ПУСТЬ ЭТА КНИГА ПРИНЕСЕТ ВАМ УДАЧУ!

С.В. Федотова

СОЛОН

Создание Windows-приложений в среде Delphi

Начальный учебный курс по основам Visual-программирования

Для студентов и преподавателей!

Delphi для повседневных задач
Создаем интерфейс Windows-приложений
Работаем с графикой и мультимедиа
Дистанционная методика обучения
Примеры и методические материалы на CD!

КОМПАКТ-ДИСК



серия Библиотека студента

УДК 004.5
ББК 32.973.26-018.2
Ф 34

С. В. Федотова

Ф34 Создание Windows-приложений в среде Delphi. — М.: СОЛОН-Пресс, 2010. — 224 с.: ил. — (Серия «Библиотека студента»)

ISBN 5-98003-176-6

Учебное пособие «Создание Windows-приложений в среде Delphi» (в комплекте с CD) предназначено для учащихся средних школ, студентов технических вузов, преподавателей информатики. Данный курс программирования предполагает последовательное изучение материала от простого к сложному. Читателю, впервые приступившему к изучению темы, предлагаются все необходимые сведения для понимания и соответствующей организации процесса программирования при создании Windows-приложения. Большая часть материалов книги посвящена разбору и решению практических задач. Все примеры программ, а также дополнительная информация методического плана представлены на компакт-диске пособия.

Освоив предлагаемый курс по этой книге, вы сможете самостоятельно приступить к разработке программ, позволяющих решать широкий круг задач пользовательского характера, в том числе создавать овальные, непрямоугольные окна Windows-приложений, проектировать нестандартный интерфейс. В дальнейшем читатель сможет перейти к более детальному самостоятельному освоению среды Visual-программирования.

Занятия по курсу позволят преподавателям организовать работу с учащимися по оформлению исследовательских работ, в проектной деятельности, помогая создавать прикладные программы, соответствующие современному высокому технологическому уровню, а также разрабатывать собственный дидактический материал, разнообразные формы опроса педагогической деятельности, создавать вспомогательные методические пособия.

По вопросам приобретения обращаться:
ООО «Альянс-книга»
Тел: (095) 258-91-94, 258-91-95, **www.abook.ru**

ISBN 5-98003-176-6

© Федотова С. В., 2010
© Макет и обложка «СОЛОН-Пресс», 2010

Часть 3
Основные элементы Visual-программирования

Глава 1. Object Pascal — алгоритмический язык Delphi

Object Pascal является базовым языком Delphi, который представляет собой развитие Borland Pascal с учетом объектно-ориентированного программирования. Визуальное программирование на Object Pascal реализуется посредством компонентной модели, построенной на библиотеке визуальных компонентов Delphi — VCL. Алгоритмическое программирование большей частью остается традиционным, основанным на средствах и приемах языка Pascal.

Алфавит

Набор символов Object Pascal так же, как в Turbo Pascal, является подмножеством набора символов кода ASCII.

- Прописные и строчные буквы латинского алфавита, а также символ подчеркивания, который используется наравне с буквами.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
коды ASCII от 65 до 90

a b c d e f g h i j k l m n o p q r s t u v w x y z
коды ASCII от 97 до 122

«_»
код ASCII 95

- Арабские десятичные цифры.

0 1 2 3 4 5 6 7 8 9
коды ASCII от 48 до 57

- Специальные символы (в скобках показан код ASCII).

(35) — (45) @ (64)

$ (36)	. (46)	[(91)
‘ (39)	/ (47)	] (93)
((40)	: (58)	^ (94)
) (41)	; (59)	{ (123)
* (42)	< (60)	} (125)
+ (43)	= (61)	
, (44)	> (62)	

- Символ пробела (код ASCII 32).
- Управляющие символы (коды ASCII от 0 до 31).

Разделители

В качестве разделителей лексем друг от друга применяются символы:

- пробел (код ASCII 32);
- табуляция (код ASCII 09);
- составной символ перехода в начало следующей строки (пара символов «возврат каретки» (код ASCII 13) и «перевод строки» (код ASCII 10)).
- Кроме этих «классических» разделителей, могут использоваться любые управляющие символы набора кодов ASCII из диапазона от символа с кодом 0 до символа с кодом 31.

Идентификаторы

Следом за зарезервированным словом **PROGRAM** в программе или **Unit** в модуле стоит пробел, разделитель слов, и далее — имя, данное программе. Это имя представляет собой пример идентификатора. **Идентификатор** — имя, свободно избираемое программистом для элементов программы (модулей, процедур, функций, констант, переменных и типов данных). При инициализации какого-либо элемента программы с помощью идентификатора Вы должны руководствоваться следующими постулатами:

- Идентификатор должен начинаться буквой или символом подчеркивания: _ . За исключением имени метки label, которое может быть числом по желанию пользователя.
- Применение других символов (букв неанглийского алфавита, знаков препинания, псевдографических символов и т. п.) в идентификаторах не допускается.

- Object Pascal не различает прописные или строчные буквы. Поэтому можно записать TimeToStr, Timetostr или даже timeTostR, не опасаясь быть не понятым компилятором. Для него все три записи эквивалентны.
- Начиная со второй позиции, в идентификаторе можно применять наряду с буквами цифры.
- Пробел является разделителем и не может стоять внутри идентификатора. Для создания идентификаторов, состоящих из двух слов, можно воспользоваться большими буквами (например, bbRun) или символом подчеркивания (bb_Run), но не пробелом (bb Run).
- Между двумя идентификаторами должен быть как минимум один разделитель.
- Там, где стоит один разделитель, может быть сколько угодно разделителей.
- Зарезервированные слова (такие, как BEGIN, END или UNIT) в качестве идентификаторов не используются.
- Идентификаторы могут быть любой длины, но сравнение их между собой производится по первым 63 символам.

Зарезервированные слова

ABSOLUTE	FUNCTION	PROGRAM
AND	GOTO	PUBLIC
ARRAY	IF	RECORD
ASM	IMPLEMENTATION	REPEAT
ASSEMBLER	IN	SET
BEGIN	INLINE	SHL
CASE	INTERFACE	SHR
CONST	INTERRUPT	STRING
CONSTRUCTOR	LABEL	THEN
DESTRUCTOR	MOD	TO
DIV	NEAR	TYPE
DO	NIL	UNIT
DOWNTO	NOT	UNTIL

ELSE	OBJECT	USES
END	OF	VAR
EXTERNAL	OR	VIRTUAL
FAR	PACKED	WHILE
FILE	PRIVATE	WITH
FOR	PROCEDURE	XOR
FORWARD		и др.

Структура программы (проект)

{I . Заголовок программы}
PROGRAM имя ; { первая обязательная фраза }
{II . Раздел указания используемых модулей}
USES Список используемых модулей;
{III . Раздел описаний}
label описания меток;
const описания констант;
type описания типов;
var описания переменных;
procedure
function описания процедур и функций;
exports описания экспортируемых имен;
{IV . Раздел операторов (операторный блок)}
BEGIN

Операторная часть программы

END.

При компиляции программы создается исполняемый файл приложения: *.exe.

Модули

Библиотека прикладных программ, как мощное средство языка, в Object Pascal поддерживается в основном за счет модульного программирования.

Модуль — автономно компилируемая программная единица. В отличие от программы модуль может содержать элементы (переменные или подпрограммы), которые допускается использовать в программе или других модулях. Чтобы можно было использовать модуль, его необходимо подключить к программе в разделе описаний uses. Самостоятельно запустить модуль нельзя, в отличие от программы. При компиляции модуля образуется не исполняемый exe-файл, а *.dcu. Дальнейшая работа компоновщика IDE сводится к преобразованию и сведению всех модулей *.dcu в единый файл загружаемого exe-приложения Windows.

Модуль состоит из заголовка, в котором после ключевого слова unit указывается имя модуля и четырех разделов: интерфейса (Interface), реализации (Implementation), инициализации (Initialization) и деинициализации (Finalization).

Структура модуля:

Unit <Имя модуля>; *// Раздел интерфейса* **Interface** **Uses** <список модулей>; **Const** <список констант>; **Type** <описание типов>; **Var** <объявление переменных>; <заголовки процедур>; <заголовки функций>;	В разделе интерфейса размещаются описания идентификаторов, которые должны быть доступны всем модулям и программам, использующим этот модуль и содержащим его имя в списке uses.
// Раздел реализации **Implementation** **Uses** <список модулей>; **Const** <список констант>; **Type** <описание типов>; **Var** <объявление переменных>; <описание процедур>; <описание функций>;	В разделе реализации (исполняемая часть модуля) располагается код подпрограмм, заголовки которых были приведены в разделе интерфейса. Можно также описывать типы, объявлять константы и переменные и описывать подпрограммы, которые используются только в этом модуле и за его пределами не видны.
// Раздел инициации **Initialization** <операторы инициирующей части>	

// Раздел деинициализации //(завершение) **Finalization** <операторы завершающей части> **End**.	(Разделы **Initialization** и **Finalization** чаще отсутствуют в обычной практике программирования.)

Имя модуля служит для его связи с другими модулями и основной программой (список uses). Элементами списка uses являются имена модулей, отделяемые друг от друга запятыми.

Инициирующая и завершающая части чаще всего отсутствуют.

Раздел инициации нужен для размещения операторов, которые должны исполняться до передачи управления основной программе, например для подготовки ее работы (инициация переменных, открытие файлов и т. п.). Завершающая часть содержит операторы, выполняющиеся после основной программы (закрытие файлов, освобождение ресурсов, выделенных в программе).

Вспомогательные алгоритмы. Процедуры и функции

Алгоритм — конечный набор команд, организованный в определенной последовательности с целью решения поставленной задачи.

Алгоритмы составляются для всех задач, если они предназначены для реализации компьютером.

Алгоритм, написанный на языке программирования, является программой. Ярко выраженная тенденция к укрупнению — отличительная черта современного программного обеспечения, состоящего из десятков тысяч, миллионов строк кода. Естественно, что для осуществления подобных разработок требуются эффективные средства организации самого процесса. Метод последовательного построения (пошаговой детализации) явился результатом развития всей системы программирования. Суть метода в разделении сложной задачи на ряд простых. Для каждой подзадачи составляется свой алгоритм, называемый **вспомогательным**.

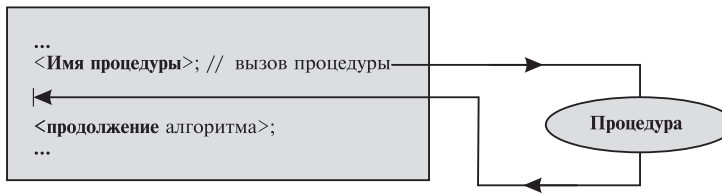
Основное требование в образовавшейся иерархической структуре — это обеспечение строгой взаимосвязи между составными частями программы. Данный подход в программировании является основополагающим в теории алгоритмизации. В структуриро-

ванных программах легко прослеживается логика основного алгоритма, они удобны в отладке, в них легче выявить ошибки. Кроме того, довольно просто осуществляется многократное использование подзадач в одном или даже нескольких алгоритмах.

В коде вспомогательные алгоритмы организуются в **подпрограммы**, к которым относятся процедуры и функции.

Процедура — фрагмент программы, имеющий собственное имя и предназначенный для выполнения последовательности команд вспомогательного алгоритма.

Упоминание имени процедуры в коде называется ее **вызовом**. При вызове выполнение текущей очередности команд приостанавливается и управление процессом передается процедуре. После выполнения последнего оператора процедурного блока управление возвращается в точку вызова к оператору, стоящему непосредственно после имени процедуры.



Описание процедуры в программе проводится перед операторной частью Begin... End. Заголовок состоит из зарезервированного слова **Procedure** и идентификатора имени. Далее идет исполнительный блок, аналогичный по структуре главной программе. Завершается процедура точкой с запятой: «**;**».

В модуле процедуры располагаются в разделе реализаций (**Implementation**) с обязательным предварительным объявлением своего имени в разделе описаний.

Пример

```
unit Unit1;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
 Forms, Dialogs, StdCtrls, Buttons;
type
 TForm1 = class(TForm)
```

```
  BitBtn1: TBitBtn;
  procedure BitBtn1Click(Sender: TObject);
 private
   { Private declarations }
 public
   { Public declarations }
 end;
var
 Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.BitBtn1Click(Sender: TObject);
 begin
   Close;
 end;
end.
```

При описании процедуры к идентификатору имени могут добавляться **формальные** параметры-значения, заключенные в круглые скобки.

Формальные параметры обращения указываются в случае использования данных, передаваемых процедуре в качестве информационных установок для ее алгоритма действий, и представляют собой объявление типа данных.

Вызов процедуры, имеющей формальные параметры, осуществляется с указанием **фактических** параметров, соблюдая строгое соответствие порядка следования и заданного типа формальных параметров, т. е. по конкретным значениям задаваемых данных.

Пример

```
...
 // процедура рисования часов:
 procedure TForm1.Clock(clhour,clmin,clsec : TColor); { объявляем в
 имени формальные параметры: тип переменных, «ответственных»
 за цвет стрелок часов }
 begin
   with Form1.Canvas do
     begin
```

```
        // формальный параметр используется в коде алгоритма
        { при вызове процедуры Clock вместо него будет подставляться
значение соответствующего фактического параметра }
         Pen.Color:=clhour; { задаем цвет карандаша перед рисованием
часовой стрелки }
        MoveTo(cx,cy); // перемещаемся в центр циферблата
        Pen.Width:=3; // задаем толщину линий
        LineTo(cx+Round(hl*sin(ha)), // чертим линию
             cy-Round(hl*cos(ha)));
        MoveTo(cx,cy);
        Pen.Width:=2; // минутная стрелка
        LineTo(cx+Round(ml*sin(ma)),
             cy-Round(ml*cos(ma)));
        MoveTo(cx,cy);
        Pen.Color:=clSec; // секундная стрелка
        Pen.Width:=1;
        LineTo(cx+Round(sl*sin(sa)),
             cy-Round(sl*cos(sa)));
    end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
{ блок процедуры выполняется периодически с частотой свойства In-
terval таймера (задаем 1 раз в секунду) }
begin
  { вызываем процедуру Clock с указанием фактических параметров: }
  Clock(clWhite,clWhite,clWhite); // «стираем» стрелки, рисуя белым
  DecodeTime(Time,h,m,s,ms); { определяем следующее значение
времени}
  ha:=(h mod 12)/12*(2*Pi)+(m div 12)/60*(2*Pi);
  ma:=m/60*(2*Pi);
  sa:=s/60*(2*Pi);
  Clock(clBlue,clBlue,clSilver); { рисуем новое положение стрелок за-
данными цветами }
end;
...
```

Функция отличается от процедуры тем, что результат ее работы возвращается в виде конкретного типизированного значения этой функции (кроме файлового типа), которое можно использовать в операторах выражений Pascal наряду с другими операндами. Соответственно при описании функции обязательно указывается ее тип.

Формат заголовка:

Function <имя>(формальные параметры) **:** <тип результата>

Формальные параметры могут отсутствовать.

Delphi в своем составе имеет богатую библиотеку программных заготовок — ряд встроенных стандартных процедур и функций, а также среда допускает прямой вызов функций WinAPI. При их использовании не требуется предварительное описание, все необходимые параметры вызова определены в справочной литературе.

Индивидуализированные приложения, включающие нестандартные процедуры (пользовательские подпрограммы), программист разрабатывает самостоятельно.

Типы данных

Тип данных — множество допустимых значений данных (константы, переменные, значения функций, выражений), а также совокупность операций над этими данными.

*** Стандартные** типы данных (предопределенные) не требуют предварительного определения и действуют в любой точке программы или модуля.

*** Пользовательские** типы — это дополнительные абстрактные типы, характеристики которых программист задает самостоятельно.

Обрабатываемые в программе данные подразделяются на переменные и константы.

Простые типы данных

Простой тип определяет упорядоченное множество значений параметра.

Целые типы (стандартные порядковые)

Тип	Диапазон	Формат	Размер в байтах
Shortinit	–128 . . . +127	знаковый	1

Тип	Диапазон	Формат	Размер в байтах
SmallInt	–32 768 . . . +32 767	знаковый	2
Integer	–2147483648 . . . +2147483647	знаковый	4
LongWord	0 . . . 4 294 967 295	беззнаковый	4
Int64	$-9*10^{18}$ . . . $+9*10^{18}$	знаковый	8
Byte	0 . . 255	беззнаковый	1
Word	0 . . 65535	беззнаковый	2

Вещественный тип (стандартный)

Тип	Диапазон	Число значащих цифр
Real	$5{,}0*10^{-324}$ .. $1{,}7*10^{308}$	15 ... 16
Single	$1{,}5*10^{-45}$ .. $3{,}4*10^{38}$	7 ... 8
Double	$5{,}0*10^{-324}$ .. $1{,}7*10^{308}$	15 ... 16
Extended	$3{,}4*10^{-4951}$ .. $1{,}1*10^{4932}$	19 ... 20
Comp	$-2^{63}+1$ .. $2^{63}-1$	19 ... 20
Currency	±922 337 203 685 477,5807	19 ... 20

Логический тип (стандартный, порядковый)

Стандартный логический тип **Boolean** (размер 1 байт) представляет собой тип данных, любой элемент которых может принимать лишь два значения: True (истина) или False (ложь). 1 или 0 в машинном представлении. При этом справедливы следующие условия:

False < True ; Ord(False)=0 ; Ord(True)=1 ; Succ(False)=True ; Pred(True)=False

Символьный тип (стандартный, порядковый)

Стандартный символьный набор Char определяется множеством всех символов PC по кодировке Windows, использующей код ANSI (American National Standard Institute — Американский институт стандартизации).

Сравниваются величины символьного типа по своим кодам.

Перечисляемый тип (пользовательский, порядковый)

Перечисляемый тип определяется набором идентификаторов, с которыми могут совпадать значения параметра.

type

<имя типа> = (<идентификатор 1, идентификатор 2, ... , идентификатор n>);

Первому идентификатору присваивается порядковый номер 0, второму — 1 и т. д.

Функция Ord от величины перечисляемого типа дает порядковый номер ее значения.

ПРИМЕР:

```
type
  Operate = (Plus, Minus, Mult, Divide);
  Boolen = (False, True);
  Day = (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday);
```

К перечисляемым типам могут быть применены операции сравнения:

ПРИМЕР:

```
IF Day>Friday Then label1.Caption:= 'выходной день';
```

Перечисляемые типы могут использоваться в логических выражениях:

ПРИМЕР:

```
IF Month=May Then label1.Caption:= 'май';
```

К значениям перечисляемых типов не применимы ни арифметические операции, ни стандартные процедуры ввода-вывода.

Тип-диапазон (интервальный тип) — пользовательский, порядковый

Интервальный тип представляет собой диапазон (интервал) значений какого-либо порядкового типа, называемого базовым. Минимальное и максимальное значения интервала разделяются знаком **. .** (две точки).

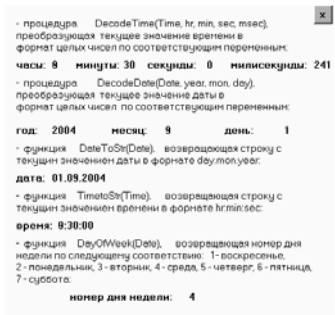
ПРИМЕР

```
type
  Month = 1 .. 12 {числа от 1 до 12}
```

Тип TDateTime — дата-время

Тип предназначается для хранения системных значений даты и времени. Функции Now, Date, Time возвращают текущие значения системной даты и времени.

- DecodeTime(Time, hr, min, sec, msec) процедура, преобразующая текущее значение времени в формат целых чисел по соответствующим переменным
- DecodeDate(Date, year, mon, day) процедура, преобразующая текущее значение даты в формат целых чисел по соответствующим переменным
- DateToStr(Date) — функция, возвращающая строку с текущим значением даты в формате day.mon.year
- TimeToStr(Time) — функция, возвращающая строку с текущим значением времени в формате hr:min:sec
- DayOfWeek(Date) — функция, возвращающая номер дня недели по следующему соответствию: 1 — воскресенье, 2 — понедельник, 3 — вторник, 4 — среда, 5 — четверг, 6 — пятница, 7 — суббота



Пример

Программа демонстрирует действие процедур и функций, оперирующих с системными значениями даты и времени.



Prim12
(файлы программы)

Листинг программы

```
unit Unit1;
interface
uses Windows, Messages, SysUtils, Variants, Classes, Graphics,
     Controls, Forms, Dialogs, StdCtrls;

type
 TForm1 = class(TForm)
   Label1: TLabel;
   Memo1: TMemo;
   Label2: TLabel;
```

```
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Button1: TButton;
    Label9: TLabel;
    Memo2: TMemo;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Memo3: TMemo;
    Label16: TLabel;
    Label17: TLabel;
    Memo4: TMemo;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Memo5: TMemo;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender: TObject);
```

```
var hr,min,sec,msec:word;
   year,mon,day:word;

begin
  DecodeTime(Time, hr, min, sec, msec);
    Label2.Caption:=IntToStr(hr);
    Label4.Caption:=IntToStr(min);
    Label6.Caption:=IntToStr(sec);
    Label8.Caption:=IntToStr(msec);

  DecodeDate(Date, year, mon, day);
    Label10.Caption:=IntToStr(year);
    Label12.Caption:=IntToStr(mon);
    Label14.Caption:=IntToStr(day);

  Label16.Caption:=DateToStr(Date);
  Label18.Caption:=TimeToStr(Time);
  Label20.Caption:=IntToStr(DayOfWeek(Date));

end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 Close;
end;

end.
```

Выражение

Выражение — это синтаксическая единица языка, определяющая способ вычисления некоторого значения выражения.

Выражение формируется из операндов (константы, переменные, функции и др.), знаков операций и действий над операндами, круглых скобок. Скобки служат для изменения порядка производимых операций, так как вычислительный процесс идет строго по принятым приоритетам.

Содержание

Введение **3**
Часть 1
Специализированные системы программирования для создания 32-разрядных Windows-приложений **5**
Глава 1. Программное обеспечение **5**
Архитектура Windows 2000 7
API — интерфейс прикладного программирования 10
Глава 2. Borland Delphi — система программирования Windows-приложений 11
Запуск среды программирования Delphi 11
Основы визуального программирования в среде Delphi 12
Часть 2
Среда программирования Delphi **17**
Глава 1. Интегрированная среда разработки Delphi 6 ..... **17**
Пиктографическое меню 17
Палитра компонентов 18
Инспектор объектов Object Inspector. Таблица свойств 24
Практика. Учебная программа «Запишем строчки» 25
Глава 2. Управление проектом **30**
Настройка среды программирования 30
Репозиторий Delphi 30
Справочная система Delphi 31
Менеджер проектов 33
Глава 3. Визуальные компоненты Delphi **34**
Объектно-ориентированное программирование (ООП). Определение понятий 34
Графический редактор Delphi 35
Работа с переключателями 38
Список выбора 41
Практика: «Школьник», «Расписание» 46
Контейнер для группы элементов 56
Компонент TTabControl. Блокнот с закладками 58
Таблица строк 60
Часть 3
Основные элементы Visual-программирования **63**
Глава 1. Object Pascal — алгоритмический язык Delphi **63**

Алфавит....63
Разделители....64
Идентификаторы....64
Структура программы (проект)....66
Модули....66
Вспомогательные алгоритмы. Процедуры и функции....68
Типы данных....72
Выражение....77
Описание операций....78
Константы....81
Переменные, присваивание....81
Структурные типы....83
Стандартные функции....87
Операторы Object Pascal....89
Структурированные операторы....89
Практика....92
Глава 2. ООП — новая концепция языка программирования....98
Принципы ООП....98
Событие Windows....99
Интерфейс диалога в Delphi (стандарт API)....101
Практика: Учебная программа «Угадай число»....103
Отладка программ....106
Глава 3. Проектирование многооконных приложений....108
Организация взаимодействия форм....108
Размещение на форме картинок. Компонент TImage....111
Элемент пользовательского интерфейса — меню приложения....114
Всплывающая подсказка....116
Практика: «Кроссворд», игра «Назови номер строки с числом»....118
Согласование действий управляющих элементов....134
Страница Dialogs. Типовые диалоги Windows....138
Практика. Учебная программа «Простой редактор». Печать текстового файла....141
Глава 4. Работа с графикой и средствами мультимедиа....149
Реализация мультимедийных технологий....149
Стандартные классы Delphi для использования графических средств....155
Класс TPen — перо....165
Класс TBrush — кисть....166
Regions — области. Непрямоугольные окна и кнопки....174

Практика. Пример тестовой оболочки 187
Методическое планирование курса 201
Приложение. Delphi 8.0 — обзор IDE 208
Технология .Net.....208
Среда программирования Delphi 8.....209
Интегрированная среда разработки Delphi 8.0.....209
Создание проекта.....211
Список литературы 216