DELPHI

НАРОДНЫЕ СОВЕТЫ

АНДРЕЙ ШКРЫЛЬ

ТРЮКИ И ХИТРОСТИ ИСПОЛЬЗОВАНИЯ СТАНДАРТНЫХ КОМПОНЕНТОВ

ГОТОВЫЕ РЕШЕНИЯ ДЛЯ ПОВСЕДНЕВНЫХ ЗАДАЧ

РАБОТА С СУБД: ORACLE, INTERBASE/FIREBIRD, MYSQL, MS SQL SERVER

ВЗАИМОДЕЙСТВИЕ С МУЛЬТИМЕДИА

РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ ИНТЕРНЕТА

РАБОТА С ОБОРУДОВАНИЕМ

ЗАЩИТА ПРОГРАММ И ШИФРОВАНИЕ ДАННЫХ

ПРОГРАММИРОВАНИЕ В .NET

ИСПОЛЬЗОВАНИЕ ДОПОЛНИТЕЛЬНЫХ КОМПОНЕНТОВ

bhv

+ CD

Андрей Шкрыль

DELPHI
НАРОДНЫЕ СОВЕТЫ

Санкт-Петербург
«БХВ-Петербург»
2007

УДК 681.3.068+800.92Delphi
ББК 32.973.26-018.1
Ш66

Шкрыль А. А.

Ш66 Delphi. Народные советы. — СПб.: БХВ-Петербург, 2007. — 400 с.: ил. + CD-ROM

ISBN 978-5-9775-0047-0

Рассмотрен широкий круг практических вопросов по программированию в Delphi: трюки и хитрости использования стандартных компонентов, готовые решения для повседневных задач, работа с СУБД (ORACLE, Interbase/Firebird, MySQL, MS SQL Server), взаимодействие с мультимедиа, разработка приложений для Интернета, работа с оборудованием, защита программ и шифрование данных, программирование в .NET, использование дополнительных компонентов и многое другое. На компакт-диске содержатся исходные коды программ, рассмотренные в книге, а также дополнительные приложения и компоненты.

Для программистов

УДК 681.3.068+800.92Delphi
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.11.06.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 32,25.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04
от 11.11.2004 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

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

ISBN 978-5-9775-0047-0

Оглавление

Введение

"Эта уже моя третья книга", — подумал я, наливая очередную чашку крепкого кофе и погружаясь в раздумья...

Удивительные и порой очень приятные сюрпризы преподносит нам жизнь. Даже во сне я не мог представить себе, что стану писателем. В детстве я недолюбливал русский язык, а сочинения по литературе мы писали всей семьей, т. к. излагать мысли красиво не получалось. Отматывая ленту жизни немного назад, вспоминаю еще один непростой этап в жизни, который был связан со сменой работы. Тогда я решил, что программирование вообще не для меня и нужно пробовать себя в чем-то другом. И вы никогда не догадаетесь, что я сделал: я собрал все свои компьютерные книги и журналы и отнес их своему очень хорошему другу, со словами: "Они мне больше не нужны, но я думаю, тебе пригодятся". Прошла пара недель, и я опять стоял на пороге квартиры моего хорошего друга, правда, уже с просьбой одолжить часть подаренной ему мною литературы. Стоит отдать ему должное, к просьбе он отнесся с пониманием.

Вы держите в руках книгу народных советов по Delphi — книгу, которая поможет вам взглянуть на программирование другими глазами. Так как основная ее задача — показать, что в жизни нет невыполнимых задач, и что вам по силам решение абсолютно каждой из них, главное запастись немного терпением, ведь это одно из главных качеств ИТ-специалиста. Книга построена по принципу "вопрос — ответ", и именно поэтому она будет полезна широкому кругу читателей. Даже если вы считаете себя гуру в программировании, я абсолютно уверен, что вы найдете в книге новые и интересные для себя вопросы. Программное создание компонентов, улучшение их свойств, работа с графикой, звуком и видео, разработка приложений для мобильных устройств и операционной системы Linux, программирование для .NET, организация индикации процесса выполнения SQL-запроса, оперативное обновление данных в клиент-серверных приложений, работа с Интернетом, распознание графических изображений, говорящий компьютер, работа с формулами, проверка орфографии и использование дополнительных разработок — это всего лишь малая часть вопросов, рассмотренных в данной книге, т. к. перечислять их можно достаточно долго.

Я постарался сделать процесс прочтения максимально комфортным для читателя и не только разместил исходные коды приложений, рассматриваемых в книге, на компакт-диске, но также и большинство используемых библиотек, программ и дополнительных компонентов. Таким образом, вам потребуются для работы над материалом операционная система, Delphi и CD-привод на вашем ПК. Но это еще не все, при запуске диска вас ждет приятный сюрприз — удобная оболочка, которая позволит быстро найти нужную информацию.

Приятного прочтения.

Благодарности

Хочу сказать спасибо своим родителям, которые привили мне любовь к получению новых знаний и упорство в достижении целей, а также поддерживали меня в сложных жизненных ситуациях. Хочу поблагодарить брата Антона и сестру Ольгу за их веру в меня.

Выражаю большую признательность своей жене Наташе за ее поддержку, помощь в поиске новых идей, а также за ее терпение.

Огромная благодарность замечательным и талантливым людям, профессионалам своего дела: Шишигину Игорю и Фленову Михаилу за их моральную поддержку.

Хочу поблагодарить Иннокентия Козлова (aka Innok), который помогал мне своими советами и делился опытом.

И, конечно, огромная признательность коллективу издательства "БХВ-Петербург", благодаря которому вы держите в руках эту книгу.

Глава 1

IDE

Прежде чем приступить к программированию на Delphi, необходимо освоить работу с интегрированной средой разработки — IDE (Integrated Development Environment). Другими словами, научиться дружить с оболочкой, в которой вы будете разрабатывать свою программу. Тем более, если вы стремитесь стать мастером своего дела, тогда без профессионального знания инструментов и сервисных функций, предоставляемых IDE, вам просто не обойтись.

Итак, вы запустили Delphi. После того как произойдет загрузка, вы попадете непосредственно в среду разработки. Теперь у вас есть все, что необходимо для освоения материала данной главы.

Вопрос 1. Я настраиваю расположение элементов IDE Delphi, располагаю инспектор объектов и другие панели по своему удобству, но после перезапуска Delphi это расположение не сохраняется. Как исправить данную ситуацию?

Ответ. Вверху, под главным меню есть выпадающий список (рис. 1.1).

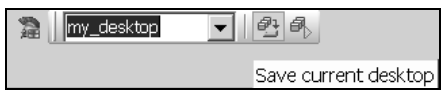

Рис. 1.1. Сохранение настройки своего рабочего стола

Настройте IDE под себя, а потом введите название вашего рабочего стола (desktop) в выпадающий список и нажмите кнопку **Save current desktop** (первая слева от выпадающего списка). Если вас посетит ностальгия, и вы захотите возвратиться к стандартному расположению панелей, просто выберите другой вариант рабочего стола из выпадающего списка.

Вопрос 2. Какие полезные комбинации клавиш, предназначенные для быстрого вызова сервисных функций, существуют в Delphi?

Ответ. Список представлен в табл. 1.1. Если вы разместите его рядом со своим монитором, то не придется полностью его запоминать и при необходимости можно достаточно быстро обратиться к той или иной функции.

Таблица 1.1. *Комбинации клавиш быстрого доступа*

Клавиша или комбинация клавиш	**Описание**
<F9>	Компиляция и запуск приложения
<Ctrl>+<F9>	Компиляция без запуска, например, очень актуально при создании DLL
<F4>	Выполнить программу до положения курсора
<F5>	Поставить точку останова (break point)
<F7>	Трассировка с заходом в процедуры
<F8>	Трассировка без захода в процедуры
<Ctrl>+<F2>	Прерывание выполнения программы. Попробуйте сначала нажать клавишу <F9>, после запуска программы перейдите в режим просмотра кода (нажмите клавишу <F12>), а теперь — <Ctrl>+<F2>, после чего выполнение программы будет приостановлено
<F12>	Переключение между формой и кодом программы
<F11>	Показать инспектор объектов или сделать его активным, если он уже отображен на экране
<Ctrl>+<F12>	Показать список всех модулей проекта
<Shift>+<F12>	Показать список всех форм проекта
<Shift>+<F11>	Добавить модуль в проект
<Ctrl>+<F11>	Открыть проект
<Ctrl>+<Shift>+<F11>	Вызов окна **Project Options**
<Ctrl>+<S>	Сохранение текущего модуля
<Shift>+<Ctrl>+<S>	Сохранение всего проекта
<Ctrl>+<O>, <O>	Вставить настройки компилятора в начало файла

Таблица 1.1 *(окончание)*

Клавиша или комбинация клавиш	Описание
<Ctrl>+<K>, <F>	Перевод слова в верхний регистр (слово должно быть предварительно выделено)
<Ctrl>+<K>, <E>	Перевод слова в нижний регистр (слово должно быть предварительно выделено)

Теперь откройте пустой проект, перейдите в его модуль, найдите строчку кода:

```
type
  TForm1 = class(TForm)
```

Удерживая клавишу <Ctrl>, щелкните левой кнопкой мыши на `TForm`. В результате загрузится модуль `Forms`, причем курсор будет находиться на определении класса `TForm`. Очень удобно, не правда ли? Таким образом, вы можете просматривать определение переменных, классов, функций и т. д., даже не зная, в каком модуле они объявлены — Delphi поможет вам в этом.

Продолжаем эксперименты. Разместите на форме кнопку, дважды щелкните на ней левой кнопкой мыши. Теперь в обработчике введите `ShowMe` и нажмите комбинацию клавиш <Ctrl>+<Пробел>. Вы увидите подсказку, как на рис. 1.2.

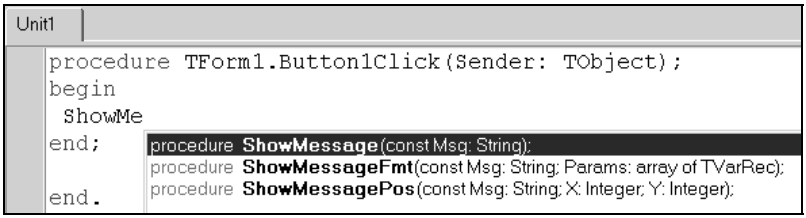

Рис. 1.2. Delphi помогает при вводе кода, позволяя после первых введенных символов увидеть подсказку

После того как процедура `ShowMessage` введена, перейдите на новую строку и введите `if`, далее нажмите комбинацию клавиш <Ctrl>+<J>. Вы увидите список шаблонов кода (рис. 1.3).

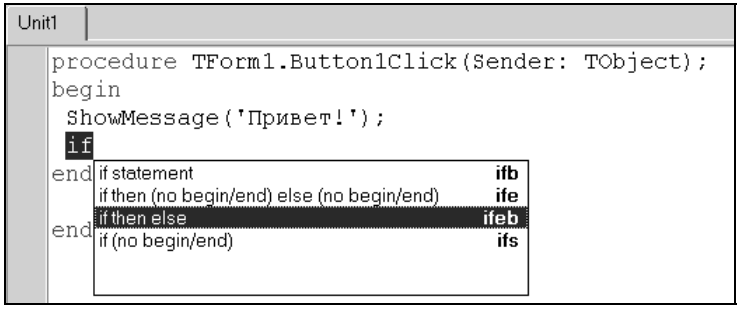

Рис. 1.3. Delphi помогает при вводе кода, позволяя использовать в своей работе уже готовые шаблоны кода

Для того чтобы отредактировать уже существующий шаблон или добавить свой, необходимо выбрать пункт меню **Tools | Editor Options**. В появившемся окне **Editor Properties** перейдите на вкладку **Source Options** и нажмите кнопку **Edit Code Templates** (в Delphi 2006 надо выполнить команду **View | Templates**).

Переведите курсор на строчку кода с процедурой `ShowMessage`. Далее нажмите комбинацию клавиш <Ctrl>+<Shift>+<*цифра*>. Вы увидите, что напротив текущей строки будет установлена закладка. Теперь нажмите комбинацию клавиш <Ctrl>+<Home> — вы окажетесь в начале модуля. А теперь <Ctrl>+<*цифра*> (нажимайте ту клавишу с цифрой, которую вы нажимали первый раз) — вы вернетесь на ту строку, где была установлена закладка. В нашем небольшом модуле это не кажется уж таким большим удобством — но при коде в 1000 и больше строк, когда вы будете отлаживать или пытаться понять логику работы чужой программы, поверьте мне, вы не раз воспользуетесь данным способом.

Теперь перейдите в модуль `Forms`. Помните, мы уже открывали его чуть раньше. Найдите любую процедуру. Например, я выбрал:

```
procedure SetActive(Value: Boolean);
```

Установите курсор в любое место объявления данной процедуры и нажмите комбинацию клавиш <Ctrl>+<Пробел>+<↓>. Ву а ля, вы попадаете в реализацию процедуры, удобно, не правда ли? Теперь вернемся назад к месту объявления процедуры. Для этого достаточно всего лишь нажать комбинацию клавиш <Ctrl>+<Пробел>+<↑>.

Вопрос 3. Я программировал в среде Visual Studio. Когда я перешел на Delphi, то не очень понравилась цветовая гамма листинга кода: в Delphi 2006

этот недостаток исправлен, но в Delphi 7 она достаточно однообразна. Как можно повлиять на данную ситуацию?

Ответ. В Delphi 7 для этого достаточно выбрать пункт меню **Tools | Editor options**, в появившемся окне **Editor properties** перейдите на вкладку **Color** и в выпадающем списке **Color Speed Setting** выберите схему **Visual Studio (TM)**. На мой взгляд, это одна из самых удобных цветовых гамм. Также вы здесь можете настроить цветовую схему под свой вкус. Если вы работаете в Delphi 2006, то выберите пункт меню **Tools | Options**, далее в появившемся окне **Options** выберите раздел **Editor Options | Color**.

Вопрос 4. Как можно быстро закомментировать некоторый фрагмент кода?

Ответ. Такая возможность появилась, начиная с Delphi 2005. Выделите фрагмент кода, который вы хотите закомментировать, и воспользуйтесь сочетанием клавиш <Ctrl>+</>. Повторное использование этого сочетания приведет к тому, что комментарии будут убраны (рис. 1.4).

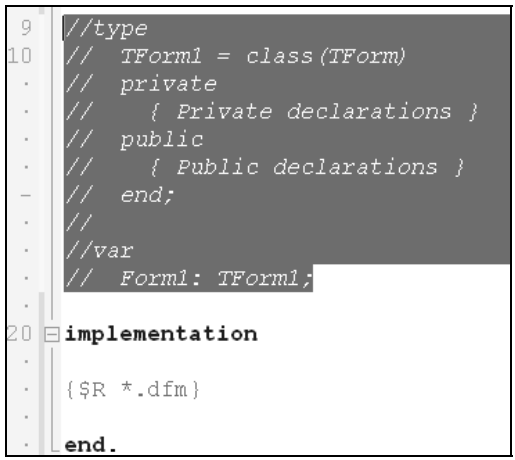

Рис. 1.4. Быстрое комментирование выделенного фрагмента кода

Вопрос 5. Начиная с Delphi 2005, появилась возможность сворачивать код, для этого предусмотрены знаки [+] и [−] в левой части редактора. Как мне получить такую же возможность для своего участка кода?

Ответ. Для этого предусмотрены специальные директивы `$REGION` и `$ENDRegion`. Они задают фрагмент кода (регион), который можно будет сво-

рачивать. На рис. 1.5 и рис. 1.6 вы можете видеть использование данных директив. Стоит отметить, что данным участкам кода можно давать имена, для этого достаточно ввести его после директивы `$REGION`.

```
procedure TForm1.Button1Click(Sender: TObject);
var my_stream:TMemoryStream;
begin
  my_stream:=TMemoryStream.Create;
  try
    {$REGION 'My_Code'}
    IdHTTP1.Get(Edit1.text, my_stream);
    my_stream.Position := 0;
    my_stream.SaveToFile('c:\file_from_network.html');
    {$ENDRegion}
  finally
    my_stream.Free;
  end;
end;
```

Рис. 1.5. Использование директив `$REGION` и `$ENDRegion`

```
procedure TForm1.Button1Click(Sender: TObject);
var my_stream:TMemoryStream;
begin
  my_stream:=TMemoryStream.Create;
  try
    My_Code
  finally
    my_stream.Free;
  end;
end;
```

Рис. 1.6. Сворачивание участка кода

Вопрос 6. Для чего предназначено средство **To-Do List**, которое вызывается пунктом меню **View | To-Do List**?

Ответ. Данное средство незаменимо, когда над кодом работает несколько человек, а коллективное средство разработки по какой-то причине не используется. Также оно удобно и в случае, когда вы реализуете одновременно несколько проектов, т. к. помогает вставлять заметки о том, что надо сделать или уже сделано. Причем можно быстро обращаться к тем участкам кода, для которых были сделаны соответствующие заметки.

Предлагаю активизировать данное средство и сразу разместить его в нижней части редактора кода (рис. 1.7), после чего сохранить рабочий стол (desktop), например, под именем My_Debuger.

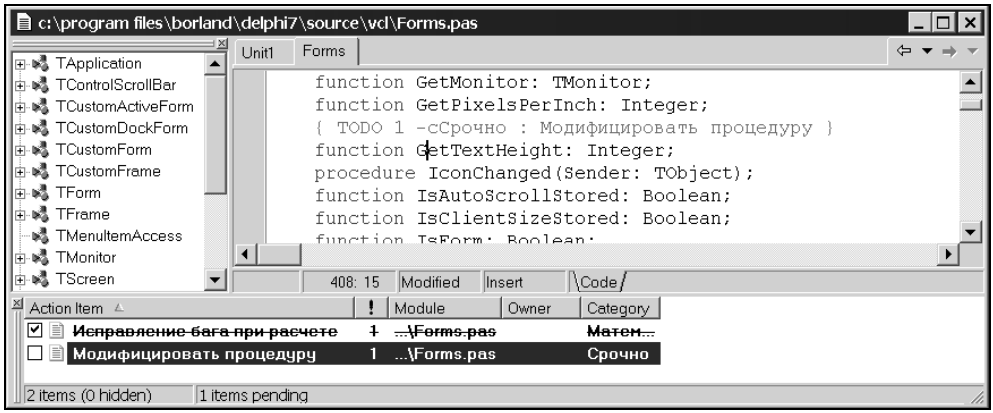

Рис. 1.7. Удобно размещаем **To-Do List**

Теперь выделите любую строчку вашего кода и щелкните правой кнопкой мыши. В появившемся выпадающем списке выберите пункт **Add To-Do Item...**, появится окно **Add To-Do Item**. Введите в нем заметку к заданию, название категории и при необходимости укажите приоритет и владельца, далее нажмите кнопку **OK**. После чего в вашем To-Do-списке добавится новая запись. Теперь добавьте точно таким же образом еще одну строчку, желательно в каком-нибудь отдаленном месте кода, и если после этого вы произведете двойной щелчок левой кнопкой мыши на первой записи, то окажетесь именно в той строчке кода, к которой привязана данная заметка.

Вопрос 7. Как сделать так, чтобы при возникновении исключений Delphi не прерывал выполнение программы?

Ответ. Сделать это достаточно легко, только не рекомендую налегать на данный способ, иначе можно столкнуться с моментом, когда программа работает совершенно неправильно, но понять причину не удается. Выберите пункт меню **Tools | Debugger options** в появившемся окне **Debugger Options** перейдите на вкладку **Language Exceptions** и сбросьте флажок **Stop on Delphi Exceptions** (если вы используете Delphi 2006, то необходимо выбрать пункт меню **Tools | Options**, далее перейти в раздел **Debugger Option | Language Exceptions** и сбросить флажок **Notify on language exceptions**).

Теперь напишем простое приложение. На форме разместите компонент `TEdit` и кнопку. Для кнопки создайте следующий обработчик:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i:integer;
begin
  i:=StrToInt(Edit1.Text);
end;
```

Как видите, мы заранее закладываем ошибку в программу. Текст, по умолчанию располагающийся в `TEdit` — `Edit1`, не может быть конвертирован в тип данных `integer`. Теперь можете попробовать выполнить программу и нажать кнопку при включенном и выключенном флажке **Stop on Delphi Exceptions**.

Вопрос 8. Как отключить предупреждения Delphi, которые появляются, например, когда переменная объявлена, но не используется (рис. 1.8)?

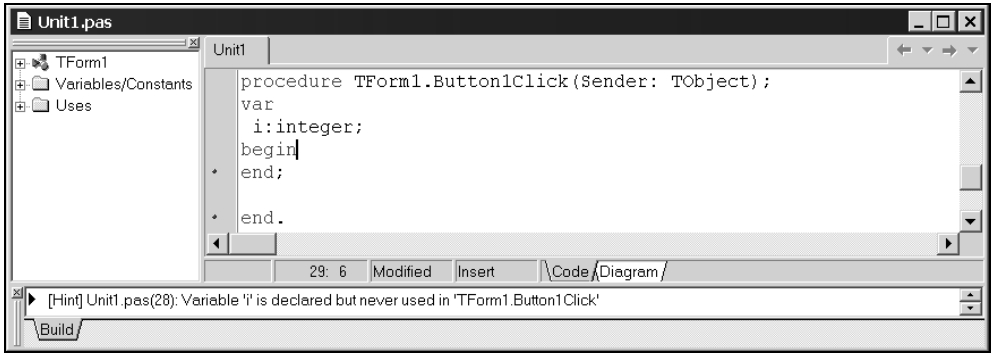

Рис. 1.8. Сообщение компилятора Delphi (в данном случае говорится о том, что в программе есть неиспользуемая переменная)

Ответ. Выберите пункт меню **Project | Options** и перейдите на вкладку **Compiler messages**, далее сбросьте нужные вам флажки.

Если необходимо, чтобы сообщения компилятора были отключены только для некоторого фрагмента кода, тогда поместите его между директивами `{$HINTS OFF}` и `{$HINTS ON}`, а при необходимости также можно использовать директивы `{$WARNINGS OFF}` и `{$WARNINGS ON}`. Например, в случае

отключения сообщения о неиспользуемой переменной код будет выглядеть так:

```
{$HINTS OFF}
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
begin
end;
{$HINTS ON}
```

> ***Замечание***
>
> В Delphi 2006 при запуске приложения по умолчанию сообщения компилятора не видны. Чтобы их отобразить, щелкните правой кнопкой мыши в любом месте формы или модуля и выберите пункт **Message View**.

Вопрос 9. Как выводить собственные сообщения при компиляции программы?

Ответ. Для этого достаточно воспользоваться директивой `$MESSAGE`, которая имеет следующий формат:

```
{$MESSAGE HINT|WARN|ERROR|FATAL 'текст сообщения'}
```

Примеры использования:

```
{$MESSAGE 'Начата компиляция программы'}
{$MESSAGE ERROR 'Произошла ошибка, дальнейшая компиляция невозможна!'}
```

Вопрос 10. У меня в программе много функций и процедур собственной разработки, причем часто одна вызывает другую, а та может вызывать еще несколько. Каким образом можно удобно работать с ними в режиме отладки, т. к. все имена помнить просто невозможно?

Ответ. Воспользуйтесь окном **Call Stack**. Для его отображения выберите пункт меню **View | Debug Windows | Call Stack**.

Рассмотрим небольшой пример. Создайте приложение, разместите на форме кнопку, для которой напишите следующий обработчик:

```
procedure TForm1.Button1Click(Sender: TObject);
// Процедура A
procedure A;
begin
end;
```

```
// Процедура B
procedure B;
begin
  // Вызываем процедуру A
  A;
end;

// Основная программа - обработчик OnClick для кнопки
begin
  // Вызываем процедуру B
  B;
end;
```

Теперь поставьте точку останова на вызов процедуры `B`, после чего выберите пункт меню **Run | Run**. Далее вызовите окно **Call Stack** и нажмите кнопку, размещенную на форме. Теперь воспользуйтесь режимом отладки приложения с помощью клавиши <F7>, параллельно обращая внимание, как меняется информация в окне **Call Stack** в зависимости от того, в какой процедуре вы находитесь (рис. 1.9).

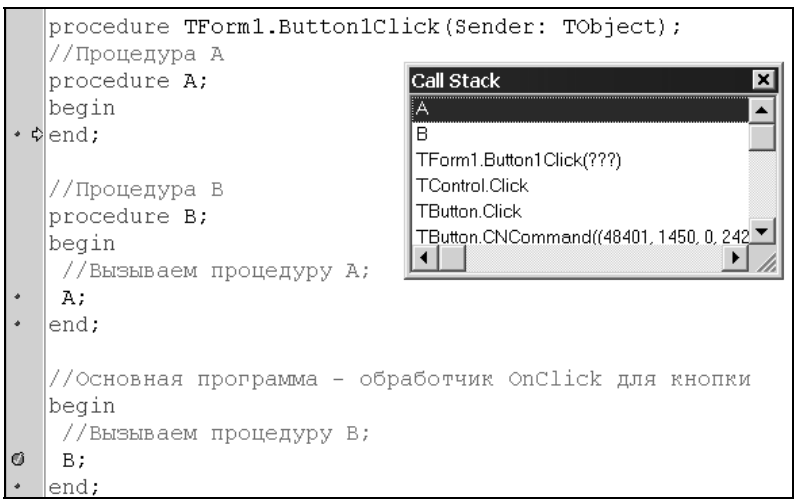

Рис. 1.9. Отладка программы с использованием окна **Call Stack**

Вопрос 11. Как пользоваться окном **CPU**, которое появляется при отладке программы?

Ответ. Рассмотрим небольшой пример. Создайте новое приложение, теперь разместите на форме кнопку, для которой напишите следующий обработчик:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Caption:='Моя кнопка';
end;
```

Теперь поставьте точку останова на данную строчку кода, после чего запустите приложение, нажмите кнопку в форме. Далее воспользуйтесь сочетанием клавиш <Ctrl>+<Alt>+<C>. В результате вы увидите вашу программу в инструкциях на языке ассемблера (рис. 1.10).

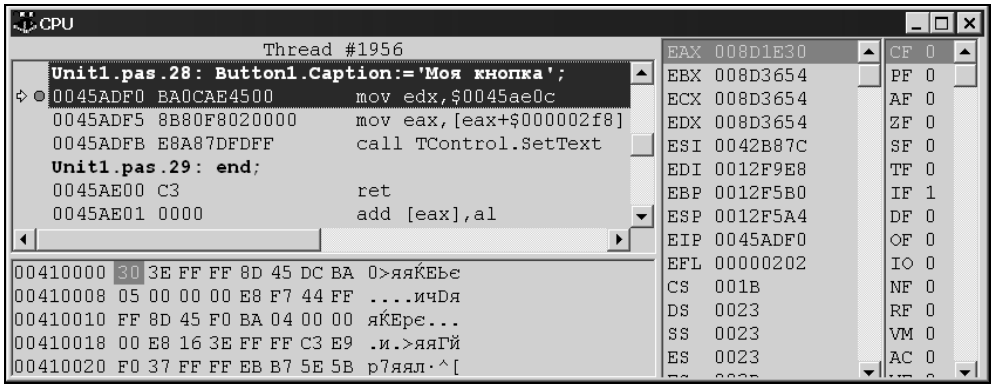

Рис. 1.10. Просмотр программы в инструкциях на языке ассемблера

Вопрос 12. Как узнать, какие модули подгружает мое приложение для своей работы?

Ответ. Создайте новое приложение. Далее нажмите клавишу <F9>, а теперь выберите пункт меню **View | Debug Windows | Modules**, после чего появится окно **Modules**, в котором вы сможете увидеть интересующую вас информацию (рис. 1.11).

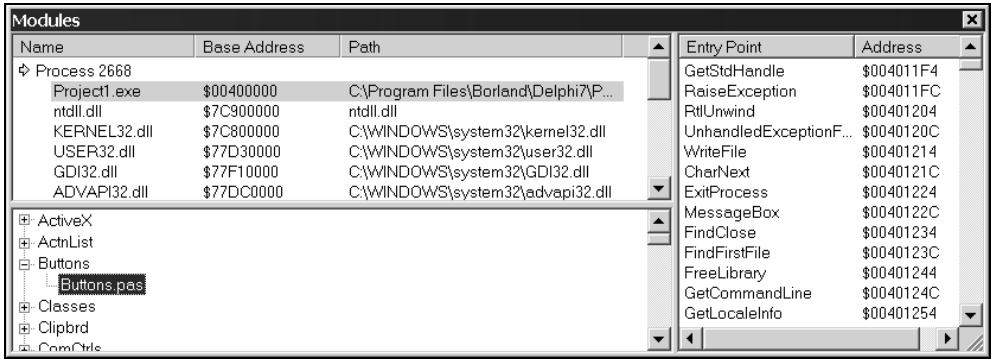

Рис. 1.11. Модули, используемые приложением для своей работы

Вопрос 13. Интересует, как можно вернуть IDE Delphi 2006 вид, напоминающий вид Delphi 7, особенно в плане палитры компонентов? Как это сделать?

Ответ.

1. Для начала сделаем, чтобы форма и ее модуль были видны одновременно. Для этого выберите пункт меню **Tools | Options**. Далее в разделе **VCL Designer** сбросьте флажок **Embedded Designer**. Изменения вступят в силу после того, как вы перезапустите Delphi.
2. Далее в качестве активного рабочего стола выберите **Classic undocked**.
3. Теперь поработаем над палитрой компонентов. Щелкните на ней правой кнопкой мыши и выберите пункт **Properties**. Далее в открывшемся окне **Options** сбросьте флажок **Show Button Captions**.

Вопрос 14. Я слышал, в Delphi 2006 включен автоматический контроль утечки памяти, как им пользоваться?

Ответ. Для этого необходимо вставить в своей программе следующую строчку кода:

```
ReportMemoryLeaksOnShutdown:=DebugHook<>0;
```

Я обычно это делаю в модуле проекта:

```
program Project1;


uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
```

```
{$R *.res}

begin
  // Включаем встроенный в Delphi 2006 менеджер памяти
  ReportMemoryLeaksOnShutdown:= DebugHook<>0;
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Если вы пользуетесь Delphi 7, то вам необходимо будет скачать менеджер памяти из Интернета. Например, можно воспользоваться одним из самых популярных — FastMM (**http://sourceforge.net/projects/fastmm/**). Его можно найти на компакт-диске. Распакуйте дистрибутив. Кстати, в поставке с ним идут примеры (папка Demos). Теперь нас интересуют следующие файлы:

- FastMM4.pas;
- FastMM4Messages.pas;
- FastMM4Options.inc.

Либо пропишите путь к ним в IDE, либо сохраните их в папке с проектом вашего приложения. Далее открывайте исходный проект с помощью пункта **Project | View Source** и самым первым модулем подключайте FastMM4:

```
program Project1;

uses
  FastMM4,
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
```

С запуском менеджера памяти мы разобрались, осталось проверить его в работе. Объявите глобальную переменную:

```
O:TObject;
```

Далее создайте для главной формы своего приложения следующий обработчик события `OnCreate`:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  O:=TObject.Create;
end;
```

Теперь запустите приложение. Как видите, мы создаем новый объект, но не уничтожаем его, как положено. Именно из-за этого могут происходить утечки памяти после завершения работы. Закройте приложение. В результате менеджер памяти сообщит вам о проблеме, которую он устранил (рис. 1.12).

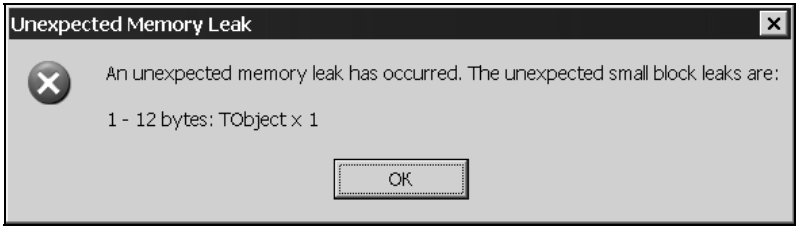

Рис. 1.12. Менеджер памяти Delphi 2006 в работе

Компакт-диск

Менеджер памяти FastMM находится в папке IDE\FastMM.

Вопрос 15. Как узнать версию компилятора? Я создаю компонент и хочу сделать его универсальным, чтобы он работал в разных версиях Delphi.

Ответ. Самый простой способ — перейти в режим командной строки и ввести `dcc32` (рис. 1.13).

Рис. 1.13. Определение версии компилятора

Как видите, таким образом можно узнать номер версии компилятора. Теперь реализуем более продвинутый пример. Создайте новое приложение, разместите на форме кнопку, для которой создайте следующий обработчик:

```
procedure TForm1.Button1Click(Sender: TObject);
const V=
  {$ifdef Ver140}'Delphi 6';{$endif}
  {$ifdef Ver150}'Delphi 7';{$endif}
```

```
  {$ifdef Ver180}'Delphi 2006';{$endif}
begin
  ShowMessage('Вы используете '+V);
end;
```

Компакт-диск

Пример — в IDE\delphi_version.

Вопрос 16. Очень часто сталкиваюсь с ошибкой "Access Violation". Хотел бы узнать, что она означает и каковы основные причины ее появления?

Ответ. Данная ошибка возникает при некорректном обращении к памяти, которое наиболее часто связано с тем, что объект, с которым происходит работа, либо еще не создан, либо уже уничтожен. Например, создайте новое приложение. Для события `OnCreate` главной формы создайте следующий обработчик:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  P:TPicture;
begin
  // Объект еще не создан, но пытаемся работать с ним.
  // В результате возникнет ошибка "Access Violation"
  P.LoadFromFile('c:\1.bmp');
end;
```

Если вы попытаетесь запустить данный пример, то возникнет ошибка "Access Violation". Точно таким же образом закончится выполнение следующего примера:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  P:TPicture;
begin
  // Создаем объект
  P:=TPicture.Create;

  // Предположим, что мы с ним поработали

  // Уничтожаем объект
  P.Free;
```

```
  // Объект уже удален, но пытаемся работать с ним.
  // В результате возникнет ошибка "Access Violation"
  P.LoadFromFile('c:\1.bmp');
end;
```

Вопрос 17. Как можно запустить Delphi без заставки?

Ответ. В командной строке напишите:

- для Delphi 7: `delphi32 -ns`;
- для Delphi 2006: `bds -ns`.

Вопрос 18. Почему при просмотре переменной, используемой циклом `for` в режиме отладки, отсчет идет в обратную сторону?

Ответ. Все дело в оптимизации. Попробуйте просмотреть значение переменной `i` в режиме отладки для следующего кода:

```
procedure TForm1.FormCreate(Sender: TObject);
var
  i:byte;
  s:string;
begin
  for i:=1 to 12 do
  begin
    s:='тест';
  end;
end;
```

А теперь выберите пункт меню **Project | Options**, перейдите на вкладку **Compiler**, сбросьте флажок **Optimization** и еще раз попробуйте поработать в режиме отладки.

Вопрос 19. Реализуя всевозможные примеры, часто много времени тратится на поиск необходимых мне компонентов (я не помню, на какой вкладке они находятся). Как можно ускорить этот процесс?

Ответ. Для Delphi 7 выберите пункт меню **View | Component List** и в появившемся окне **Components** введите первые буквы названия компонента, например, `TBut`.

При работе в Delphi 2006 нужно нажать кнопку **Filter or unfilter the current items**, расположенную в палитре компонентов, далее убедиться, что у вас

включена английская раскладка, после чего ввести первые буквы названия компонента, например, при вводе `TBut` палитра компонентов будет выглядеть так, как показано на рис. 1.14.

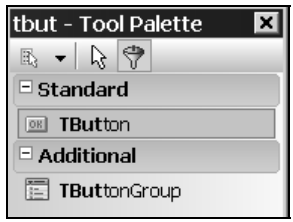

Рис. 1.14. Поиск компонента в Delphi 2006 по первым буквам его названия

Вопрос 20. Можно ли просмотреть историю изменений исходного кода моей программы, не используя средств коллективной работы?

Ответ. Да, такая возможность появилась, начиная с Delphi 2005. Перейдите в режим просмотра модуля формы и выберите вкладку **History**, расположенную снизу окна IDE (рис. 1.15).

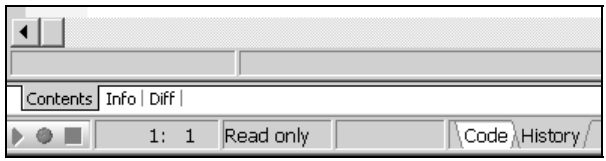

Рис. 1.15. Просмотр истории изменений кода программы

Вопрос 21. Я потерял исходные коды своей программы, остался только exe-файл. Можно ли по нему восстановить код, и как это сделать?

Ответ. Существует несколько программ для данного случая. Я обычно использую DeDe (Delphi Decompiler), которую можно скачать с **www.vr-online.ru**. Только сразу стоит отметить один важный нюанс: полностью восстановить исходный код нельзя, т. к. это связано с тем, что при компиляции программы на языке высоко уровня (которым является Delphi) происходит ее преобразование в язык более низкого уровня — машинные коды. В результате максимум, чего можно добиться, — это получить ресурсы, используемые программой (например, формы) и код на языке ассемблера. Поэтому

во избежание дополнительных сложностей делайте вовремя резервные копии своей информации.

Вопрос 22. Как уменьшить размер получаемого после компиляции программы exe-файла?

Ответ. Для этого существуют всевозможные упаковщики. Например, мне очень нравится ASPack, его можно скачать с сайта **http://www.aspack.com/downloads.html** (официальный ресурс). Также очень популярен UPX. Принцип работы их простой: вы указываете свой exe-файл, затем программа сжимает его и на выходе получается уже упакованный файл меньшего размера. Например, если вы создадите пустое приложение (в Delphi 2006), затем откомпилируете его, то на выходе получится файл размером 386 Кбайт. Теперь воспользуемся ASPack, в результате наша программа стала "весить" 186 Кбайт — на лицо уменьшение в более чем в 2 раза.

Вопрос 23. В Delphi 7 есть такой пункт меню **Component | Install component**, посредством которого выбирается модуль PAS и производится установка компонента. Как это действие осуществить в Delphi 2006?

Ответ. В Delphi 2006 для этого нужно воспользоваться пакетом. Выберите пункт меню **File | New | Package - Delphi for Win32**. Далее в менеджере проекта щелкните правой кнопкой мыши на папке **Contains** и выберите команду **Add...**. В появившемся диалоговом окне выберите необходимый модуль и нажмите кнопку **OK**. Затем еще раз щелкните правой кнопкой мыши, только теперь на имени пакета, по умолчанию он называется Package1.bpl, и выберите команду **Install** (рис. 1.16).

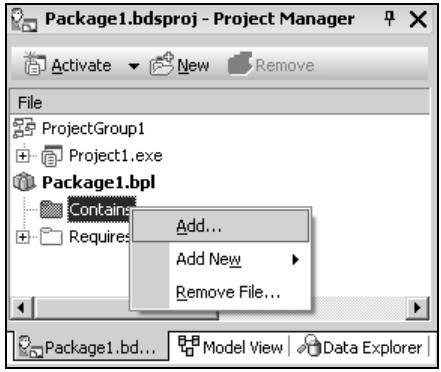

Рис. 1.16. Создание пакета

Глава 2

Delphi

В этой главе рассматривается работа со стандартными компонентами, входящими в поставку Delphi, различные трюки с ними, усовершенствование предоставляемых ими возможностей, а также создание на их базе новых компонентов — собственной разработки. Будет уделено внимание реализации полезных программ, таких как "Увеличительное стекло", "Работа с Корзиной Windows", "Завершение сеанса пользователя" и многих других

Вопрос 1. Как с помощью скроллинга мыши уменьшать и увеличивать масштаб просмотра картинки?

Ответ. Разместите на форме компонент `TImage`, загрузите в него любую картинку, постарайтесь установить размеры `TImage` в соответствии с загруженной картинкой, для свойства `Strech` установить значение `True` и для события `OnMouseWheelDown` и `OnMouseWheelUp` вашей формы напишите следующий код:

```
procedure TForm1.FormMouseWheelUp(Sender: TObject; Shift: TShiftState;
  MousePos: TPoint; var Handled: Boolean);
begin
  Image1.Width:=Image1.Width * 2;
  Image1.Height:=Image1.Height * 2;
end;

procedure TForm1.FormMouseWheelDown(Sender: TObject; Shift: TShiftState;
  MousePos: TPoint; var Handled: Boolean);
begin
  Image1.Width:=Image1.Width div 2;
  Image1.Height:=Image1.Height div 2;
end;
```

Компакт-диск

Пример — в Delphi\scrolling_for_picture.

Вопрос 2. Как сделать, чтобы форма "прилипала" к краю экрана?

Ответ. Установите для нее свойство `ScreenSnap` в значение `True`. Оно отвечает за режим прилипания, а в свойстве `SnapBuffer` задайте число, характеризующее расстояние между краем экрана и краем формы, при котором происходит прилипание. (Рекомендую задать 50.) Запустите проект и попробуйте медленно переместить форму к краю экрана, вы увидите, как она прилипнет.

Компакт-диск

Пример — в Delphi\stick_to_edge.

Вопрос 3. Как поменять цвет у `TProgressBar`?

Ответ. Следующая команда сделает `TProgressBar` желтого цвета:

```
PostMessage(ProgressBar1.Handle, $0409, 0, clYellow);
```

Компакт-диск

Пример — в Delphi\change_col_for_prgbar.

Вопрос 4. Как использовать `TProgressBar` в качестве средства ожидания, чтобы пользователь видел, что программа при длительном процессе не зависла?

Ответ. Общий алгоритм программирования `TProgressBar` достаточно прост:

1. Сколько операций надо выполнить — устанавливаем это число в свойстве `max` для `TProgressBar`.
2. Начинаем выполнение длительного процесса, который, как правило, состоит из множества мелких операций.
3. После выполнения очередной операции увеличиваем свойство `Position` для `TProgressBar` на единицу.
4. Вызываем процедуру `ProcessMessages`, чтобы не занимать все процессорное время, а дать поработать другим программам.
5. Возвращаемся к пункту 3, пока не будет выполнена последняя операция.

Рассмотрим реализацию программы сравнения двух картинок попиксельно, для показа хода сравнения будет использоваться `TProgressBar`. Разместите

на форме два компонента TImage, загрузите в оба картинки формата BMP. Далее разместите два компонента TLabel: в свойстве Caption первого введите "Различий найдено:", а второй просто поставьте рядом. Далее разместите кнопку **Сравнить картинки**, а под ней TProgressBar, свойство Smooth которого установите в True (рис. 2.1).

Рис. 2.1. Вид главной формы к примеру "Использование TprogressBar"

Теперь для кнопки **Сравнить картинки** напишите следующий обработчик:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  // i-координата пиксела по горизонтали
  i:integer;
  // j-координата пиксела по вертикали
  j:integer;
  // Количество различий
  k:integer;
begin
  k:=0;
  // Если картинки разные по размеру, то смысла сравнивать нет,
  // поэтому осуществляем выход
```