Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования

Владимирский государственный университет

Кафедра конструирования и технологии радиоэлектронных средств

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ ПО ДИСЦИПЛИНЕ “СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ”

Составитель
Д.П. АНДРИАНОВ

Владимир 2010

УДК 681.3.06
ББК 32.973.26-018.1
М54

Рецензент
Кандидат технических наук
доцент кафедры конструирования и технологии
радиоэлектронных средств
Владимирского государственного университета
В.В. Евграфов

Печатается по решению редакционного совета
Владимирского государственного университета

М54 **Методические** указания к лабораторным работам по дисциплине “Системное программирование ” / Владим. гос. ун-т ; сост. Д. П. Андрианов. – Владимир : Изд-во Владим. гос. ун-та, 2010. – 28 с.

Составлены в соответствии с программой курса “Системное программирование” в части, касающейся вопросов технологий программирования. Содержат краткие теоретические сведения, примеры реализации процедур, рекомендации по составлению исходных текстов по трем лабораторным работам.

Предназначены для студентов специальности 210202 – проектирование и технология электронно-вычислительных средств дневной и контрактно-заочной форм обучения.

Ил. 2. Библиогр.: 3 назв.

УДК 681.3.06
ББК 32.973.26-018.1

ВВЕДЕНИЕ

Методические указания к лабораторным работам по дисциплине “Системное программирование” содержат краткие теоретические сведения, примеры реализации процедур, рекомендации по составлению исходных текстов программ. Представленная серия лабораторных работ направлена на изучение современных технологий программирования: использование динамических библиотек, визуальных компонентов и механизма автоматизации OLE. Практика программирования должна развивать у студентов навыки решения конкретных задач, понимание того, как решить задачу и зачем делать какие-либо определенные действия.

В первой лабораторной работе рассматриваются вопросы создания динамических библиотек (DLL). Общеизвестно, что динамические библиотеки играют важную роль в функционировании как операционной системы, так и прикладных программ. Приводятся примеры статического и динамического связывания DLL, а также вызова окна из библиотеки.

Вторая лабораторная работа посвящена изучению технологий использования визуальных компонентов в условиях ограничений со стороны вычислительных комплексов.

В третьей лабораторной работе рассматриваются технологии программирования, основанные на использовании связывания и внедрения объектов OLE.

Лабораторные работы позволяют закрепить теоретические знания, получить практические навыки по использованию современных технологий разработки программного обеспечения.

Лабораторные работы рекомендуется проводить в среде ускоренного программирования Delphi, версии 5 – 7 или 2005. Для формирования интерфейса приложений рекомендуется использовать наиболее общеупотребительные визуальные компоненты, аналоги которых имеются во всех продуктах RAD, например Visual Basic, C++ Bilder и т.д.

Лабораторная работа № 1

ИСПОЛЬЗОВАНИЕ В ПРИЛОЖЕНИЯХ ДИНАМИЧЕСКИХ БИБЛИОТЕК

Цель работы: научиться разрабатывать и использовать в приложениях динамические библиотеки.

1. Основные положения

Динамически присоединяемая библиотека DLL – это специального вида исполняемый файл с расширением .dll, используемый для хранения функций и ресурсов отдельно от исполняемого файла.

Создание DLL повышает гибкость программы, функции и ресурсы загруженной DLL доступны для любой программы, динамические библиотеки могут использоваться приложениями, написанными на других алгоритмических языках.

Библиотеки DLL могут связываться с приложением двумя путями:

- статически: DLL загружается сразу, как только начинает выполняться приложение;
- динамически: DLL загружается в тот момент, когда необходимо выполнить какую-то хранящуюся в ней функцию.

1.1. Создание DLL

В хранилище (репозитарии) Delphi имеется заготовка библиотечного модуля (команда основного меню File > New > DLL). Для создания динамической библиотеки достаточно оставить только три оператора из имеющейся заготовки:

```
library Project1;          //объявление имени модуля библиотеки

uses SysUtils, Classes; //используемые стандартные библиотеки

end.                       //конец модуля библиотеки
```

(Из текста заготовки удалены комментарий, директива обращения к файлу ресурсов {$R *.RES} и оператор begin начала блока, выполняемого в момент загрузки DLL).

В качестве примера создадим библиотеку с функцией перевода трехбитового числа (диапазон изменений от 0 до 7) в его бинарное символьное представление (диапазон изменений соответственно '000'..'111').

```
library convert;

uses SysUtils, Classes;

var Sha: array [0..7] of string =
('00000000','00000001','00000010','00000011','00000100','00000101',
'00000110','00000111');

function ByteToBin(n: Byte): PChar; stdcall;
begin
  ByteToBin:=PChar(sha[n]);
end;

exports ByteToBin;

end.
```

Соглашения о вызовах определяют порядок следования параметров в стеке. Спецификатор stdcall (указывается после заголовков экспортируемых из DLL функций) задает правила передачи параметров (в последовательности справа налево.) Подобный вызов обеспечивает обработку фиксированного числа параметров.

Возможно использование спецификатора register (только для приложений, написанных на Pascal!) – параметры помещаются в стек слева направо.

Предложение exports указывает экспортируемые процедуры и функции, т.е. вызываемые из внешнего приложения. В предложении exports возможно объявление индексов.

Трансляция модуля с исходным текстом динамической библиотеки сопровождается появлением диагностического окна с сообщением: "Cannot debug project unless a host application is defined. Use the Run\Parameters...
dialog box".

1.2. Вызов DLL из текста приложения

1.2.1. Статическое связывание

В качестве примера статического связывания рассмотрим приложение с элементами интерфейса: однострочное поле редактирования TEdit, кнопка управления TSpeedButton и поле со спаренными кнопками TSpinEdit.

В исполняемом разделе после директивы обращения к файлу описания формы {$R *.DFM} следует перечень функций, вызываемых из DLL. Каждое объявление завершается спецификатором external с указанием имени вызываемой DLL.

```
unit Unit1;

interface

uses
  Windows, SysUtils, Classes, Forms, Spin, Buttons, StdCtrls, Controls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    SpeedButton1: TSpeedButton;
    SpinEdit1: TSpinEdit;
    procedure SpeedButton1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

function ByteToBin(n: Byte): PChar; stdcall; external 'convert.dll';

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
    Edit1.Text:=ByteToBin(SpinEdit1.Value);
end;

end.
```

1.2.2. Динамическое связывание

Динамическое связывание проводится в три этапа:
1) загрузка библиотеки функцией LoadLibrary API Windows и получение указателя на вызываемую функцию с помощью GetProcAddress. Предварительно необходимо объявить в глобальных переменных модуля указатель (дескриптор) H: THandle;
2) собственно вызов необходимой функции;
3) выгрузка библиотеки из памяти с помощью FreeLibrary (функция FreeLibrary уменьшает на 1 число ссылок на загруженную библиотеку, т.е. для полной очистки памяти необходимо вызывать FreeLibrary столько раз, сколько раз была загружена библиотека!).

Рассмотренный выше пример вызова функции из динамической библиотеки при динамическом связывании будет выглядеть следующим образом:

```
unit Unit1;

interface

uses
```

```
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Di-
alogs,
  Spin, Buttons, StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    SpeedButton1: TSpeedButton;
    SpinEdit1: TSpinEdit;
    SpeedButton2: TSpeedButton;
    procedure SpeedButton1Click(Sender: TObject);  //вызов функции
    procedure SpeedButton2Click(Sender: TObject);  //загрузка библиоте-
ки
    procedure FormDestroy(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  H: THandle = 0;
  ByteToBin: function(n: Byte): PChar; stdcall;

implementation

{$R *.DFM}

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
    Edit1.Text:=ByteToBin(SpinEdit1.Value);
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
    H := LoadLibrary('convert.dll');
```

```
    if H <>0 then ByteToBin := GetProcAddress(H,'ByteToBin')
       //if H <>0 then @ByteToBin := GetProcAddress(H,'ByteToBin')
       //использование адреса функции в памяти
    else ShowMessage('Нет загрузки convert.dll');
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
     while FreeLibrary(H) do;  //выгрузка с помощью оператора цикла
                                    "до тех
                                  //пор, пока H не очистится"
     //   if H <>0 then FreeLibrary(H); //простая выгрузка с предвари-
                                            тельной
                                          //проверкой состояния указателя
end;

end.
```

1.2.3. Загрузка DLL-библиотеки, содержащей форму

С помощью динамической библиотеки можно вызывать формы из связанных с ней модулей. В тексте библиотеки делается ссылка на вызываемую форму, в экспортируемых функциях (процедурах) создается экземпляр формы, после окончания работы с формой следует уничтожить форму, т.е. освободить память компьютера от ненужного объекта.

Пример DLL-библиотеки, организующей ввод двух чисел и вычисление их суммы:

```
library newDLL;
uses
  Forms, SysUtils, Classes, Dialogs,
  UDLL in 'UDLL.pas' {frmDLL};
function Action01: ShortString;      //String;
var Form: TfrmDLL;
begin
   Form := TfrmDLL.Create(Application);
   Form.ShowModal;
```

```
  with Form do
  try
    Result:=IntToStr(StrToInt(Edit1.Text)+StrToInt(Edit2.Text));
    except
      on EConvertError do
        ShowMessage('Неправильно набрано число!');
    end;
  Form.Free;
end;
exports Action01;
end.
```

Текст модуля, содержащего форму

```
unit UDLL;
interface
uses
  Windows, Classes, Controls, Forms, ExtCtrls, StdCtrls;
type
  TfrmDLL = class(TForm)
    btnEnter: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmDLL: TfrmDLL;
implementation
{$R *.DFM}
procedure ShowMyForm(AOwner: TComponent);
var MyForm: TfrmDLL;
begin
   MyForm:= TfrmDLL.Create(AOwner);
   MyForm.ShowModal;
```

```
    MyForm.Free;
  end;
  exports ShowMyForm;
  end.
```

Текст программы, вызывающей динамическую библиотеку:

```
program DynamicDLL;
uses
  Windows, Forms, SysUtils, Dialogs;
var H: THandle;
    Action01: function: ShortString;
begin
  H:=LoadLibrary('newDLL.dll');
  if H<>0 then
  begin
    Action01:=GetProcAddress(H,'Action01');
    ShowMessage(Action01);
    FreeLibrary(H);
  end
  else ShowMessage('Нет загрузки
newDLL.dll');
end.
```

Результат запуска программы – окно ввода данных – показан на рис. 1.

Рис. 1. Окно, генерируемое в результате вызова DLL

2. Выполнение работы

2.1. Задание на подготовку к работе

1. На своем профиле создать новую папку для хранения разрабатываемого приложения.
2. Запустить Delphi. Среда программирования создаст «пустой» проект.
3. Записать «пустой» проект в папку на своем профиле (пункт основного меню File->Save Project as).

2.2. Лабораторное задание

Задача лабораторной работы – в среде Delphi разработать динамическую библиотеку, генерирующую окно с набором кнопок:

1) настройки типа линий для графического редактора;
2) настройки заливки фигур графического редактора;
3) выбора варианта фигур графического редактора;
4) для редактирования списка строк;
5) выбора из коллекции иконок;
6) выбора ответа в окне тестирования.

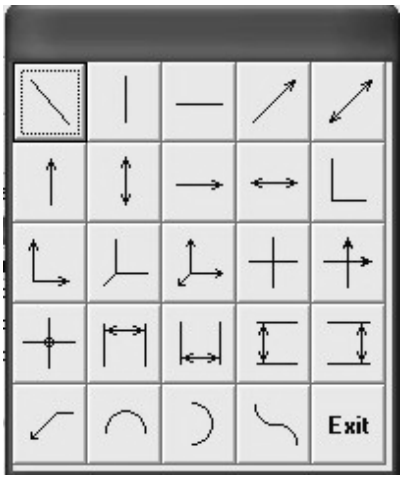

Рис. 2 Пример окна с набором кнопок, генерируемого DLL

3. Рекомендации при разработке приложения

3.1. В качестве элементов интерфейса используйте визуальные компоненты TMemo, TEdit, TButton, TRadioGroup, TMainMenu, TSpinEdit. (приведены названия классов среды программирования Delphi).

3.2. Пример окна с набором кнопок для выбора варианта фигур представлен на рис. 2.

4. Содержание отчета

Отчет оформляют в форме Zip-архива, содержащего все файлы проекта (Project1.cfg, Project1.dof, Project1.dpr, Project1.exe, Project1.res, Unit1.dcu, Unit1.dfm, Unit1.pas). Zip-архив именуется по фамилии студента, номеру группы и названию лабораторной работы.

5. Контрольные вопросы

1. Что такое модальная форма?
2. Чем отличается метод Free от метода Destroy?
3. Обязательно ли прописывать вызов стандартных библиотек в модуле DLL?

4. Для чего используют индексы в декларации exports?
5. Для чего используют оператор try?
6. Чем отличается тип Pchar от string?
7. Возможен ли вызов DLL в методе FormCreate?
8. Для чего нужен дескриптор динамической библиотеки?
9. В каких случаях при вызове DLL можно не указывать полный путь?
10. Как вызвать функцию по ее адресу?

Лабораторная работа № 2

ИСПОЛЬЗОВАНИЕ В ПРИЛОЖЕНИЯХ НЕСТАНДАРТНЫХ КОМПОНЕНТОВ БЕЗ ИХ РЕГИСТРАЦИИ

Цель работы: научиться использовать в приложениях незарегистрированные визуальные компоненты.

1. Основные положения

Среда Delphi предоставляет в распоряжение программиста палитру библиотек компонентов, разработанных в компании «Borland». Использование "чужих" разработок, в том числе компонента программиста, подразумевает их регистрацию в среде программирования с размещением ссылок на компонент в палитре. На практике подобный подход часто оказывается неэффективным. Работа на сетевых компьютерах с ограничением прав доступа, политика безопасности не дают в полной мере использовать все возможности технологии RAD. Оказавшись в "ущемленном" положении, малоопытные программисты забывают или вовсе не знают об отладочном этапе разработки компонента, когда его тестирование, т.е. проверка работоспособности, проводится до его регистрации. Рассматриваемый подход имеет существенный недостаток: работа с визуальным компонентом ведется "вслепую", без его отражения на форме и в инспекторе объектов (после вызова конструктора Delphi во всплывающем окне показывает свойства и методы компонента). Все свойства, значения которых отличаются от принятых по умолчанию, и отклики на события программист должен прописать вручную, без помощи инспектора объектов.

Для визуализации незарегистрированного компонента на форме работающего приложения необходимо:

1) подключить модуль компонента (достаточно иметь файл с расширением *.dcu) в декларации Uses;

2) объявить экземпляр компонента;

3) вызвать конструктор компонента;

4) указать владельца компонента;

5) указать координаты расположения компонента.

Например, для нестандартного компонента TZDiagram, расположенного в модуле ZDiagram, достаточно указать в переменных ссылку на экземпляр объекта:

```
var
  Form1: TForm1;
  Zdiagram1: TZDiagram;
```

а в событии onCreate формы прописать действия:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Zdiagram1 := TZDiagram.Create(Self);        //вызов конструктора
  with Zdiagram1 do
  begin
    parent:=self;          // владелец компонента, окно программы
    left:=10;              //левая координата
    top:=80;               //верхняя координата
  end;
end;
```

В случае если необходимо создать несколько экземпляров (countA) одного класса, используется динамический список Tlist:

```
const countA = 5;                    //количество экземпляров компо-
                                       нента
var
  Form1: TForm1;
  List: TList;                      //динамический список объектов
  Zdiagram1: TZDiagram;             //рабочий экземпляр объекта
```

```
procedure TForm1.FormCreate(Sender: TObject);
const xx: array[1..countA] of integer = (5,5,5,275,275);
      yy: array[1..countA] of integer = (5,210,420,5,210);
var n: byte;
begin
  List := TList.Create;                 //создаем список
  for n := 1 to countA do               //заполняем список в цикле
  begin
    List.Add(TZDiagram1.Create(self));
    ZDiagram1:= List.Items[n-1];
    ZDiagram1.Left : =xx[n];
    ZDiagram1.Top : =yy[n];
    ZDiagram1.parent : =self;
    List.Items[n-1] := ZDiagram1;
  end;
```

2. Выполнение работы

2.1. Задание на подготовку к работе

1. На своем профиле создать новую папку для хранения разрабатываемого приложения.
2. Запустить Delphi. Среда программирования создаст «пустой» проект.
3. Записать «пустой» проект в папку на своем профиле (пункт основного меню File->Save Project as).

2.2. Лабораторное задание

Задача лабораторной работы – разработать приложение Delphi, в котором проводится презентация незарегистрированного нестандартного компонента. В качестве исходных данных можно использовать компоненты библиотеки ExpressForumLibrary, распространяемой бесплатно и с исходными текстами.

Указанная библиотека включает в себя 15 визуальных компонентов:

dfxLabel

dfxColorButton

dfxPictureButton
dfxOutlookBar
dfxBackGround
dfxDesigner
dfxExpressionExplorer
dfxProgressBar
dfxGoupBox
dfxCheckBox
dfxClock
dfxTimer
dfxConnector
dfxQuickTyper
dfxShapedForm

3. Рекомендации при разработке приложения

3.1. В качестве элементов интерфейса используйте визуальные компоненты TMemo, TEdit, TButton, TRadioGroup, TMainMenu, TSpinEdit, TTimer (приведены названия классов среды программирования Delphi).

3.2. Если свойства компонента имеют именованные константы, организуйте демонстрацию вариантов настройки компонента.

4. Содержание отчета

Отчет оформляют в форме Zip-архива, содержащего все файлы проекта (Project1.cfg, Project1.dof, Project1.dpr, Project1.exe, Project1.res, Unit1.dcu, Unit1.dfm, Unit1.pas). Zip-архив именуется по фамилии студента, номеру группы и названию лабораторной работы.

5. Контрольные вопросы

1. Что означает идентификатор self?
2. Как организовать отклик на событие onClose компонента?
3. Можно ли добавлять новые свойства (поля) к незарегистрированному компоненту?
4. Как прописать несколько экземпляров незарегистрированного компонента?

5. Можно ли присваивать значения свойств компонента до вызова его конструктора?
6. Где в исходном тексте приложения следует объявлять родителя незарегистрированного компонента?
7. Можно ли объявлять незарегистрированный компонент родителем для других компонентов?
8. Можно ли управлять доступностью (enabled) незарегистрированного компонента?
9. Будет ли доступен незарегистрированный компонент во время работы приложения, если не был указан его родитель?
10. Будет ли доступен незарегистрированный компонент во время работы приложения, если не были указаны координаты его расположения (top, left)?

Лабораторная работа № 3

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ, ОСНОВАННЫЕ НА ИСПОЛЬЗОВАНИИ СВЯЗЫВАНИЯ И ВНЕДРЕНИЯ ОБЪЕКТОВ OLE

Цель работы: изучить возможности интегрирования средств приложений Microsoft Office в проекты среды Delphi.

1. Основные положения

1.1. Создание OLE-объекта

Приложения MS Office представляют собой объекты-серверы, которые могут управляться внешними программами, поддерживающими интерфейс COM.

С точки зрения объектной модели для MS Word и MS Excel корневым объектом является Application, через него возможен доступ к выделенному объекту Selection, коллекции открытых документов Documents или открытых рабочих книг WorkBooks и коллекции элементов управления, диалогов, свойств приложений Word или Excel.

Доступ к OLE-объекту реализуется с помощью функции CreateOleObject, при этом следует выполнить последовательность действий:

0) присоединить библиотеку ComObj (оператор Uses);
1) объявить переменную для хранения ссылки на объект;
2) создать OLE-объект;
3) сделать OLE-объект видимым;
4) создать экземпляр документа;
5) открыть документ.

Пример процедуры, реализующей действия пунктов 1 – 5 при программной организации работы с редактором Word:

```
procedure TForm1.word1Click(Sender: TObject);
var Path: string;
W: variant;
begin
   W:=CreateOleObject('Word.Application');  //создание OLE-объекта
   W.Visible:=not W.Visible;                 //управление видимостью OLE-
                                               объекта
   W.Documents.Add;                        //создание экземпляра документа
   GetDir(0,Path);                           //определение места старта про-
                                               граммы
   If not OpenDialog1.Execute then Exit;    //организация диалога поиска
                                               файла
   ChDir(Path);                              //восстановление места стар-
                                               та программы
   W.Documents.Open(OpenDialog1.FileName);  //открытие файла в ре-
                                                 дакторе.
end;
```

Для запуска электронных таблиц Exsel из Delphi-приложения достаточно следующей процедуры:

```
procedure TForm1.exsel1Click(Sender: TObject);
var Path: string;
    W: variant;
begin
```

```
    W:=CreateOleObject('Excel.Application');  //создание OLE-объекта
    W.Visible:=not W.Visible;
    W.WorkBooks.Add;                          //создание экземпляра книги
    GetDir(0,Path);
    if not OpenDialog1.Execute then Exit;
    ChDir(Path);
    W.WorkBooks.Open(OpenDialog1.FileName);   //открытие книги
end;
```

Метод **Open** имеет следующие основные аргументы:

- FileName: string – путь и имя файла;
- ReadOnly: boolean – режим "только для чтения"
- PasswordDocument: string – пароль для открытия;
- Format – формат (расширение файла) открываемого документа (в скобках указано значение именованной константы):

0 (wdOpenFormatAuto) - автоматически
1 (wdOpenFormatDocument) - .DOC
2 (wdOpenFormatRTF) - .RTF
3 (wdOpenFormatTemplate) - .DOT
4 (wdOpenFormatText) - .TXT
5 (wdOpenFormatUnicodeText) - .TXT (в формате UNICODE)

Пример использования метода Open:

```
W.Document.Open(FileName := 'c:\Док1.doc', ReadOnly:=true, PasswordDocument:='123', Format:=wdOpenFormatText);
```

Возможна работа со списком открытых документов с помощью методов

Documents.Item для редактора Word:

Например, выбор документа из коллекции по номеру number будет таким:

```
W.Documents.Item(number).Activate,
```

а выбор с выделением документа из коллекции по имени FileName:

```
W.Documents.Item(FileName).Select;
```

WorkBooks.Item для электронных таблиц Excel.

1.2. Редактирование документа Word

Вставка текста в документ возможна двумя способами:

- в начало содержимого объекта **Range** – InsertBefore;
- в конец содержимого объекта **Range** – InsertAfter.

Например, вставка в документ содержимого компонента Memo1:

```
W.ActiveDocument.Range.InsertAfter(Memo1.Text);
```

Возможна работа с фрагментом документа (например с 10 по 20 строки), при этом необходимо создать новый объект:

```
var NewRange: variant;
begin
  NewRange:=W.ActiveDocument.Range(10,20);
  NewRange.InsertAfter(Memo1.Text);
end;
```

Для чтения текста из документа используется свойство Range.Text.

Сохранение документа можно обеспечить вызовом методов:

- Save объекта-документа, например W.ActveDocument.Save;
- SaveAs (синтаксис аналогичен методу Open):

```
W.ActveDocument.SaveAs(FileName:='c:\Doc1.doc');
```

Закрытие приложения Word:

```
- W.Application.Documents.Close;          //закрытие всех открытых документов
- W.ActiveDocument.Close;                 //закрытие документов коллекции
- W.ActiveDocument.Close(true);           //сохранение и закрытие документа
- W.Quit; W:=UnAssigned;                  //закрытие приложения и очистка памяти
```

Корректное завершение работы с приложением Word должно обеспечить:

- закрытие документов;

- закрытие приложения;
- очистку памяти.

Объект **Selection** ассоциируется с выделенным объектом (фрагмент текста, ячейка таблиц, надписи, рисунки и т.д.).

Выделение фрагмента текста в документе возможно следующим образом:

```
 var NewRange: variant;
    begs, ends: integer;
 begin
   NewRange:=W.ActiveDocument.Range(begs,ends);
   NewRange.Select;
 end;
```

или

```
 begin
   W.ActiveDocument.Range(begs,ends).Select;
 end;
```

Объект **Selection** имеет набор атрибутов:

<u>свойства:</u>

- Text – текстовое содержимое;
- Start – начальная позиция выделенного объекта;
- End – конечная позиция выделенного объекта;
- Style – стиль выделенного текста;
- StoryType – тип выделяемого объекта (комментарий, заголовок, текст и т.д.);
- Type – тип выделенного объекта.

<u>коллекции:</u>

- Characters – символы выделенного объекта;
- Words – слова выделенного объекта.

<u>объекты:</u>

- Find – поиск, поиск и замена в документе;
- Font – шрифт выделенного объекта.

<u>методы:</u>

- Copy – копирование текста в буфер обмена;
- Paste – вставка текста из буфера обмена;
- Cut – вырезание выделенного фрагмента текста;

- Delete – удаление выделенного фрагмента текста;
- Delete(a,b) – удаление фрагмента из b символов начиная с позиции a;
- SetRange(a,b) – выделение текста между позициями a и b;
- Move(Unit, Count) – перемещение объекта Selection в документе в зависимости от аргумента Unit (указаны именованные константы и числовые значения), выполняется переход к следующему:

- символу	wdCharacter=1;
- слову	wdWord=2;
- предложению	wdSentence=3;
- абзацу	wdParagraph=4;
- разделу	wdSection=8;
- текстовой области	wdStory=6;
- ячейке	wdCell=12;
- столбцу	wdColumn=9;
- строке	wdRow=10;
- таблице	wdTable=15;
- линии	wdLine=5;

- InsertAfter – вставка текста после объекта;
- InsertBefore – вставка текста до объекта;
- TypeText – вставка текста на место объекта или с позиции курсора
- ConvertToTable – преобразование выделенного текста в таблицу;
- CopyAsPicture – копирование с преобразованием в формате BMP.

1.3. Работа с электронными таблицами Excel

Рабочая книга **WorkBook** содержит коллекцию листов **Sheets**, в которых хранится информация. Коллекция листов **Sheets** имеет свойства:

- Count – количество элементов коллекции;
- Item(i: integer) – набор объектов.

методы:

- Select – выделение листов рабочей книги;
- Copy – копирование листов в новую рабочую книгу;
- PrintOut – вывод на печать;
- Add – добавление нового листа в рабочую книгу.

Доступ к листу рабочей книги осуществляется через список:

```
var Sheet: variant;
    number: integer;
```

```
begin
  Sheet:=Sheets.Item(number);
end;
```

Методы Copy и Add могут вызываться с уточнением after или before – после или до оригинала:

```
Sheet.Copy(before:=Sheet);
```

Программное удаление листа рабочей книги возможно только после подавления сообщения приложения:

```
begin
  W.DisplayAfters:=false;   //подавление сообщений
  Sheet.Delete;             //удаление листа
  W.DisplayAfters:=true;    //восстановление контроля
end;
```

Лист рабочей книги Excel состоит из ячеек – прямоугольных областей с определенными координатами в границах листа. Для доступа к свойствам и содержимому ячеек используются объекты **Range** и **Cells**.

Для **Range** аргумент – строка адреса, для **Cells** – номера строки и столбца:

```
NewRange:=W.ActiveSheet.Range['B2'];
NewRange:=W.ActiveSheet.Cells[2,2];
```

Объект **Range** имеет:

<u>свойства:</u>

value - значение ячейки;
text - текст ячейки (используется, если формат значения неизвестен);
formula – формула для расчета значений ячеек области;
left – левая координата области;
top – верхняя координата области;
height – высота области;
width – ширина области;
style – стиль текста области;
font – шрифт текста области;

cells – ячейки области;
count – количество ячеек области;
item – ссылка на элемент коллекции;
columns – коллекция столбцов области;
rows – коллекция строк области.

методы:

clear – очистка содержимого и формата ячеек области;
copy – копирование области в буфер;
cut – перенос области в буфер;
delete – удаление области из документа;
find – поиск данных в области;
insert – вставка ячеек;
replace – поиск и замена данных;
select – выделение содержимого ячеек;
sort – сортировка содержимого ячеек;
merge – объединение ячеек;
justify – равномерное заполнение текстом ячеек;
activate – активизация объекта;
calculate – вычисление результата математического выражения.

2. Выполнение работы

2.1. Задание на подготовку к работе

1. На своем профиле создать новую папку для хранения разрабатываемого приложения.
2. Запустить Delphi. Среда программирования создаст «пустой» проект.
3. Записать «пустой» проект в папку на своем профиле (пункт основного меню File->Save Project as).

2.2. Лабораторное задание

Задача лабораторной работы – разработать приложение Delphi, в котором:

1) через редактор Word генерируются текстовые документы (например уведомления, поздравления, письма);
2) через электронные таблицы Excel выводятся результаты расчетов (например математических функций, данные из файлов).

3. Рекомендации при разработке приложения

3.1. В качестве элементов интерфейса используйте визуальные компоненты TMemo, TEdit, TButton, TRadioGroup, TMainMenu, TOpenDialog, TSaveDialog (приведены названия классов среды программирования Delphi).

3.2. При разработке приложения, генерирующего текстовые документы, шаблон документа вводите из текстового файла или файла инициализации.

3.3. При разработке приложения, выводящего результаты расчета, шаблон таблицы вводите из файла инициализации.

4. Содержание отчета

Отчет оформляют в форме Zip-архива, содержащего все файлы проекта (Project1.cfg, Project1.dof, Project1.dpr, Project1.exe, Project1.res, Unit1.dcu, Unit1.dfm, Unit1.pas). Zip-архив именуется по фамилии студента, номеру группы и названию лабораторной работы.

5. Контрольные вопросы

1. Как запрограммировать запись текста в документ Word?
2. Как запрограммировать изменение шрифта текста в документе Word?
3. Как запрограммировать копирование содержимого документа Word в компонент список строк TStringList?
4. Как запрограммировать выделение блока строк в документе Word?
5. Нужно ли открывать рабочую книгу WorkBook при создании таблицы в документе Word?
6. Возможно ли программирование прорисовки иконок в ячейках электронных таблиц?
7. Как запрограммировать запись текста в ячейку электронной таблицы?
8. Каким методом коллекции WorkBooks можно создать новую рабочую книгу?
9. Как запрограммировать запись в ячейку Excel значения в числовом формате?
10. Как запрограммировать доступ к рабочей книге Excel?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Архангельский, А. Я.* Приемы программирования в Delphi / А. Я. Архангельский. – М. : Бином-Пресс, 2003. – 784 с. – ISBN 5-9518-0020-X.

2. *Фленов, М. Е.* Программирование в Delphi глазами хакера / М. Е. Фленов. – СПб. : БХВ-Петербург, 2004. – 368 с. – ISBN 5-94157-351-0.

3. *Корняков, В. Н.* Программирование документов и приложений MS Office в Delphi / В. Н. Корняков. – СПб. : БХВ-Петербург, 2005. – 496 с. – ISBN 5-94157-535-1.

ОГЛАВЛЕНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ
ПО ДИСЦИПЛИНЕ
“СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ ”

Составитель
АНДРИАНОВ Дмитрий Петрович

Ответственный за выпуск – зав. кафедрой профессор В.П. Крылов

Подписано в печать 15.03.10
Формат 60x84/16. Усл. печ. л. 1,63. Тираж 100 экз.
Заказ
Издательство
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.