РАЗРАБОТКА ПРИЛОЖЕНИЙ В СРЕДЕ *DELPHI*

Содержание

1. Цель работы

1. Знакомство со средой визуального программирования Delphi
2. Получение начальных навыков разработки приложений в среде Delphi

2. Теоретическая часть

2.1. Основные элементы интерфейса среды Delphi

Delphi состоит из ряда элементов, которые всегда присутствуют на экране: главного окна (включающего панель инструментов и палитру компонентов), инспектора объектов и двух окон – окна для визуального создания приложений и окна редактирования кода, а также утилит, которые становятся доступными в определенных случаях, например, дизайнера меню, графического редактора и др.

Главное окно

Помимо главного меню, содержащего базовые команды типа File, Edit, Search, View, Project, Run, Component, Database, Tools и Help, включает панель инструментов (полоса быстрого доступа к командам) и палитру компонентов.

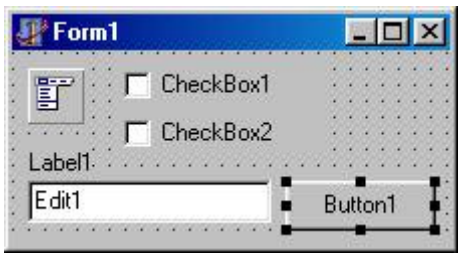
Панель инструментов

Содержит кнопки, работающие по принципу «укажи и щелкни» и выполняющие наиболее часто употребляемые команды главного меню.

Палитра компонентов

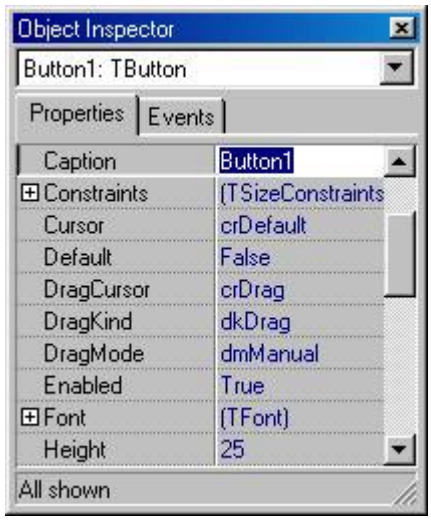
Содержит пиктограммы, которые представляют компоненты VCL (Visual Component Library). Позволяет выбрать компоненты, которые должны присутствовать в разрабатываемом приложении. Компоненты бывают визуальными и невизуальными. Первые предназначены для организации интерфейса с пользователем. Это различные кнопки, списки, статический и редактируемый тексты, изображения и многое другое. Эти компоненты отображаются при выполнении разрабатываемого приложения. Невизуальные компоненты отвечают за доступ к системным ресурсам: драйверам баз данных, таймерам и т.д. Во время разработки они отображаются своей пиктограммой, а при выполнении приложения, как правило, невидимы. Компонент может принадлежать либо другому компоненту, либо форме.

Окно для визуального создания приложений



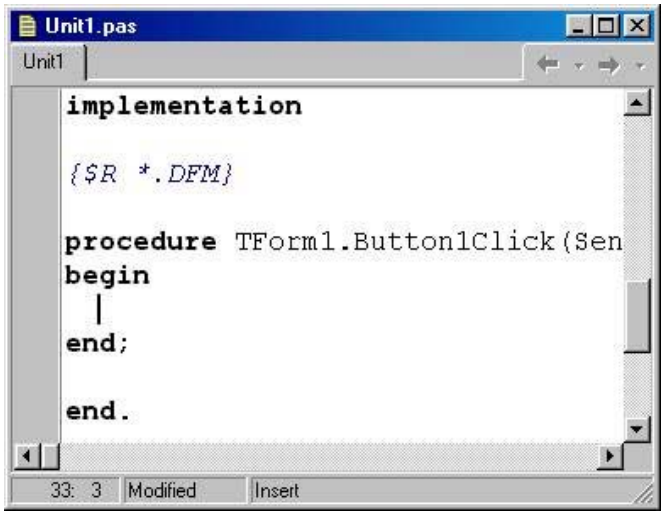
Во многих приложениях представляет собой форму. Формой называется визуальный компонент, обладающий свойством окна Windows. При разработке, на форме помещаются необходимые компоненты (например, элементы требуемого диалога). Простые программы имеют только одну форму, а более сложные приложения могут обладать множеством таких форм.

Инспектор объектов (Object Inspector)



Позволяет устанавливать свойства выбранных компонентов и назначать методы-обработчики событий во время разработки программы. Свойство представляет атрибут компонента, например размер кнопки или шрифт метки. Событие обозначает такое действие, как щелчок мышью или нажатие клавиши.

Окно редактирования кода



Содержит текст программы на языке Pascal, связанный с каждой формой в приложении. Delphi автоматически создает этот программный код, в который можно добавлять операторы, выполняемые при выборе команды меню или щелчке на кнопке. Это окно также используется для редактирования других модулей программы и текстовых файлов.

2.2. Файлы, создаваемые средой, при разработке проектов

Среда Delphi работает с проектами – наборами файлов, из которых состоит разрабатываемое приложение. Главный файл проекта имеет расширение **.DPR**. Для каждого проекта может быть только один такой файл. По умолчанию проект имеет название Project1. Этот файл связывает вместе все файлы, из которых состоит приложение.

Для каждой формы, включаемой в проект, создается отдельный модуль (файл с исходным текстом, имеющий расширение **.PAS**. В этом файле хранится код, который пишется разработчиком приложения – объявление переменных, типов, код обработчиков сообщений для интерфейсных элементов, дополнительный код и т.п. В проект можно включать модули и не связанные с формами.

Файл с исходным текстом с помощью директивы компилятора $R подключает двоичный образ формы (который сохраняется в файле с расширением **.DFM** и имеет формат файла ресурсов Windows). Этот файл – бинарный, и он подключается непосредственно к исполнительному файлу в момент компиляции программы. Файл формы – это список свойств всех компонентов, включенных в форму, значения которых были изменены сравнительно со значениями, задаваемыми по умолчанию (в конструкторе соответствующего объекта). Кроме этого, файл форм связывает графическое представление формы с обработчиком сообщений.

Для каждого проекта создается файл опций (файл с расширением **.DOF**), в который записываются значения опций компилятора, компоновщика и названия рабочих каталогов. Для того чтобы восстановить оригинальные значения, необходимо удалить или переименовать этот файл.

В файл ресурсов с расширением **.RES** входят ресурсы, не вошедшие в формы, например, код значка приложения, иконка которого будет видна при его свертывании.

Результатом компиляции всех Delphi-проектов является **исполняемый файл**. Это может быть либо программа (файл с расширением **.EXE**), либо динамически загружаемая библиотека (файл с расширением **.DLL**).

2.3. Компиляция, сборка и выполнение программ

Программа может быть откомпилирована и выполнена на любой стадии создания. Это бывает удобно для проверки работы интерфейсных элементов и правильности их взаимодействия, а также для проверки функциональности отдельных фрагментов создаваемого кода.

Для компиляции исходных файлов, входящих в проект, используется команда **Project|Compile** главного меню интегрированной среды разработчика или комбинация клавиш **Ctrl-F9**. При этом выполняются следующие действия:

компилируются файлы с исходным текстом всех модулей, содержимое которых изменялось после последней компиляции. В результате для каждого файла с исходным текстом модуля создается файл с расширением **.DCU**. Если исходный текст модуля по каким-то причинам недоступен компилятору, то модуль не перекомпилируется;

если были внесены изменения в интерфейсную часть модуля, то перекомпилируется не только этот модуль, но и модули, использующие его (через директиву **uses**);

перекомпиляция модуля происходит также при изменении объектного файла (файла с расширением **.OBJ**), используемого в данном модуле;

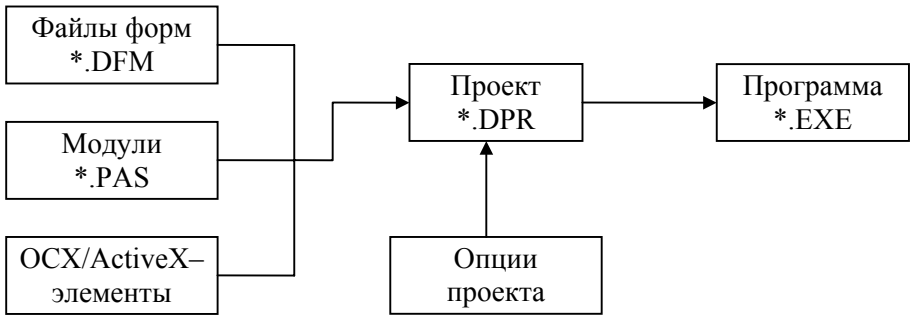
перекомпиляция модуля происходит также при изменении включаемого файла (файла с расширением **.INC**), используемого тем или иным модулем.

После того как откомпилированы все модули, входящие в проект, Delphi компилирует файл проекта и создает **.EXE** файл с именем, эквивалентным имени проекта.

При сборке проекта в отличие от компиляции компилируются все файлы, входящие в проект вне зависимости от того, были внесены в них изменения после предыдущей компиляции или нет. Для сборки проекта используется команда **Project|Build All**.

Для выполнения программы используется команда **Run|Run** главного меню интегрированной среды разработчика или клавиша **F9**. При вызове этой команды происходят те же действия, что и при вызове команды **Project|Compile**, но после компиляции программа запускается на выполнение.

Процесс создания исполняемого файла показан на рисунке.



3. Практическая часть

3.1. Разработка приложения «Memo Pad»

1. С помощью Проводника Windows создайте каталог C:\Projects\Memopad, где будут храниться файлы вашего приложения.

2. Начните новый проект. Выберите форму, щелкнув внутри нее, а в окне Object Inspector измените ее свойство Caption на Memo Pad.

3. Значение свойства Name измените на MainForm.

4. Для создания файлов проекта выполните команду File/Save Project панели инструментов или щелкните на кнопке Save Project панели инструментов.

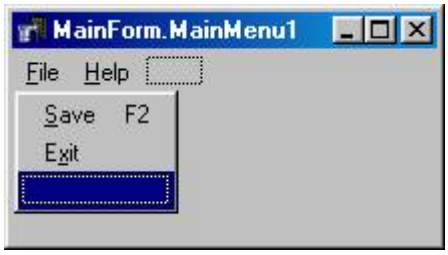
5. Delphi отобразит на экране один за другим два диалоговых окна. В первом из них, озаглавленном Save Unit1 As, перейдите в каталог, созданный при выполнении п. 1, и введите Main в строке File Name. Нажмите <Enter> или щелкните на OK. Во втором диалоговом окне с заголовком Save Project1 As в качестве имени файла введите Memopad, нажмите <Enter> или щелкните на OK.

6. На форме разместите компоненты MainMenu и PopupMenu. В данном случае для обозначения объектов используйте значения по умолчанию свойства Name обоих меню: MainMenu1 и PopupMenu1. Переместите пиктограммы в верхний левый угол формы (месторасположение не играет роли).

7. Дважды щелкните на объекте MainMenu1. Откроется окно Menu Designer. Для создания меню File введите &File и нажмите <Enter>, чтобы изменить свойство Caption этого объекта меню.

8. Введите &Save для создания пункта Save. Выберите свойство ShortCut, щелкните на кнопке с направленной вниз стрелкой и в качестве клавишного эквивалента для данного пункта выберите <F2>.

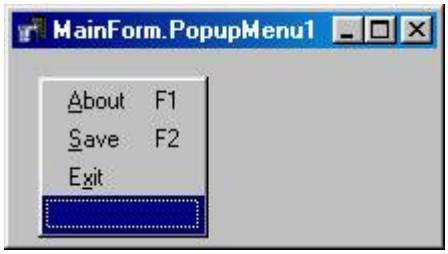
9. В Menu Designer щелкните на свободном месте ниже пункта Save и выберите свойство Caption. Для создания пункта Exit введите E&xit.



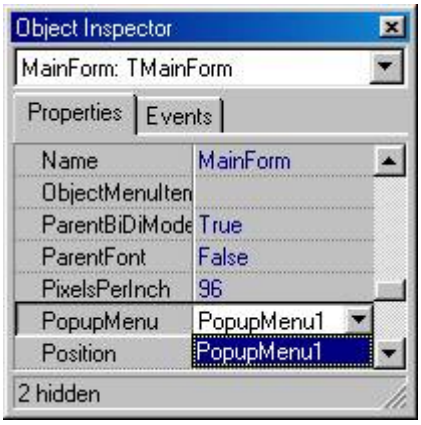
10. Точно так же создайте меню Help, а в нем пункт About. Назначьте <F1> в качестве клавишного эквивалента для данной команды, как это было сделано с клавишей <F2> для пункта Save.

11. Закройте окно Menu Designer. Проверьте работу этого меню с помощью клавиатуры не щелкая пока на пунктах меню.

12. Откройте Menu Designer, дважды щелкнув на объекте PopupMenu. Появится всплывающее меню. Введите в это меню команды About, Save и Exit. Перед буквами, которые должны быть подчеркнуты, вставьте амперсанды (&). С помощью свойства ShortCut каждой команды назначьте командам клавишные эквиваленты F1 и F2.



13. Закройте окно Menu Designer, а затем щелкните на форме, чтобы отменить выбор всех компонентов. В раскрывающемся списке окна Object Inspector вы увидите MainForm:TMainForm. Свойству PopupMenu формы присвойте значение PopupMenu1. Введите имя или выберите его из раскрывающегося списка данного свойства. В результате, после щелчка правой кнопкой мыши, форма отобразит объект PopupMenu.



14. С помощью клавиши <F9> скомпилируйте и запустите приложение и проверьте команды меню программы (ни одна из которых еще не работает), и, щелкнув правой кнопкой мыши, вызовите всплывающее меню, которое в данный момент также не действует. Чтобы вернуться в Delphi, воспользуйтесь комбинацией клавиш <Alt+F4> или кнопкой закрытия окна.

15. На форме разместите метку (компонент Label), и ее свойству Name присвойте значение MemoLabel1. Нажмите <Enter> и убедитесь, что свойству Caption метки Delphi присвоила то же значение.

16. Нежелательно, чтобы объект показывал свое внутреннее имя. Для изменения отображаемого заголовка метки присвойте ее свойству Caption значение «Примечание:».

17. Разместите на форме примечание (компонент Memo). Все символы значения свойства Lines следует удалить, чтобы область ввода была пустой. Для этого выберите свойство Lines объекта Memo1 и щелкните на находящейся справа кнопке с многоточием (кнопка подстановки). В результате откроется диалоговое окно String list editor, которое используется для удаления строки Memo1. Для сохранения изменений нажмите <Enter> или щелкните на кнопке ОК редактора.

18. Измените размер Memo1 в соответствии с размерами формы.

19. Чтобы создать код, который будет выполняться при выборе пунктов меню, необходимо в окне формы выбрать нужный пункт (File/Save) и щелкнуть на нем. При этом в текст программы вставляется заготовка процедуры, называемой обработчиком события, а курсор размещается там, где следует ввести оператор, который будет определять действия данного пункта меню. Между begin и end введите следующий оператор:

```
Memo1.Lines.SaveToFile('memos.txt');
```

20. Завершите создание других команд приложения MemoPad, щелкнув на пункте File/Exit меню формы. Между ключевыми словами begin и end введите

```
Close;
```

21. Вернитесь в окно формы и щелкните на пункте Help/About. Между begin и end введите следующий оператор:

```
MessageDlg('Memo Pad'#13#10'© 2003 ВолгГТУ'#13#10'Версия 1.00',
mtInformation, [mbOK], 0);
```

22. Для создания обработчиков событий, поступающих от всплывающего меню, дважды щелкните на объекте PopupMenu1 (на пиктограмме изображено меню со стрелкой). При этом вновь откроется окно Menu Designer. В окне Object Inspector щелкните на закладке страницы Events. В Menu Designer выберите пункт About и присвойте событию OnClick этого пункта значение About1Click. Имя обработчика события About1Click можно ввести вручную или выбрать из раскрывающегося списка события OnClick. Событию OnClick пункта Save присвойте значение Save1Click, а событию OnClick пункта Exit – Exit1Click.

23. Закройте окно Menu Designer. Щелкните на форме и, если необходимо, выберите закладку страницы Events в окне Object Inspector. Дважды щелкните на пустой строке ввода справа от события OnActivate и введите следующий оператор if-then-else между ключевыми словами begin и end в окне редактирования модуля. Данный оператор считывает текст из указанного файла в объект Lines компонента:

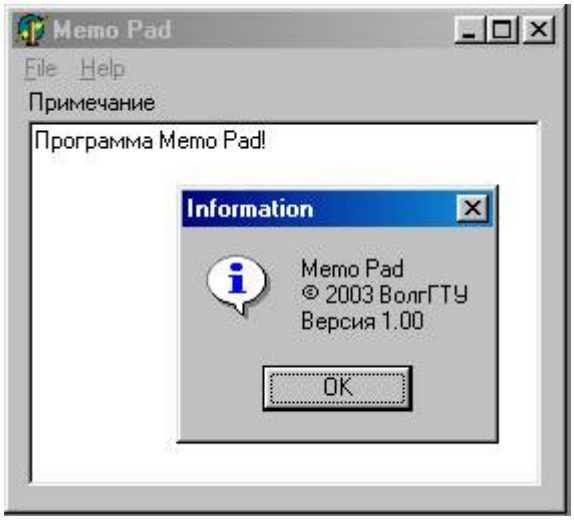
```
If FileExists('memos.txt')
   then Memo1.Lines.LoadFromFile('memos.txt')
   else Memo1.Lines.SaveToFile('memos.txt');
```

24. Выберите форму и дважды щелкните на пустой строке справа от события OnClose. Для этого события Delphi вставит в текст программы заготовку процедуры, выполняемой при закрытии окна программы. Введите следующий оператор между ключевыми словами begin и end, чтобы сохранить объект Lines в указанном файле:

```
Memo1.Lines.SaveToFile('memos.txt');
```

25. Для компиляции, компоновки и запуска завершенной программы нажмите клавишу <F9>. Проверьте работу меню программы.

26. Проведите русификацию программы и продемонстрируйте работу программы преподавателю.



3.2. Разработка заставки к программе Memo Pad

1. Выберите пункт *File⇒New Form* или щелкните на кнопке *New Form* панели инструментов. В результате появится новая форма.

2. Свойству Name этой формы присвойте значение SplashForm, а значение ее свойства Caption удалите. Кроме того, свойству BorderStyle присвойте значение bsNone, а три подсвойства комплексного свойства BorderIcons установите в False.

3. Сохраните проект. Когда Delphi предложит ввести имя файла, убедитесь, что текущим является каталог C:\Projects\Memopad. Файлу модуля проекта присвойте имя Splash.

4. Свойству Enabled формы SplashForm присвойте значение False (запрет пользователю на управление окном с помощью клавиатуры и мыши).

5. Измените размеры окна SplashForm. Так как окно не имеет рамки, разместите на нем фаску (компонент Bevel на странице Additional). Это позволит обозначить границы окна. Свойству Align фаски присвойте значение alClient, а свойствам Shape и Style объекта – соответственно bsFrame и bsRaised. Можно использовать другие значения.

6. Разместите на SplashForm экземпляры компонентов Image и Label. В свойстве Caption компонента Label поместите свою фамилию и номер группы.

7. Свойствам FormStyle и Position формы SplashForm присвойте значения соответственно fsStayOnTop и poScreenCenter (заставка будет находиться на переднем плане и в центре экрана).

8. Выберите *Project ⇒ Options*. В нижней части появившегося диалогового окна *Project Options* выберите закладку страницы Forms. Обе формы MainForm и SplashForm присутствуют в списке *Auto-create forms*. Выберите SplashForm и щелкните на кнопке с изображением направленной вправо стрелки, чтобы переместить данную форму в *Available forms*. Все формы Delphi по умолчанию создаются в памяти автоматически, что приводит к нерациональному использованию памяти и системных ресурсов. В подобных случаях имя формы следует удалять из списка автосоздания. Закройте диалоговое окно.

9. Преобразуйте исходный код проекта так, чтобы заставка отображалась до появления главного окна. Выполните команду *Project ⇒ View Source*. Измените операторы между *begin* и *end* так, чтобы код операторного блока файла проекта Memopad.Dpr выглядел следующим образом.

```
begin
  SplashForm := TSplashForm.Create(Application);
  { Метод Create создает объект формы заставки}
  SplashForm.Show;
  SplashForm.Update;
  Application.CreateForm(TMainForm, MainForm);
  SplashForm.Hide;
  SplashForm.Free;
  { Метод Free освобождает память,
  выделенную для объекта SplashForm}
  Application.Run;
end.
```

10. Чтобы заставка оставалась на экране несколько секунд, выберите форму MainForm программы. Создайте обработчик события OnCreate формы. Перед ключевым словом *begin* объявите переменную CurrentTime типа LongInt. Между *begin* и *end* вставьте два оператора, первый из которых вызывает функцию Windows GetTickCount для присвоения переменной CurrentTime текущего значения времени работы Windows в секундах, а второй (оператор *while*) обеспечивает дополнительную задержку еще на 4 секунды.

```
procedure TMainForm.FormCreate(Sender: TObject);
var
  CurrentTime: LongInt;
begin
  CurrentTime := GetTickCount div 1000;
  { Функция GetTickCount возвращает время
  работы Windows в миллисекундах }
  while ( (GetTickCount div 1000) < (CurrentTime + 4) ) do
  { Ничего не выполняется};
end;
```

11. Нажав <F9>, откомпилируйте, скомпонуйте и запустите проект. Программа отобразит заставку и через несколько секунд удалит ее, а затем отобразит главное окно.

12. Выберите MainForm щелкнув на ней. В окне *Object Inspector* перейдите на страницу Events и щелкните на поле значения справа от события OnCloseQuery. Введите следующий код между ключевыми словами *begin* и *end*.

```
begin
  if MessageDlg('Закончить программу?',
     mtConfirmation, [mbYes,mbNo], 0) = mrYes
     then CanClose := True
     else CanClose := False;
end;
```

Выполнение данного кода приведет к появлению на экране диалогового окна с сообщением и кнопками *Yes* и *No*. Если пользователь выберет *Yes*, функция MessageDlg вернет значение mrYes, а оператор *if* установит CanClose в True. Если пользователь выберет *No*, программа установит CanClose в False.