intTemp = 10
   Debug.Print "We Are Now In Func1()"
   Debug.Print intTemp
   Call Func2
End Sub
Sub Func2 ()
   Dim strName As String
   strName = "Bill Gates"
   Debug.Print "We Are Now In Func2()"
   Debug.Print strName
   Call Func3
End Sub
Sub Func3 ()
   Debug.Print "We Are Now In Func3()"
   MsgBox "Hi There From The Func3() Sub Procedure"
End Sub
```

Теперь можно приступать к отладке. Прежде всего поместите точку останова в обработчик событий Щелчок (Click) кнопки `cmdDebug` на строке с текстом `Call Func1`. Ниже показано, как это сделать.

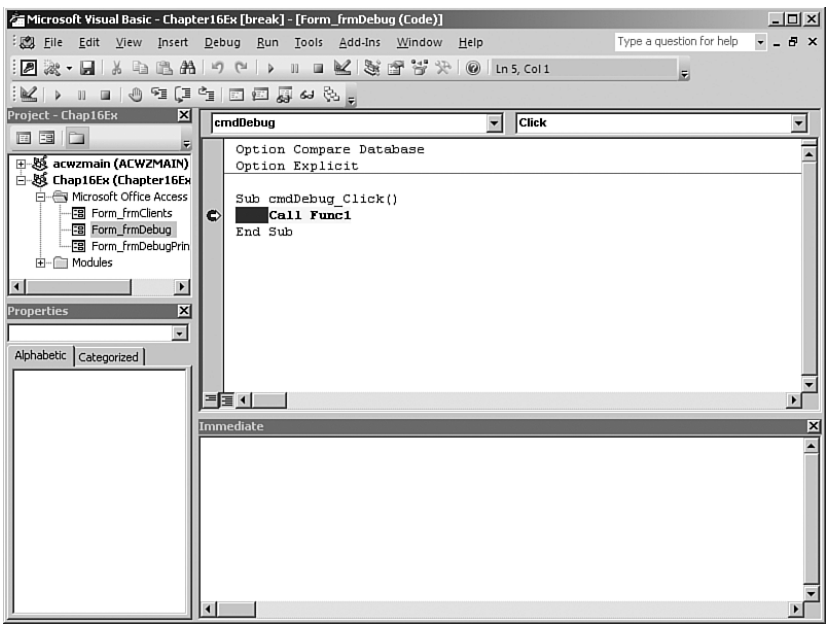

Рис. 16.6. *Прекращение выполнения кода в точке останова*

1. Щелкните в том месте строки кода, где находится текст `Call Func1`.
2. Щелкните на серой области поля, нажмите функциональную клавишу <F9>, щелкните на кнопке Добавить/удалить точку останова (Toggle Breakpoint) панели инструментов Отладка (Debug) или выберите элемент Отладка⇨Добавить/удалить точку останова (Debug⇨Toggle Breakpoint). Строка с точкой останова отобразится шрифтом другого цвета (по умолчанию — красного).

3. Перейдите к режиму формы и щелкните на кнопке Начать процесс отладки (Start Debug Process). Программа Access приостановит работу непосредственно перед вызовом на выполнение той строки, в которой помещена точка останова. В коде VBA строка с текстом `Call Func1` будет выведена с использованием шрифта другого цвета (по умолчанию — желтого), а это означает, что данная строка должна быть выполнена в следующую очередь (рис. 16.6).

Пошаговое выполнение кода

В программе Access 2007 предусмотрены три основных способа пошагового выполнения кода. Но различия между этими способами невелики. Вариант с применением команды Шаг с заходом (Step Into) позволяет последовательно выполнять каждую строку кода в подпрограмме или функции, а команда Шаг с обходом (Step Over) предусматривает выполнение процедуры без пошаговой обработки каждой содержащейся в ней строки кода. В варианте с применением команды Шаг с выходом (Step Out) выполняется весь код вложенной процедуры, после чего происходит возврат к процедуре, из которой вызвана на выполнение текущая строка кода. Чтобы успешно решать конкретные проблемы отладки, необходимо знать, какой из указанных вариантов должен использоваться в том или ином случае, но такие навыки можно приобрести, лишь постоянно участвуя в разработке.

Использование команды Шаг с заходом (Step Into)

После достижения точки останова можно продолжить выполнение кода, переходя последовательно с одной строки на другую, или перейти в обычный режим выполнения до достижения следующей точки останова. Чтобы обеспечить выполнение кода по одной строке одновременно, щелкните на кнопке Шаг с заходом (Step Into) панели инструментов Отладка (Debug), нажмите клавишу <F8> или выберите элемент Отладка⇨Шаг с заходом (Debug⇨Step Into).

В следующем примере показан процесс пошагового выполнения кода, вывода значений переменных в окно Просмотр значений переменных (Immediate) и модификации значений переменных с использованием окна Просмотр значений переменных (Immediate).

В данном случае процесс отладки может быть продолжен от точки останова, которая была задана в предыдущем примере. Сделайте два шага в процессе пошагового выполнения (нажимая клавишу <F8>). После этого должен произойти переход к функции `Func1` и останов перед выполнением строки кода `intTemp = 10` (рис. 16.7). Обратите внимание на то, что в коде VBA не произошел останов на строке `Dim intTemp As Integer`. Отладчик не останавливается на объявлениях переменных.

На следующем этапе в этом коде должен произойти вывод с помощью инструкций Отладка (Debug) в окно Просмотр значений переменных (Immediate). Рассмотрим, произойдет ли это, открыв окно Просмотр значений переменных (Immediate). В рассматриваемом коде еще не встретилась ни одна инструкция, при выполнении которой произошел бы вывод в окно Просмотр значений переменных (Immediate). Нажмите клавишу <F8> (клавишу пошагового выполнения) еще три раза, до тех пор, пока не выполнится строка `Debug.Print intTemp`. Окно отладчика должно принять вид, показанный на рис. 16.8. Обратите внимание на результаты выполнения инструкций `Debug.Print`.

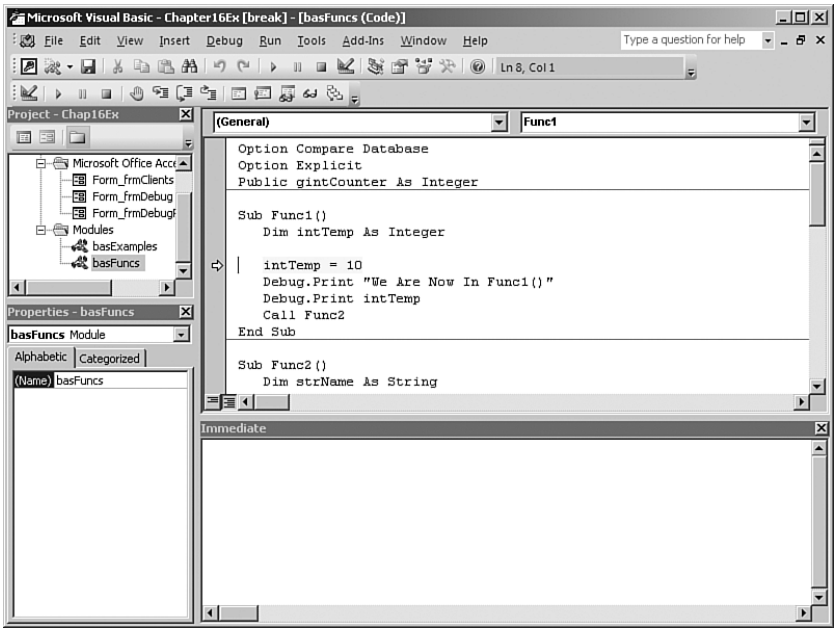

Рис. 16.7. В окне Просмотр значений переменных *(Immediate) произошел останов в коде функции* `Func1`

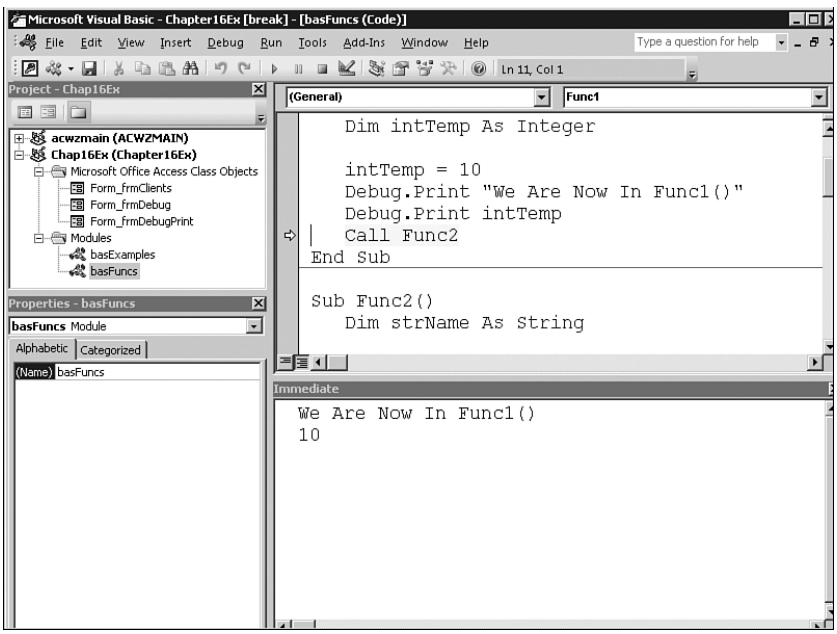

Рис. 16.8. В окне Просмотр значений переменных *(Immediate) отображаются строки, сформированные инструкциями* `Debug.Print`

Таким образом, мы ознакомились с тем, как можно вывести значения переменных и результаты вычисления выражений в окно Просмотр значений переменных (Immediate), а теперь рассмотрим способы использования окна Просмотр значений переменных (Immediate) для модификации значений переменных и свойств элементов управления. Начнем с изменения значения `intTemp`. Щелкните в окне Просмотр значений переменных (Immediate) и введите строку `intTemp = 50`. После нажатия клавиши <Ввод> (Enter) фактически происходит изменение значения `intTemp`. Введите выражение `?intTemp`, после чего можно убедиться в том, что программа Access выведет в ответ значение 50. Значение переменной `intTemp` отображается также в окне Локальные (Locals). Обратите внимание на то, что на рис. 16.9 переменная `intTemp` отображается с указанием ее значения и типа.

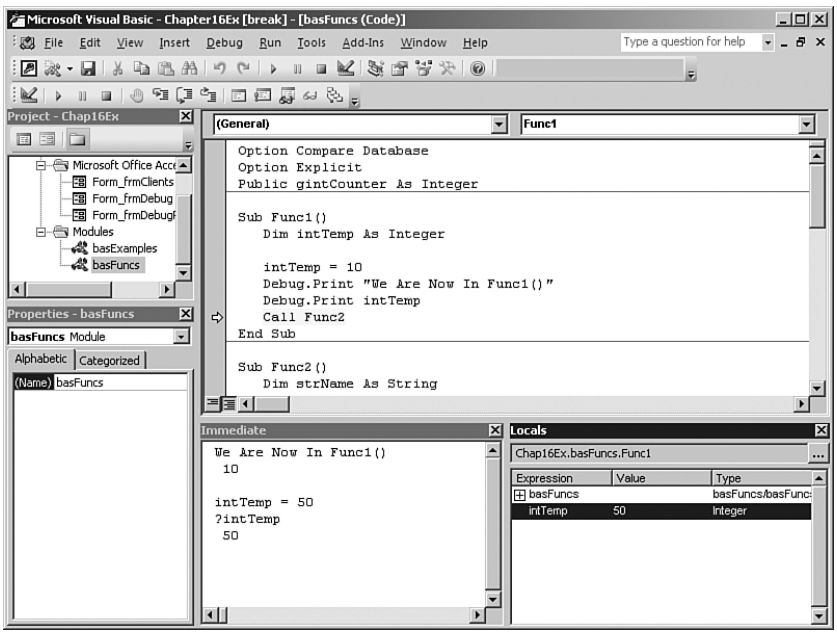

Рис. 16.9. *Окна* Просмотр значений переменных *(Immediate) и* Локальные *(Locals) после изменения значения* `intTemp`

Выполнение кода до достижения следующей точки останова

В процессе отладки иногда возникает такая ситуация, когда после достижения точки останова обнаруживается, что еще не достигнут тот участок кода, который является причиной возникновения нарушений в работе. Иными словами, фактически ошибка скрывается в коде другой функции. При этом может оказаться нецелесообразным дальнейшее продолжение пошагового выполнения кода до достижения той функции, которая на самом деле требует отладки. В таком случае можно воспользоваться раскрывающимся меню `Procedure`, чтобы найти функцию, вызывающую сомнение, а затем определить точку останова на той строке, в которой должно быть продолжено пошаговое выполнение кода. После этого можно продолжить выполнение кода до тех пор, пока программа Access не достигает указанной строки. Чтобы перейти к дальнейшему выполнению кода, щелкните на

кнопке Продолжить (Continue) панели инструментов Отладка (Debug), нажмите клавишу <F5> или выберите элемент Выполнить⇨Продолжить (Run⇨Continue). Выполнение кода будет продолжено, а в следующей точке останова прекратится. Чтобы проверить, как действует этот вариант отладки, продолжите процесс отладки на следующем примере.

НА ЗАМЕТКУ

Может быть также принято решение возобновить выполнение кода с той точки, на которой в данный момент находится курсор. Для этого выберите команду Выполнить до курсора (Run to Cursor) в меню Отладка (Debug) или нажмите комбинацию клавиш <Ctrl+F8>.

Предположим, стало ясно, что причиной неправильной работы приложения является код функции `Func3`. Но может оказаться нецелесообразным пошаговое перемещение в коде вплоть до функции `Func3`. Из этого положения можно легко найти выход. Воспользуйтесь раскрывающимся меню `Procedure`, чтобы найти функцию `Func3` (рис. 16.10). Определите точку останова на строке с текстом `Debug.Print "We Are Now In Func3()"`. После этого можно продолжить выполнение кода до тех пор, пока программа Access не достигнет этой строки. Щелкните на кнопке Продолжить (Continue) панели инструментов Отладка (Debug), нажмите клавишу <F5> или выберите элемент Выполнить⇨Продолжить (Run⇨Continue). Выполнение кода будет продолжено, а затем прекратится перед той точкой останова, которая была только что задана. Снова нажмите клавишу <F5>. Выполнение кода будет завершено. Возвратитесь к окну Режим формы (Form View).

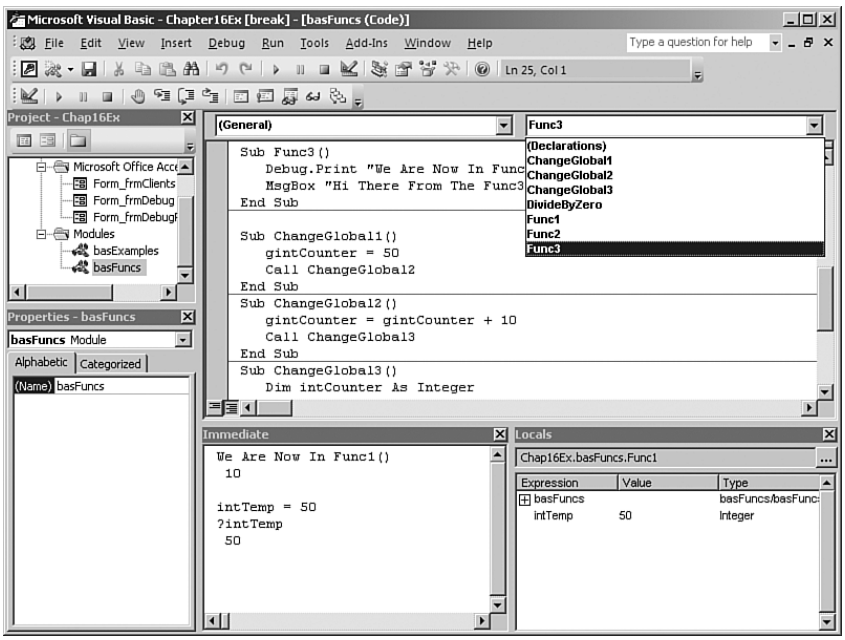

Рис. 16.10. Раскрывающееся меню `Procedure` *позволяет выбрать другую функцию*

Использование команды Шаг с обходом (Step Over)

Иногда подпрограммы, применяемые в коде, являются полностью отлаженными и проверенными. А стоящая перед вами задача состоит в том, чтобы продолжить пошаговое выполнение рассматриваемой процедуры, но нет необходимости следить за выполнением подпрограмм. В этом случае применяется команда Шаг с обходом (Step Over). Чтобы пропустить подпрограмму или функцию в ходе пошагового выполнения, щелкните на кнопке Шаг с обходом (Step Over) панели инструментов Отладка (Debug), нажмите комбинацию клавиш <Shift+F8> или выберите элемент Отладка⇨Шаг с обходом (Debug⇨Step Over). Код, находящийся в пропускаемой подпрограмме или функции, выполняется, но не происходит последовательный переход по строкам этого кода. Чтобы провести эксперименты со средством Шаг с обходом (Step Over), рассмотрим следующий пример.

Щелкните в открытой форме, а затем еще раз щелкните на кнопке Начать процесс отладки (Start Debug Process). Существующие точки останова не удалены, поэтому программа Access переходит к строке кода `Call Func1`. Выберите элемент Снять все точки останова (Clear All Breakpoints) в меню Отладка (Debug) или воспользуйтесь комбинацией клавиш <Ctrl+Shift+F9>, чтобы удалить все точки останова. Выполните в пошаговом режиме подряд пять строк кода (нажимая клавишу <F8>), пока очередной предназначенный для выполнения строкой не станет строка `Call Func2`. Предположим, что функции `Func2` и `Func3` уже проверены, поэтому известно, что они не могут служить причиной возникновения проблем в коде. После того как строка с вызовом функции `Func2` будет выделена подсветкой в коде как предназначенная для выполнения программой Access в следующую очередь, щелкните на кнопке Шаг с обходом (Step Over) панели инструментов. Программа Access выполнит функции `Func2` и `Func3`, и теперь все готово для того, чтобы продолжить пошаговое выполнение, начиная с функции `Func1`. В этом случае программа Access переходит к строке `End Sub`, которая непосредственно следует за строкой с вызовом функции `Func2`.

Использование команды Шаг с выходом (Step Out)

Средство Шаг с выходом (Step Out) можно использовать для выхода из процедуры, выполняемой в настоящее время, и возврата к процедуре, в которой была вызвана рассматриваемая строка кода. Этим средством можно воспользоваться, если непреднамеренно произошел переход к пошаговому выполнению процедуры, после чего стало очевидно, что она полностью проверена и не требует отладки. В этом случае необходимо выполнить весь код вызванной процедуры, к которой произошел переход, а затем вернуться в вызывающую процедуру, чтобы можно было продолжить процесс отладки. Чтобы ознакомиться на практике с работой этого средства, выполните следующий пример.

1. Определите точку останова на вызове функции `Func2`.
2. Щелкните на кнопке Reset (Сбросить) панели инструментов, чтобы остановить выполнение кода.
3. Активизируйте форму `frmDebug` и щелкните на кнопке Начать процесс отладки (Start Debug Process).
4. Сделайте один шаг в процессе пошагового выполнения, чтобы перейти на первую строку в коде функции `Func2`.

5. Предположим, после этого стало очевидно, что сделан один лишний шаг в ходе пошагового выполнения. Фактически вы намеревались пройти без останова функцию `Func2` и все процедуры, которые в ней вызываются. И в этом случае задача легко решается! Щелкните на кнопке Шаг с выходом (Step Out), чтобы выйти из пошагового выполнения функции `Func2` и возвратиться к строке, которая следует за строкой кода с вызовом функции `Func2`. В этом случае должен произойти переход к инструкции `End Sub` в коде функции `Func1`.

Определение следующей выполняемой инструкции

После пошагового выполнения кода, проверки правильности его логической организации и внесения изменений в некоторые переменные может потребоваться снова выполнить код, начиная с одной из предыдущих инструкций. Такую задачу можно проще всего решить, щелкая и перетаскивая желтую стрелку, находящуюся на поле, к той инструкции, с которой необходимо продолжить выполнение. При желании можно также щелкнуть в любом месте строки кода, с которой должно быть продолжено выполнение, а затем выбрать элемент Отладка⇨Задать следующий оператор (Debug⇨Set Next Statement). Независимо от выбранного вами метода, указанный оператор будет выделен шрифтом контрастного цвета (обычно желтого), а это служит обозначением строки кода, которая в программе Access будет выполнена в следующую очередь. После этого можно приступить к пошаговому выполнению кода, нажав клавишу <F8>, или продолжить обычное выполнение кода, нажав клавишу <F5>. Программа Access позволяет задавать следующую выполняемую строку кода только в процедуре. Это средство можно применять для повторного выполнения строк кода или для пропуска той строки кода, которая может стать причиной нарушения в работе приложения.

В следующем примере показано, как изменить значение переменной, а затем снова вызвать код на выполнение после изменения значения.

По результатам предыдущего примера мы остались на последней строке коде (на инструкции `End Sub`) в функции `Func1`. Предположим, что теперь необходимо изменить значение переменной `intTemp` и все выполнить повторно, как описано ниже.

1. Перейдите к окну Просмотр значений переменных (Immediate) и введите `intTemp = 100`.
2. Укажите в качестве следующей инструкции, с помощью которой выполняется печать, строку `Debug.Print "We Are Now in Func1()"`. Для этого щелкните и перетащите желтую стрелку с инструкции `End Sub` на строку `Debug.Print "We Are Now In Func1()"`. Обратите внимание на то, что эта строка будет выделена шрифтом контрастного (желтого) цвета, который указывает на то, что данная инструкция представляет собой следующую строку кода, предназначенную для выполнения программой Access.
3. Нажмите два раза клавишу <F8> (клавишу пошагового выполнения). Теперь код выполняется со значения переменной `intTemp`, установленным равным 100. Снова проверьте значения, показанные в окне Просмотр значений переменных (Immediate). Обратите внимание на то, как изменились результаты.

Использование окна Стек вызовов (Call Stack)

Выше в данной главе было описано, как задавать точки останова, осуществлять пошаговое выполнение кода и пропускать в ходе этого отлаженный код, использовать окно Просмотр значений переменных (Immediate), указывать следующую строку, подлежащую выполнению, и продолжать выполнение до достижения следующей точки останова. После перехода к некоторой точке останова часто возникает необходимость определить, какие функции были вызваны в коде, что привело к достижению данной точки. В этом случае может применяться средство Вызовы (Calls).

Чтобы открыть окно Стек вызовов (Call Stack), щелкните на кнопке Стек вызовов (Call Stack) панели инструментов или выберите элемент Вид⇨Стек вызовов (View⇨Call Stack) (рис. 16.11). Если необходимо узнать, с помощью какой строки кода была вызвана конкретная функция или подпрограмма, дважды щелкните на имени этой функции или щелкните на функции, а затем на кнопке Показать (Show). Безусловно, при этом программа Access не переводит точку выполнения на вызывающую функцию или подпрограмму, но появляется возможность просматривать код данной процедуры. Если требуется продолжить выполнение кода, нажмите клавишу <F8>. При этом снова происходит переход к той же процедуре, код которой обрабатывался в режиме пошагового выполнения, и выполняется следующая строка кода. А нажатие клавиши <F5> приводит к тому, что выполнение кода будет продолжено до достижения следующей точки останова или контрольного значения. Если требуется возвратиться к той строке, которая рассматривалась перед этим, без выполнения каких-либо еще строк кода, то выберите элемент Отладка⇨Отобразить следующий оператор (Debug⇨Show Next Statement).

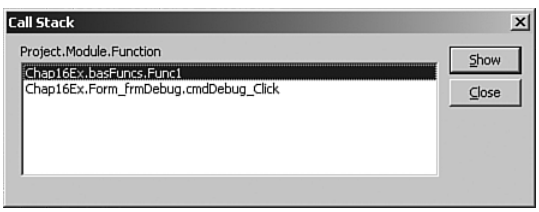

***Рис. 16.11**. Окно* Стек вызовов *(Call Stack) предоставляет возможность просматривать стек вызовов*

Чтобы проверить такую возможность на практике, выполните следующий пример.

1. Щелкните на кнопке Reset (Сбросить), чтобы остановить выполнение кода, если все еще применяется режим работы с точками останова.
2. Удалите точку останова со строки вызова функции `Func2`.
3. Перейдите к процедуре `Func3` модуля `basFuncs`. Задайте точку останова на строке `Debug.Print "We Are Now in Func3()"`.
4. Вызовите на выполнение форму `frmDebug` и щелкните на управляющей кнопке. Программа Access переведет указатель выполнения в код функции `Func3`, на ту строку, на которой задана точка останова.
5. Откройте окно Стек вызовов (Call Stack), щелкнув на кнопке Стек вызовов (Call Stack) панели инструментов. Если необходимо просмотреть строку кода, в которой вызывается функция `Func2` из функции `Func1`, дважды щелкните на имени функции

`Func1`. Безусловно, Access не переводит точку выполнения в код функции `Func1`, но появляется возможность рассмотреть код этой процедуры. Чтобы возвратиться к следующей строке кода, предназначенной для выполнения, выберите элемент Отладка⇨Отобразить следующий оператор (Debug⇨Show Next Statement).

6. Нажмите клавишу <F5>, после чего остальная часть кода будет выполнена.

Работа с окном Локальные (Locals)

Окно Локальные (Locals) позволяет просматривать все переменные в текущем кадре стека, а также просматривать и изменять их значения. Чтобы получить доступ к области окна Локальные (Locals), щелкните на кнопке Окно локальных значений (Locals Window) панели инструментов или выберите элемент Окно локальных значений (Locals Window) в меню Вид (View). Появятся три столбца: Выражение (Expression), Значение (Value) и Тип (Type). В столбце Выражение (Expression) показаны переменные, определяемые пользователем типы, массивы и другие объекты, видимые в текущей процедуре. В столбце Значение (Value) отображается текущее значение переменной или выражения. Столбец Тип (Type) указывает, данные какого типа содержит переменная. Окна Локальные (Locals) позволяют просматривать переменные, содержащие иерархическую информацию (например, массивы), с помощью кнопки Развернуть/свернуть (Expand/Collapse).

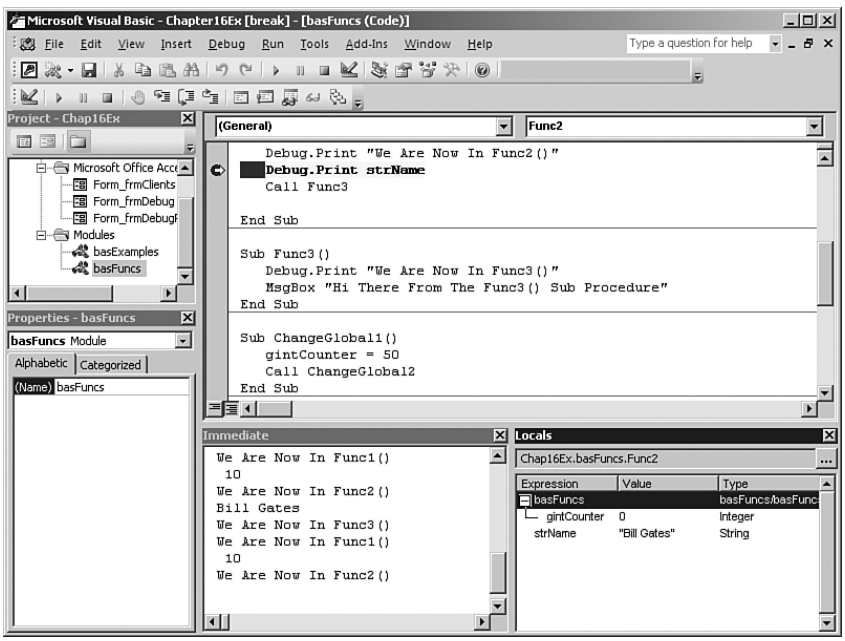

***Рис. 16.12.** Окном Локальные (Locals) можно воспользоваться для просмотра переменных, доступных в подпрограмме*

Информация, содержащаяся в окне Локальные (Locals), изменяется динамически. Программа Access автоматически обновляет ее в ходе выполнения кода, а также при перемещении от процедуры к процедуре. На рис. 16.12 показано, как можно воспользовать-

ся окном Локальные (Locals) для просмотра переменных, имеющихся в подпрограмме `Func2`. Прежде чем приступать к самостоятельному выполнению этого примера, удалите все существующие точки останова. Поместите точку останова в функцию `Func2`, на строку кода `Debug.Print strName`. Щелкните на кнопке `Reset` (Сбросить), если код все еще выполняется, и щелкните на кнопке Начать процесс отладки (Start Debug Process), чтобы выполнить код до точки останова. Щелкните на кнопке Окно локальных значений (Locals Window) панели инструментов Отладка (Debug). Щелкните на знаке "плюс", чтобы просмотреть содержимое общей переменной `gintCounter`.

НА ЗАМЕТКУ

Предусмотрена возможность изменять значения переменных в окне Локальные (Locals), но нельзя модифицировать их имена или типы.

Работа с контрольными выражениями

Иногда окно Просмотр значений переменных (Immediate) не позволяет в полной мере проверить значение выражения или переменной. Но в процессе отладки часто возникает необходимость постоянно контролировать значение некоторого выражения. Для этого можно применить контрольное выражение. После добавления контрольного выражения оно появляется в окне Контрольное значение (Watch). Как будет показано ниже, предусмотрена возможность создавать контрольные значения нескольких типов.

Использование средства Автоматическое отображение значений (Auto Data Tips)

Самым простым и легким способом просмотра значения, содержащегося в переменной, является использование средства Автоматическое отображение значений (Auto Data Tips), которое предназначено для работы с модулями. Это средство становится доступным, только если обработка кода ведется в режиме с применением точек останова. Во время работы в режиме применения точек останова можно переместить указатель мыши на переменную или выражение, значение которого необходимо проверить. Появится всплывающая подсказка с текущим значением. Чтобы ввести в действие средство Автоматическое отображение значений (Auto Data Tips) в редакторе VBE, выберите элемент Инструменты⇨Параметры (Tools⇨Options), откройте вкладку Редактор (Editor) и установите флажок элемента, относящегося к средству Автоматическое отображение значений (Auto Data Tips), который находится под опциями Параметры кода (Code Settings).

Использование быстродействующего контрольного значения

Быстродействующее контрольное значение представляет собой наиболее широко применяемый тип контрольного значения. Чтобы добавить быстродействующее контрольное значение, выделите подсветкой имя переменной или выражение, значение которого необходимо проконтролировать, и щелкните на кнопке Контрольное значение (Quick Watch)

панели инструментов. Откроется диалоговое окно Контрольное значение (Quick Watch), показанное на рис. 16.13. Можно щелкнуть на кнопке Добавить (Add), чтобы добавить выражение в качестве постоянного контрольного значения, или выбрать Отмена (Cancel), чтобы рассмотреть текущее значение, не добавляя его как контрольное значение. После щелчке на кнопке Добавить (Add) откроется окно Контрольные значения (Watches), как показано на рис. 16.14. Это окно рассматривается более подробно в следующем разделе.

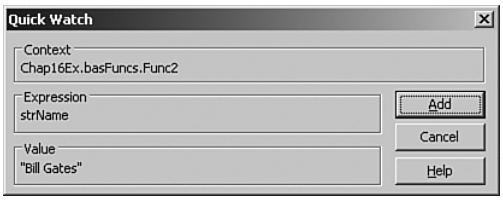

***Рис. 16.13**. Диалоговое окно* Контрольное значение *(Quick Watch) позволяет быстро просматривать значения переменных или добавлять выражения в качестве постоянных контрольных значений*

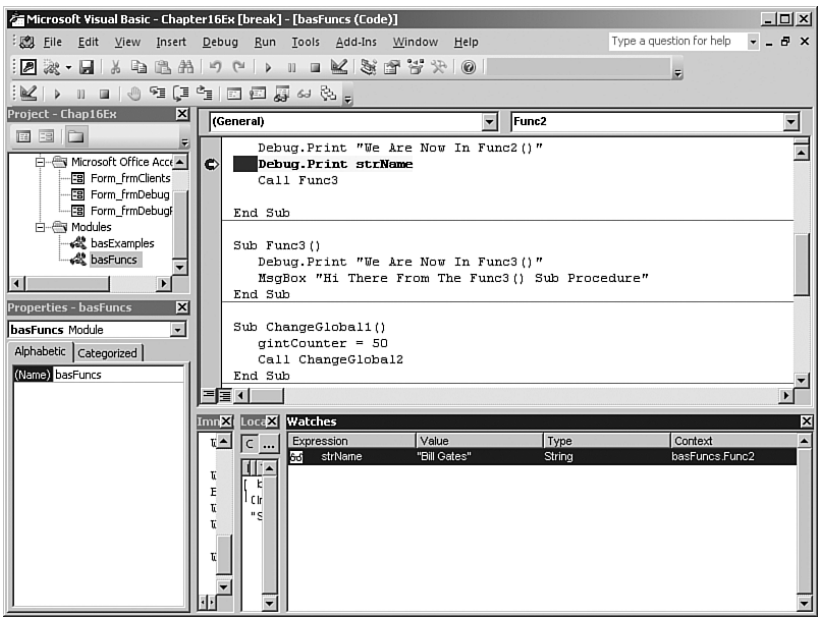

***Рис. 16.14**. Добавление контрольного выражения в окне* Контрольные значения *(Watches)*

Добавление контрольного выражения

Как было описано выше, можно добавить контрольное выражение с помощью быстродействующего контрольного значения. Но добавление контрольного значения таким образом не позволяет полностью реализовать те возможности, которые следуют из характера самого контрольного значения. Если необходимо добиться достижения большей степени управления применительно к контрольному значению, то следует выбрать команду Отладка⇨Добавить контрольное значение (Debug⇨Add Watch). Откроется диалоговое окно Добавить контрольное значение (Add Watch), показанное на рис. 16.15.

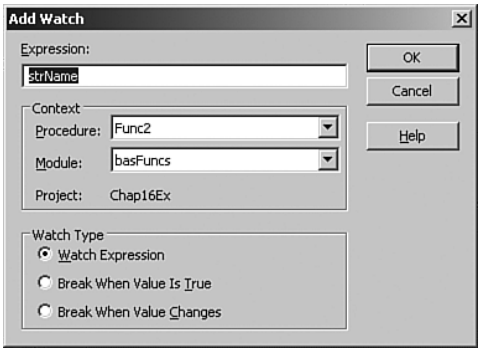

***Рис. 16.15**. Диалоговое окно* Добавить контрольное значение *(Add Watch) позволяет легко определить все необходимые свойства контрольного выражения*

СОВЕТ

Если быстродействующее контрольное значение или обычное контрольное значение добавлено с помощью команды Отладка⇨Добавить контрольное значение (Debug⇨Add Watch), появляется возможность легко настраивать свойства контрольного значения, щелкнув правой кнопкой мыши на контрольном значении в окне Контрольные значения (Watches). После этого достаточно выбрать команду Изменить контрольное значение (Edit Watch).

Один из удобных способов добавления контрольного значения в окне Контрольные значения (Watches) состоит в том, чтобы щелкнуть и перетащить переменную или выражение из программного модуля в окно Контрольные значения (Watches). Программа Access добавляет такое контрольное значение, используя настройки, предусмотренные по умолчанию.

В текстовом поле Выражение (Expression) введите переменную, имя свойства, вызов функции или любое другое допустимое выражение. При этом важно выбрать процедуру и модуль, к которым относится контролируемое выражение. После этого укажите, необходимо ли просто контролировать значение выражения в окне Просмотр значений переменных (Immediate), останавливать выполнение кода после того, как выражение принимает значение `True`, или останавливать выполнение кода после каждого изменения в значении выражения. В следующих разделах рассматриваются два последних варианта.

В приведенном ниже примере показано, как добавить контрольное значение и просмотреть контролируемую переменную при пошаговом выполнении кода. Этот пример может служить иллюстрацией того, как переменная появляется и исчезает из области определения, а ее значение изменяется в ходе выполнения кода.

1. Остановите выполнение кода, если он в данный момент работает, и удалите все заданные в нем точки останова.
2. Щелкните на имени переменной `strName` в коде функции `Func2`.
3. Щелкните правой кнопкой мыши и выберите команду Добавить контрольное значение (Add Watch).

Щелкните на кнопке OK, чтобы подтвердить использование процедуру `Func2` в качестве контекста для переменной и указать, что `basFuncs` является модулем, в котором определена переменная.

1. Задайте точку останова на строке `strName = "Bill Gates"`.
2. Вызовите на выполнение форму `frmDebug` и щелкните на управляющей кнопке. Рассмотрите окно Контрольные значения (Watches) и обратите внимание на то, что значением переменной `strName` является строка с нулевой длиной.
3. Сделайте один шаг в процессе пошагового выполнения и убедитесь в том, что переменная `strName` принимает значение, равное `Bill Gates`.
4. Сделайте еще три шага в процессе пошагового выполнения. Обратите внимание на то, что произошел переход к процедуре `Func3`, но, несмотря на это, переменная `strName` все еще имеет значение `Bill Gates`. Причина этого состоит в том, что переменная все еще находится в памяти в контексте функции `basFuncs.Func2`.
5. Сделайте еще четыре шага, после чего произойдет переход к инструкции `End Sub`, которая находится в коде функции `Func2`. Переменная `strName` все еще находится в том же контексте.
6. Сделайте еще один шаг. Переменная `strName` наконец выходит из рассматриваемого контекста, поскольку выполнение функции `Func2` завершается.

Редактирование контрольных выражений

После добавления контрольного значения может потребоваться откорректировать свойство контрольного значения или полностью его удалить. Для изменения или удаления контрольных выражений используется диалоговое окно Изменить контрольное значение (Edit Watch).

1. Откройте окно Контрольные значения (Watches).
2. Выберите выражение, которое необходимо изменить.
3. Выберите элемент Отладка⇨Изменить контрольное значение (Debug⇨Edit Watch) или щелкните правой кнопкой мыши и выберите команду Изменить контрольное значение (Edit Watch). Откроется диалоговое окно, которое показано на рис. 16.16.
4. Внесите изменения в контрольное значение или щелкните на кнопке Удалить (Delete), чтобы его удалить.

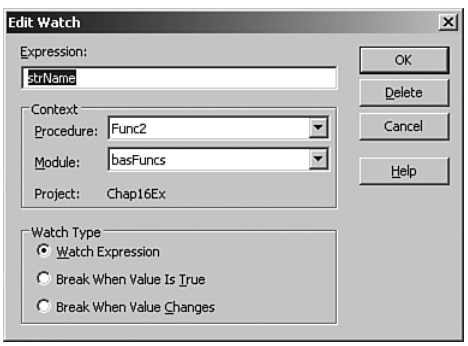

Рис. 16.16. *Диалоговое окно* Изменить контрольное значение *(Edit Watch) можно использовать для изменения свойств контрольного значения после его добавления*

Останов выполнения кода после того, как некоторое выражение принимает значение `True`

Одна из перспективных областей применения контрольных выражений состоит в том, что с их помощью можно останавливать выполнение кода каждый раз, когда выражение становится равным `True`. Благодаря этому, например, можно обеспечить останов выполнения кода каждый раз, когда некоторая переменная с атрибутом `Public` достигает определенного значения. Необходимость в этом может возникнуть, если в какой-то момент изменяется значение переменной `Public` или `Private` и необходимо узнать, при каких обстоятельствах возникает такое изменение. Рассмотрим следующий код, находящийся в модуле `basFuncs` базы данных `CHAP16EX.ACCDB`:

```
Sub ChangeGlobal1()
   gintCounter = 50
   Call ChangeGlobal2
End Sub
Sub ChangeGlobal2()
   gintCounter = gintCounter + 10
   Call ChangeGlobal3
End Sub
Sub ChangeGlobal3()
   Dim intCounter As Integer
   For intCounter = 1 To 10
      gintCounter = gintCounter + intCounter
   Next intCounter
End Sub
```

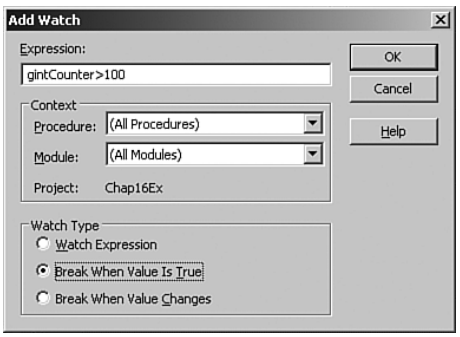

Рис. 16.17. *Применение рассматриваемого контрольного значения, которое приводит к тому, что выполнение кода останавливается каждый раз, когда выражение принимает значение* `True`

Можно обнаружить, что переменная `gintCounter` каким-то образом достигает значения, большего 100, но по какой причине это происходит, не совсем понятно. Чтобы найти решение этой задачи, добавьте контрольное значение, которое показано на рис. 16.17. Обратите внимание на то, что выражением, с помощью которого выполняется проверка, является `gintCounter > 100`. Таким образом, задана точка останова, в которой выполнение кода останавливается каждый раз, когда рассматриваемое выражение принимает значение `True`. Для проверки кода введите `ChangeGlobal1` в окне Просмотр значений переменных (Immediate) и нажмите клавишу <Ввод> (Enter). Выполнение кода должно быть остановлено в процедуре `ChangeGlobal3`, указывая на то, что переменная принимает недопустимое значение именно в этой процедуре.

Останов выполнения кода после изменения значения выражения

Иногда возникает необходимость останавливать выполнение кода не тогда, когда выражение принимает значение `True`, а после каждого изменения значения выражения. В этом состоит превосходный способ определить, в каком месте загадочным образом происходит изменение значения переменной. Как и останов выполнения кода после принятия значения `True`, вариант с остановом при изменении значения великолепно подходит для отслеживания нарушений в работе, связанных с использованием переменных `Private` и `Public`. Обратите внимание на то, как задано контрольное значение на рис. 16.18. Оно находится в контексте всех процедур, входящих в состав всех модулей. Это контрольное значение установлено так, что выполнение кода останавливается при каждом изменении значения переменной `gintCounter`. После вызова процедуры `ChangeGlobal1` можно обнаружить, что выполнение кода этой процедуры прекращается сразу же после того, как в коде задается значение `gintCounter`, равное 50. Если для продолжения выполнения будет нажата клавиша <F5>, то работа кода остановится, но на этот раз в процедуре `ChangeGlobal2`, сразу же после того, как в ней значение `gintCounter` будет увеличено на 10. Иными словами, выполнение кода прекращается после каждого изменения значения `gintCounter`.

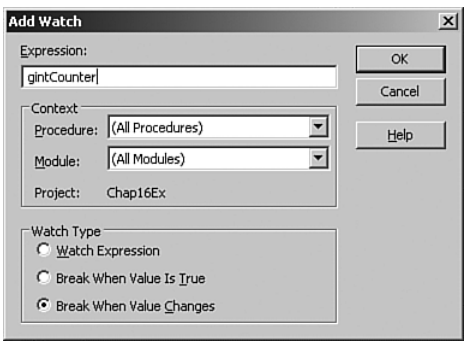

***Рис. 16.18.** Контрольное значение, которое прекращает выполнение кода после каждого изменения значения выражения*

Продолжение работы после возникновения ошибки этапа прогона

В ходе проверки программы часто обнаруживаются ошибки этапа прогона, которые можно очень легко исправить. При возникновении ошибки этапа прогона открывается диалоговое окно, подобное приведенному на рис. 16.19.

После щелчка на элементе Отладка (Debug) программа Access открывает окно Программный код (Code) с указанием той строки кода, в которой возникла ошибка. После устранения причины этой ошибки щелкните на кнопке Продолжить (Continue) панели инструментов или выберите команду Выполнить⇨Продолжить (Run⇨Continue).

На рис. 16.20 показано, как отображается ошибка деления на нуль, например, после того как пользователь щелкнул на элементе Отладка (Debug) при открытом диалоговом

окне Ошибка выполнения (Runtime Error). В окне Локальные (Locals), приведенном на этом рисунке, показано, что программист задал значение `int2`, равное 20. Теперь выполнение кода может быть продолжено без ошибок.

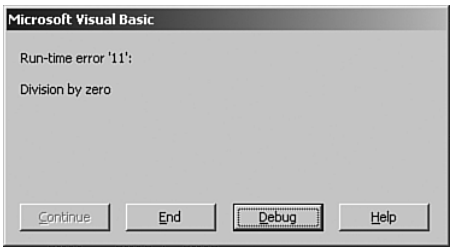

***Рис. 16.19**. Диалоговым окном* Ошибка выполнения *(Runtime Error) можно воспользоваться для исправления ошибок этапа прогона*

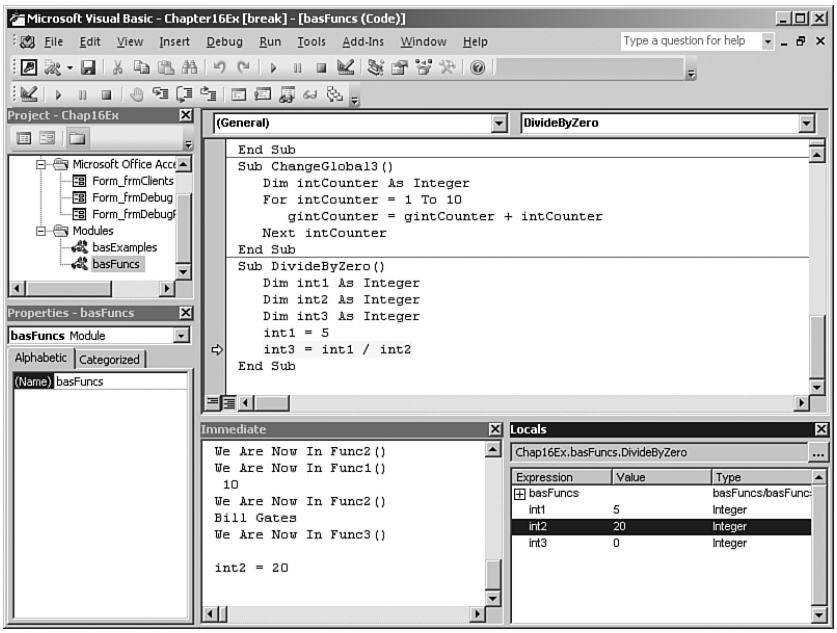

***Рис. 16.20**. Это окно показывает, как действует режим отладки после возникновения ошибки деления на нуль*

После того как появляется ошибка в коде VBA, часто отображается сообщение о том, что в данный момент весь выполняемый код может быть переустановлен в исходное состояние. Но следует учитывать, что если выбран вариант с переустановкой кода, то все переменные (включая переменные с атрибутом `Public` или `Static`) теряют свои значения. Можно также щелкнуть на кнопке Reset (Сбросить) панели инструментов. На этом этапе должно быть принято решение о том, следует ли продолжить работу с использованием уже установленных значений переменных или переустановить значения переменных и только после этого продолжить выполнение приложения.

НА ЗАМЕТКУ

В процессе отладки происходит взаимодействие средств обработки ошибок VBA и отладчика, а для настройки такого взаимодействия служит вкладка Общее (General) диалогового окна Параметры (Options). В главе 17 рассматриваются доступные при этом опции.

Нюансы, связанные с использованием окна Просмотр значений переменных (Immediate)

Безусловно, отладчик Access превосходно справляется со своими задачами, но в процессе отладки можно столкнуться с целым рядом потенциальных проблем, включая описанные ниже.

- В процессе отладки может быть прервано выполнение кода, что особенно заметно во время работы с формами. Если такое происходит, то вместо этого следует поместить в код инструкции `Debug.Print` и проследить за происходящем после завершения работы кода.
- Практически по тем же причинам, по которым возникает предыдущая проблема, является сложной отладка приложений, в коде которых применяются обработчики событий `LostFocus` и `GotFocus`. Дело в том, что событие `LostFocus` элемента управления активизируется не только в процессе работы приложения, но и, например, при переходе в окно редактора VBE. А возврат к форме вынуждает программу Access активизировать событие `GotFocus` элемента управления. И в этом случае наилучшее решение состоит в использовании инструкций `Debug.Print`. Можно также рассмотреть возможность записи информации в журнал регистрации ошибок для ее анализа после выполнения кода.
- В процессе отладки могут возникать нарушения при выполнении кода, в котором используются свойства `Screen.ActiveForm` и `Screen.ActiveControl`. Дело в том, что в ходе работы с редактором VBE какие-либо активные формы и активные элементы управления отсутствуют. Чтобы устранить указанную проблему, следует избегать использования свойств `Screen.ActiveForm` и `Screen.ActiveControl` в коде при любой возможности.
- Наконец, необходимо учитывать, что причиной возникновения нарушений в работе может стать сброс кода. Если в процессе отладки происходили изменения настроек среды, то после сброса кода остаются неизменными те настройки среды, которые изменились в ходе отладки кода. Поэтому, если выполнение приложения после устранения ошибки будет продолжено без восстановления первоначальных настроек среды, может возникнуть целый ряд других проблем. В связи с этим рекомендуется подготовить специальную сервисную процедуру, предназначенную для восстановления применяемой среды.

Применение утверждений

Утверждения применяются для того, чтобы в коде был вызван отладчик при обнаружении определенной ситуации в ходе работы пользователя. В качестве примера можно привести следующий код, который находится в модуле `basExamples`:

```
Sub Assertion()
    Dim intAge As Integer
    intAge = InputBox("Please Enter Your Age")
    Debug.Assert (intAge >= 0)
    MsgBox "You are " & intAge
End Sub
```

В этом примере задается значение переменной `intAge`, равное значению, введенному в окне ввода. Инструкция `Debug.Assert` служит в качестве "утверждения", согласно которому введенное значение должно быть больше или равно нулю. Если это утверждение является правильным, то выполнение кода продолжается в соответствии с ожиданиями. Если же утверждение неверно, в коде вызывается отладчик.

Рекомендуется всегда включать в код комментарии, поясняющие, по каким причинам может завершиться неудачей проверка того или иного утверждения. Это позволяет упростить процесс анализа нестандартной ситуации после ее возникновения. Кроме того, важно учитывать, что не следует развертывать приложение с оставшимися в нем инструкциями `Debug.Assert`, поскольку в процессе эксплуатации приложения конечными пользователями нарушение условий любого утверждения приведет к тому, что на экране пользователя без каких-либо предупреждений откроется окно отладчика, поэтому неизбежным станет вызов представителей службы поддержки!

Рекомендации по отладке

Ниже приведены рекомендации, позволяющие значительно упростить процесс отладки приложений.

- Прежде чем приступать к отладке, выясните, в чем состоит проблема. Убедитесь в том, что получена вся необходимая информация от пользователя, касающаяся того, при каких условиях возникла проблема. Не имея этой жизненно важной информации, придется потратить очень много времени, осуществляя попытку вновь вызвать проблемную ситуацию, вместо поиска ее решения.
- Вносите изменения одновременно только в отдельные строки кода. Автору часто приходилось быть свидетелем того, как безрассудно разработчики пытаются сразу же откорректировать целый ряд строк кода. Но вместо устранения проблемы, которую они первоначально пытались решить, появляется множество дополнительных проблем.
- Обсудите обнаруженную проблему с другими разработчиками. Иногда достаточно найти словесную формулировку проблемы, чтобы понять, в чем состоит суть возникшего затруднения. Если, сформулировав суть проблемы, вы не сможете сами найти ответ, то, по крайней мере, будете иметь возможность связаться с другим лицом, которое подскажет нужное решение.
- Если все усилия оказываются безрезультатными, сделайте перерыв. Автору часто приходилось засиживаться почти до рассвета в попытках решить проблему. Но после того как все попытки оканчиваются неудачей, приходится признавать себя побежденным и отправляться на отдых. А самое удивительное то, что часто удается найти решение "неразрешимой" задачи на следующее утро, после принятия душа!

Практические примеры: отладка реальных приложений

Методы, описанные в этой главе, могут помочь в решении практически всех проблем, возникающих перед разработчиком в процессе создания его собственных приложений. Но вначале рекомендуется воспользоваться отладчиком для пошагового выполнения и более глубокого изучения процесса отладки на примере процедур, приведенных в рассматриваемом образце базы данных.

Резюме

Многие утверждают, что если бы программирование действительно представляло собой точную науку, а не искусство, то не было бы причин использовать отладчик. Но на самом деле в ходе разработки приложений встречаются чрезвычайно сложные ситуации, поэтому программист обязан в совершенстве усвоить весь процесс использования отладчика. К счастью, редактор VBE, входящий в состав Access 2007, предоставляет превосходные инструменты, способные помочь в процессе отладки.

В этой главе прежде всего были описаны методы, способствующие снижению с самого начала вероятности появления ошибок в приложении. Затем было показано, как использовать окно Просмотр значений переменных (Immediate) для проверки и модификации значений переменных и свойств. В ней также приведено описание методов использования контрольных значений и точек останова, кроме того, показаны способы просмотра стека вызовов. Благодаря использованию всех описанных методов можно значительно упростить процесс проверки и отладки приложений.