Л.А. Соловцова, Т.А. Галаган

ПРОГРАММИРОВАНИЕ В DELPHI

Лабораторный практикум

Министерство образования Российской Федерации
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Л.А. Соловцова , Т.А. Галаган

ПРОГРАММИРОВАНИЕ В DELPHI

Лабораторный практикум

Благовещенск
2003

32.973.26-18.2
С60

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного университета

Л.А. Соловцова, Т.А.Галаган
Программирование в Delphi:
Лабораторный практикум /Амурский гос. ун-т, Благовещенск, 2003. -39с.

Практикум содержит 5 лабораторных работ, которые посвящены вопросам создания приложений с использованием интегрированной среды разработки Delphi.

Лабораторный практикум рекомендуется использовать для студентов специальности 0719 «Информационные системы в технике и технологиях» и 2202 «Автоматизированные системы обработки информации и управления» при проведении учебной практики.

Рецензенты: Е.К. Ясевич, начальник информационного отдела областной избирательной комиссии.

ВВЕДЕНИЕ

Delphi одна из самых популярных систем визуального программирования. Как любая подобная система, Delphi предназначена для разработки программ и имеет две характерные особенности: создаваемые ею программы могут работать не только под управлением Windows, а сама она относится к классу инструментальных средств ускоренной разработки программ.

Визуальное конструирование форм избавляет программиста от многих аспектов разработки интерфейса программы, библиотека визуальных компонентов предоставляет программисту огромное разнообразие уже созданных программных заготовок, которые немедленно или после несложной настройки готовы к работе в рамках создаваемой программы. Все это избавляет программиста от рутинной, монотонной работы.

Лабораторный практикум посвящен основам разработки приложений в этой популярной среде. Первая лабораторная работа знакомит с интегрированной средой разработки приложений. Здесь кратко рассматриваются основные составные части Delphi и создаваемого проекта, вопросы управления проектом, создания интерфейса проекта, написания обработчиков событий.

Вторая лабораторная работа посвящена изучению основных визуальных компонент и методам работы с ними.

Изучив третью лабораторную работу студенты должны научиться обрабатывать ошибки в своих программах, используя механизм исключений, ознакомиться с основными классами исключений, создавать собственные классы исключений, строить последовательность обработки исключений.

Четвертая лабораторная знакомит с понятиями процесс и поток, методами построения программы с несколькими потоками.

В пятой лабораторной рассматриваются вопросы создания и использования DLL при разработке приложений.

Практикум содержит множество примеров, варианты заданий, контрольные вопросы, что даст возможность студентам освоить азы создания приложений в Delphi.

Лабораторная работа №1

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ ПРОГРАММ DELPHI

Delphi – среда разработки программ, ориентированных на работу в Windows. В качестве языка программирования в Delphi используется объектно-ориентированный язык Objekt Paskal. В основе идеологии Delphi лежат технология визуального проектирования и событийного программирования, применение которых позволяет существенно сократить время разработки и облегчить процесс создания приложений, программ работающих в среде Windows.

На экране после запуска Delphi появляется три окна: главное, окно формы (Form1), окно инспектора объектов (Object Inspector). В главном окне находится меню команд, панель инструментов и палитра компонентов. Окно формы представляет собой заготовку – макет разрабатываемого приложения. Окно инспектора объектов позволяет видеть и менять характеристики (свойства объектов проекта). После запуска Delphi в этом окне находятся свойства формы Form1.

Вопрос организации и управления проектами – вопрос предшествующий организации проекта Delphi. Чтобы не потерять файлы, не выполнить запись поверх файлов, необходимых для работы проекта, знать какие файлы соответствуют каким проектам, необходимо для проекта создать каталог под катологом компилятора. Преимуществом такого подхода является то, что всегда известно где находятся модифицируемые файлы и исполняемые модули, не создает трудностей создание резервных копий для проектов. Чтобы работа над проектом была организованной не рекомендуется использовать имена по умолчанию, а давать формам, компонентам, переменным уникальные имена.

Для организации проекта необходимо выполнить следующие действия.

1. Создать каталог для своих проектов под каталогом,в котором находится Delphi .
2. Создать каталог для проекта.
3. Создать новый проект, используя команду меню File| New (Файл| Новый), выбрать пиктограмму Application (Приложение) из меню New Item(новый элемент), OK.
4. Сохранить проект используя команду File| Save| Project AS. Выбрать каталог для сохранения программного модудя Unit1.pas, заменить unit1 на новое имя. При сохранении

файла проекта Projekt1.dpr дать проекту новое, уникальное имя.

Стартовая форма создается путем изменения свойств (характеристик) формы Form1, которые определяют ее внешний вид. Они перечислены на вкладке (странице) Properties (свойства) окна Objekt Inspector. В левой колонке перечислены названия свойств, в правой – их значения. Свойства, присущие большинству компонент Delphi, представленны в приложении 1.

Для организации проекта необходимо изменить название формы, т.е. изменить свойство Caption (заголовок) и свойство Name (имя формы, используемое в программном коде для определения формы).

Помимо обычных свойств могут быть вложенные свойства, которые помечаются значком «+». При двойном щелчке на имени вложенного свойства раскрывается список уточняющих свойств, а значок «+» меняется на «-».

В Delphi поля редактирования, командные кнопки, поля статического текста и другие элементы управления, которые использу ются для создания приложений называются компонентами.

Формы и компоненты рассматриваются как объекты.

Чтобы компонент добавить к форме, надо сначала щелкнуть на палитре компонентов на кнопке с нужной пиктограммой, затем в той точке формы, где должен находиться правый верхний угол компонента. В результате этого на форме появится компонент, например “Поле редактирования”, стандартного размера. Чтобы получить компонент нужного размера необходимо после щелчка на палитре компонентов поместить курсор мыши в ту точку формы, где должен находиться левый верхний угол, нажать кнопкцу мыши и, удерживая ее нажатой, переместить курсор в точку, где должен находиться правый нижний угол компонента. Каждый компонент формы окружен 8 маленькими квадратиками (маркеры выделения). При этом компонент называют выделенным или маркированным, а его свойства отбражаются в окне Object Inspector. Также как и для формы. Delphi позволяет легко изменить размер и положение компонента на форме.

Чтобы изменить положение компонента, надо установить курсор мыши на изображение компонента, нажать левую кнопку мыши и, удерживая ее нажатой, переместить границу компонента в нужную точку формы, отпустить кнопку мыши.

Чтобы изменить размер компонента, надо его выделить (щелкнуть на изображении компонента), спозиционировать кнопку мыши на одном из маркеров компонента, нажать левую кнопку мы-

ши и, удерживая ее нажатой, изменить положение границ компонента, затем отпустить кнопку мыши.

Свойства компонента можно изменить в окне Object Inspector. Чтобы свойства нужного компонента появились в окне Object Inspector, надо выделить нужный компонент или выбрать его имя из раскрывающегося списка объектов, кнопка раскрытия которого находится в верхней части окна Object Inspector.

Рассмотрим создание приложение для решения задачи нахождения суммы двух чисел. Начнем с организации проекта как это рассматривалось выше. На форме расположим следующие компоненты два «Поля редактирования» (Edit1, Edit2), две «Командные кнопки» (Button1, Button2), три «Метки» (Label1, Label2, Label3). Все они принадлежат библиотеке визуальных компонент STANDART, подробное описание которой рассматривается в следующей лабораторной работе. “Поля редактирования” предназначены для ввода слагаемых. При щелчке по одной из “Командных кнопок “ будет выполняться сложение, а по другой – выход из приложения. Назначение одной из "Меток" – вывод результата, остальные - тексты, поясняющие назначение полей редактирования.

Свойства для различных компонентов приведены в приложении 1. Свойство Caption метки Label1 установить равным "первое слагаемое", для метки Label2 - "второе слагаемое", а для Label3 - "". Для командных кнопок Button1, Button2 установить значение Caption равным соответственно "Вычислить", "Выход". Для полей редактирования Edit1, Edit2, Edit3 очистить значение свойства Text.

Созданная форма показывает, как работает приложение. Очевидно, что пользователь должен ввести значения слагаемых в поля редактирования, затем щелкнуть на кнопке Вычислить. Щелчок кнопки мыши это пример события. Событие это то, что происходит во время работы приложения. В Delphi у каждого события есть имя, например щелчок кнопкой мыши это OnClick, имена для других событий приведены в приложении2.

Реакцией на событие должно быть какое-либо действие. Например реакцией на событие OnClick, произошедшее на кнопке Вычислить, должно быть вычисление суммы двух слагаемых. Реакция на событие реализуется как процедура его обработки, называемая обработчиком события. Таким образом, задача создания приложения состоит в написании необходимых обработчиков событий.

Рассмотрим содание обработчика событий для кнопки Вычислить.

1. Необходимо выделить объект, для которого создается обработчик.

2. Щелкнуть на ярлычке вкладки Events (Событие) окна Object Inspector. В результате этих действий в окне Object Inspector появится карточка со списком событий, которые способен воспринимать маркированный компонент (командная кнопка). В левой колонке вкладки перечислены имена событий, на которые может реагировать маркированный объект. Если для него определен обработчик события, то в правой колонке рядом с именем события выводится его имя.
3. Чтобы создать обработчик события, надо сделать двойной щелчок в поле имени события. В результате открывается окно редактора кода с макетом процедуры-обработчика события, который имеет вид:

```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;
```

4. Между Begin и end можно печатать инструкции языка Pascal. Ниже приведен текст процедуры для события OnClik кнопки Вычислить.

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,s:integer;
begin
a:=StrToInt(Edit1.Text);
b:=StrToInt(Edit2.Text);
s:=a+b;
Label3.Caption:='Сумма ='+IntToStr(s);
end;
```

Исходные данные программа получает из полей редактирования, обращаясь к свойству Text, так как именно это свойство определяет содержимое поля редактирования. Для преобразования изображения числа (характеристика Text содержит текст - данные символьного типа) в программе используется процедура StrToInt, которая помещает в переменные a и b строковое изображение первого и второго слагаемого. Вывод значения суммы программа осуществляет присвоением свойству Caption метки Label3 значения - текста сообщения.

Следующий шаг при создании приложения - сохранение проекта, которая выполняется щелчком мыши на пиктограмме Сохранить. После сохранения можно выполнить компиляцию программы. Если в программе нет синтаксических ошибок, то на экране должно

появиться диалоговое окно, информирующее об успешном завершении компиляции. Следует отметить, что после установки Delphi среда программирования автоматически настроена таким образом, что это окно на экране не появляется. Компилятор создает исполняемый файл программы, который можно будет запустить непосредственно из Windows. Все файлы, созданные при сохранении и компиляции проекта помещаются в тот каталог, где находится файл проекта.

Запускается приложение с помощью команды Run меню Run. После ввода первого и второго слагаемого и нажатия кнопки Вычислить программа вычисляет сумму двух введенных чисел.

Программа Delphi представляет собой набор программных единиц - модулей. Один из модулей, модуль проекта, содержит инструкции, с которых начинается выполнение программы, он формируется средой Delphi.

Ниже приведен текст модуля проекта для нашего приложения.

```
program summa_pr;

uses
  Forms,
  summa in 'summa.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Кроме модуля проекта программа включает в себя модуль формы, который содержит описание стартовой формы приложения и поддерживающих ее работу процедур. Каждой форме соответствует свой модуль.Текст модуля формы для нашего приложения имеет вид:

```
unit summa;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;
```

```
type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Edit2: TEdit;
    Label2: TLabel;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var a,b,s:integer;
begin
a:=StrToInt(Edit1.Text);
b:=StrToInt(Edit2.Text);
s:=a+b;
Label3.Caption:='Сумма ='+IntToStr(s);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```

Значительное количество инструкций в модули записывает Delphi.

Задание

1. Написать программу для реализации одного из заданий.
2. Спроектировать форму так, чтобы она содержала окна для входных и выходных данных, кнопки для запуска приложения, очистки окон редактирования, метки для обеспечения понятности проекта.
3. Решить вопрос организации и управления проектом.
4. Рассмотреть методы отладки приложений.

Вариант 1. Найти корни квадратного уравнения $ax^2+dx+c=0$.

Вариант 2. Найти наименьшее общее кратное двух заданных чисел.

Вариант 3. Заданы три положительных числа. Определяют ли эти числа стороны треугольника и если да, то какого равностороннего, равнобедренного или простого.

Вариант 4. Определить являются ли два целых числа взаимно простыми. (Два числа называются взаимно простыми, если они не имеют общих делителей).

Вариант 5. Найти наибольший общий делитель двух заданных чисел.

Вариант 7. Определить является ли заданное число числом Армстронга. (Число из k цифр, для которых сумма k-х степеней его цифр равна самому числу. Например: $153=1^3+5^3+3^3$).

Вариант 8. Определить является ли число совершенным. Совершенным называется такое натуральное число, которое равно сумме всех своих делителей, за исключением самого числа, например: 28=1+2+4+4+14.

Вариант 9. Определить является ли заданное число автоморфным числом. Автоморфными называются числа, которые содержатся в последних разрядах их квадрата, например: $5^2=25$, $25^2=625$.

Вариант 10. Определить является ли заданная пара чисел парными простыми числами. Парными простыми числами называется два простых числа, разность между которыми равна 2. Например, 3 и 5, 11 и 13, 15 и 17.

Вариант 11. Определить является ли заданное число числом Марсенна. Число Марсенна – это число, которое может быть представлено в виде 2^p-1, где p-простое число.

Вариант 12. Определить является ли заданное число числом Пифагора. Числа Пифагора определяются соотношением $c^2=a^2+b^2$, где a, b, c – целые числа.

Контрольные вопросы

1. Из каких составных частей состоит интегрированная среда разработки программ Delphi?
2. Что представляет собой программа Delphi?
3. Какие действия необходимо выполнить для правильной организации проекта?
4. Как создается обработчик событий для кнопки?
5. Какие приняты соглашения по именам составных частей проекта ?

Лабораторная работа №2

БИБЛИОТЕКА ВИЗУАЛЬНЫХ КОМПОНЕНТ

Библиотека визуальных компонент разбита на одиннадцать, сгруппированных по логическому признаку, страниц. Наиболее часто используются компоненты следующих страниц: Standart, Additional, Dialogs, System. Рассмотрим подробнее каждую из них.

Страница Standart. На ней собраны наиболее часто используемые компоненты:

MainMenu – позволяет конструировать и создавать линейку меню формы и выпадающее меню (визуальный компонент). Конструктор меню активируется после двойного щелчка на значении свойства Items в инспекторе объектов. Здесь надо изменить надпись на название меню, которое требуется добавить. Например можно изменить ее на «&File», где знак & означает, что меню будет появляться при нажатии клавиш «Alt»+ «F». После добавления в меню опции File справа от нее появляется окошко, которое позволяет после щелчка на нем добавить следующий пункт меню. Для добавления пунктов в выпадающее меню необходимо щелкнуть на имеющемся пункте верхнего меню. Далее щелкнуть на нем: окно подсветиться и инспектор объектов приготовится к изменению надписи в нем. После этого в меню добавить команду и изменить свойство Caption в инспекторе объектов. Чтобы добавить разделитель в выпадающее меню, необходимо ввести дефис в свойство надписи элемента меню. По окончанию построения меню закрыть окно конструктора меню.

Для добавления кода любой из опций меню, выбрать нужный пункт меню как при работе программы: при этом разработчик приложения попадает в участок кода, соответствующий процедуре об-

работки события выбора этого пункта меню. Можно воспользоваться имеющейся процедурой события, которая была создана раньше. Для этого в инспекторе объектов перейти на страницу Events и щелкнуть на стрелке вниз в поле события OnClick, чтобы посмотреть имеющиеся варианты и выбрать необходимый.

PopurMenu – позволяет конструировать и создавать всплывающее меню, возникающее при нажатии пользователем правой кнопки мыши (невизуальный компонент). Конструктор этого меню можно вызвать, выбрав из всплывающего меню формы Menu Designer. Этот конструктор работает точно также, как и конструктор главного меню, только все опции разворачиваются вниз в единственном окне. Для добавления какой-либо опции всплывающего меню, необходимо выбрать ее в конструкторе меню.

Label – используется для размещения текста, который не изменяется пользователем, на формах и других конструкциях (визуальный). Свойства компонента представлены в таблице.

Свойства компонента Label

Свойство	Описание
AutoSize: Boolean	Указывает, будет ли метка изменять свои размеры взависимости от от помещенного в ее свойство Caption текста
FocusControl: TwinConTrol	Содержит имя оконного компонента, который связан с меткой клавишами быстрого вызова
Layout: TText Layout	Оперделяет выравнивание текста по вертикали относительно границ метки
SchowAccelChar:Boolean	Если содержит значение True, символ & в тексте метки предшествует символу клавиши быстрого вызова
Transparent: Boolean	Определяет прозрачность метки: если содержит значение False, пространство метки закрашивается собственным цветом
WordWrap: Boolean	Разрешает\ запрещает разрыв строки на границе слова

Edit – используется для ввода пользователем однострочных текстов. Может использоваться для отображения текста(визаальный). Его свойства представлены в таблице.

Свойства компонента Edit

Свойство	Описание
AutoSelect: Boolean	Определяет, будет ли выделяться весь текст в момент получения компонентом фокуса ввода
AutoSize: Boolean	Если содержит значение True, высота компонента автоматически меняется при изменении свойства FontSize
BorderStyle: TborderStyle	Определяет стиль обрамления компонента
CanUndo: Boolean	Содержит значение True, если сделанные пользователем изменения можно убрать методом Undo
CharCase: TEdit CharCase	EcUpperCase – все буквы прописные ecLowerCase – все буквы строчные
HideSelection	Если содержит значение False , выделение текста сохраняется при потере компонентом фокуса ввода
MaxLength: Integer	Определяет максимальную длину текстовой строки
Modified:Boolean	Содержит значение True, если текст был изменен
OnChange: TnotifyEvent	Определяет обработчик события, которое возникает после любого изменения текста
OEMConvert: Boolean	Содержит значение True если необходимо перекодировать текст из кодировки Ms-Dos в кодировку Windows
PasswordChar: Char	Если символ определен, то он заменяет собой любой символ текста при отображении в окне. Используется для ввода поролей.
ReadOnly: Boolean	Если содержит значениеTrue, текст не может изменяться
SelLength: Integer	Содержит длину выделенной части текста
SelStart: Integer	Содержит номер первого символа выделенной части текста
SelText: String	Содержит выделенную часть текста
Text: String	Содержит весь текст

Memo – используется для ввода или отображения многострочных текстов (визуальный).Несколько примеров его использования:

```
{очистить содержимое TMemo}
Memo1.Clear;
{скопировать в Memo текст, введенный в Edit}
Memo1.Lines.Add ( Edit1.Text );
{скопировать в Memo текст из ComboBox }
Memo1.Lines.Add (ComboBox1.Text);
```

Button – используется для создания кнопок, которыми пользователь выполняет команды в приложении (визуальный).

Chexbox – позволяет пользователю выбирать или выключать опции программы (визуальный).

RadioButton – предлагает пользователю набор альтернатив, из которых выбирается одна. Набор реализуется требуемым количеством радиокнопок, размещенных в одной структуре (форме, панели) (визуальный). Пример использования компонента:

```
{изменить цвет компонента Memo1 при активизации радиокнопок}
if  RadioButton1.Checked then  Memo1.Color:=clWhite;
if  RadioButton2.Checked then  Memo1.Color:=clAqua;
```

ListBox – представляет собой стандартное окно списка Windows, позволяющее пользователю выбирать пункты из списка (визуальный).

ComboBox – объединяет функции Listbox и Edit. Пользователь может либо ввести текст, либо выбрать его из списка (визуальный).

Scrollbar – представляет собой стандартную линейку прокрутки.и служит для управления положением видимой части форм или компонент управления (визуальный).

Radiogroup – комбинация GroupBox с набором RadioButton, служит специально для создания группы радиокнопок. Можно размещать в компоненте несколько радиокнопок, но никакие другие органы управления не разрешены (визуальный).

Panel – контейнер для группирования органов управления. Может использоваться также для построения линеек состояния, панелей и палитр инструментов (визуальный).

Страница ADDITIONAL. На странице с этой закладкой собрана еще одна группа компонентов, которые также часто используются:

BitBtn – используется для создания кнопок, на которых располагается битовая матрица (например кнопка ОК с галочкой) (визуальный). Ее отличительная особенность – свойство Glyph, спомощью которого определяется растровое изображение на поверхности кнопки. В комплект Delphi входит множество рисунков, разработанных специально для размещения на этих кнопках. По умолчанию рисунки для кнопок размещаются в папке Program Files\Common Files\Borland Schared\Images\Buttons. Свойство Kind может принимать следующие значения: bkCustom, bkAbort, bkAll,bkCansel, bkClose, bkHelp, bkIgnore, bkNo, bkRetry, bkYes, при этом кнопка имеет вид одной из стандартных разновидностей кнопок таких как Abort, All, Cansel, Close, Help и т.д. Щелчок на любой из таких кнопок, кроме кнопок bkCustom, bkHelp, закрывает модальное окно и возвращает в программу результат вида mrXXX (bkAbort –mrAbort, bkOk – mrOk, bkCansel – mrCansel) Кнопка bkClose для модального окна возвращает значение mrClose, а для главного окна программы – закрывает его и завершает работу программы. Кнопка bkHelp автоматически вызывает раздел справочной службы, связанный со свойством HelpContex формы, на которую кнопку помещена.

SpeedButton – спецализированная кнопка для использования с компонентом Panel. Применяется для создания линеек инструментов и специальных наборов, включающих в себя постоянно нажатые кнопки. Индикатором состояния кнопки служит логическое свойство Down, которое имеет значение True, если кнопка утоплена. Свойство доступно для записи, что позволяет изменять состояние кнопки программно.

MaskEdit – (визуальный) используется для форматирования данных или для ввода символов в соответствии с шаблоном, который задается свойством EditMask: String. Если это свойство не задано, компонент работает как обычное текстовое поле Edit.Свойство IsMasked: Boolean доступно только для чтения и содержит значение True если строка шаблона задана. Свойство EditText: String содержит текст до наложения на него маски ввода (то есть то, что ввел пользователь), а свойство Text: String может содержать либо исходный текст, либо результат наложения на него маски.

Шаблон состоит из трех частей, отделенных друг от друга точкой с запятой. Первая часть задает маску ввода, вторая – это символ 0 или 1, определяющий, нужно ли записывать в свойство Text результат наложения маски (1) или требуется оставить исходный текст (0). В третьей части указывается символ, который в поле с маской ввода, будет присутствовать в местах, предназначенных для ввода символов. Например, для ввода семизначного номера телефона текст перед началом ввода может выглядеть так

(095) XX-XX-XXX

Маска состоит из описателей мест ввода, специальных символов и литералов. Описатель указывает, какой именно символ (например, L– буква ,A– буква или цифра, C– любой символ,0– цифра) может ввести пользователь в данное место, описатель указывает место для одного символа. Литерал вставляется в текст, показываемый в окне редактора, пользователь не может его менять. Специальные символыформируют дополнительные указания редактору, например, \ – следующий символ литерал, ! – подавляет начальные пробелы, > – все слеедующие за этим символом буквы преобразуются в прописные буквы,< – все слеедующие за этим символом буквы преобразуются в строчные буквы .

StringGrid – для отображения информации в строках и столбцах таблицы (визуальный). Таблица делится на две части – фиксированную и рабочую. Фиксированная часть служит для показа заголовков,колонок и рядов, а также для ручного управления их размерами. Обычно фиксированная часть занимает левую колонку и верхний ряд таблицы, с помощью свойств FixedCols и FixedRows можно задать другое количество фисированных колонок и рядов. Рабочая область – это остальная часть таблицы, котора может иметь произвольное количество колонок и рядов, более того, эти величины могут изменяться программно.

Центральным свойством компонента является свойство Cells: String – двухмерный массив ячеек, каждая из которых может содержать произвольный текст. Конкретная ячейка определяется парой чисел – номером колонки и номером ряда, на пересечении которых она находится. Свойства ячеек позволяют программно прочитать или записать содержимое ячеек. Например:

Cells(i,j)=’ячейка рабочей зоны’;

Количество ячеек по каждому измерению хранит пара свойств ColCount (количество колонок) и RowCount (количество рядов). Значения этих свойств и , следовательно размеры таблицы могут меняться как на этапе разработки программы так и входе ее выполнения. Все свойства этого компонента представлены в таблице.

Свойства компонента StringGrig

Свойство	Описание
BorderStyle : TBorderStyle	Определяет рамку компонента: bsSingle – рамка толщиной 1 пиксел, bsNone – рамки нет
Cells(ACol, Arow:Integer)	Определяет содержимое ячейки с табличными координатами Acol, Arow

Col: LongInt	Содержит номер колонки с ячейкой, имеющей фокус ввода
ColCount: LongInt	Количество колонок таблицы
Cols[Index: Integer]	Содержит вс строки колонки с индексом Index
ColWidths[Index: LongInt]	Ширина колонки с индексом Index
DefaultColWidth: Integer	Значение ширины колонки, заданное по умолчанию
DefaultDrawing: Boolean	Разрешает\ запрещает автоматическую прорисовку служебных элементов таблицы.
DefaultRowHeight: Integer	Значение высоты рядов, заданное по умолчанию
EditorMode: Boolean	Разрешает\ запрещает редактирование ячеек. Игнорируется, если свойство Options включает значение goAlwayseShowEditor или не включает значение goEditing
FixedColor: Tcolor	Определяет цвет фиксированной зоны
FixsedCols:Integer	Количество колонок фиксированной зоны
FixsedRows:Integer	Количество рядов фиксированной зоны
GridHeight: Integer	Значение высоты таблицы
GridLineWidth: Integer	Толщина линий таблицы
GridWidth: Integer	Значение ширины таблицы
LeftCol: LongInt	Номер самого левого столбца, видимого в зоне прокрутки
Objects[ACol, Arow:Integer]: Tobject	Обеспечивает доступ к объекту, связанного с ячейкой
Options: TgridOptions	Параметры таблицы
Row: LongInt	Номер ряда ячеек, имеющих фокус ввода
RowCount: LongInt	Количество рядов таблицы
Свойство	Описание
RowHeights[Index:LongInt]: Integer	Значение высоты ряда с индексом Index
Rows[Indexs:Integer]: Tstrings	Содержит все текстовые строки ряда с индексом Index
TabStops[Index:LongInt]: Boolean	Разрешает\ заперщает выбор колонки с индексом Index при обходе ячеек с помощью клавиш Tab. Игнорируется,

	если свойство Opyions не содержит значение goTabs
TopRow: LongInt	Номер самого верхнего ряда, видимого в прокручиваемом зоне ячеек
VisibleColCount: Integer	Количество колонок, видимых в зоне прокрутки
VisibleRowCount: Integer	Количество рядов, видимых в зоне прокрутки.

DrawGrid – используется для отображения в строках и столбцах нетекстовых данных.

Image – используется для отображения графики (визуальный).

Bevel – используется для рисования прямоугольника, изображаемого выступающим или утопленным.

ScrollBox – используется для создания зон отображения в виде прокруток (визуальный).

Страница System.

Timer – компонент служит для отсчета интервалов реального времени. Его свойство Interval определяет интервал времени в мили секундах, который должен пройти от включения таймера до наступления события OnTimer. Таймер включается при установке значения True в его свойство Enabled. Однажды включенный таймервсе время будет возбуждать событие OnTimer до тех пор, пока его свойство Enabled не примет значение False.

PaintBox – назначение этого компонента – предоставить пользователю простое окно с канвой для рисования произвольных изображений. Канва содержится в свойстве Canvas компонента, графические инструменты в свойствах Font, Pen и Brush, а собственно рисование осуществляется в обработчике события OnPaint. Представленный ниже обработчик создает окно, в котором нарисован красный овал с надпись внутри Delphi.

```
procedure TForm1.PaintBox1Click(Sender: TObject);
var x,y : Integer;
begin
  with PaintBox1.Canvas do
      begin
        Brush.Color:=clRed;
        Ellipse(0,0, Width, Height);
        Font.Name:='Arial';
        Font.Size:=Height div 5;
        Font.Style:=[fsBold, fsItalic];
        X:=(Width-TextWidth('Delphi')) div 2;
```

```
            Y:= (Height-TextHeight('D')) div 2;
            TextOut(X,Y,'Delphi');
          end;
    end;
```

MediaPlayer – медиаплейер, который представляет набор кнопок для управления различными мультимедийными устройствами (компакт-дисками, звуковыми картами). Если компьютер оснащен звуковой картой, то компонент вставляется в форму, в его свойство FileName вводится название любого файла с расширением .wav, установить в свойство AutoOpen компонентазначение True и запустить программу – после щелчка мышью по кнопке запуска. После запуска должен зазвучать выбранный музыкальный фрагмент.

Страница Dialogs. В состав Windows входит ряд типовых диалоговых окон, таких как окно выбора загружаемого файла, выбора шрифта, настройки принтера. В Delphi реализованы классы, объекты которых дают прогрммисту удобные способы создания и использования таких окон.

Работа со стандартными окнами осуществляется в три этапа.

Вначале на форму помещается соответствующий компонент (невизуальный) и осуществляется настройка его свойств. На втором этапе осуществляется вызов стандартного для классов диалоговых окон метод Execute, который создает и показывает на экране диалоговое окно. Вызов этого метода обычно располагается внутри обработчика какого-либо события. Например, обработчик выбора в меню команды «Открыть файл»может вызывать метод Execute компонента TopenDialog, обработчик щелчка на кнопке «Сохранить» может вызвать такой же метод компонента TsaveDialog. Только после обращения к методу Execute на экране появится соответствующее диалоговое окно. Это окно является модальным, поэтому сразу после обращения к методу Execute дальнейшее выполнение программы приостанавливается до тех пор пока пользователь не закроет окно. Поскольку Execute логическая функция, она возвращает в программу значение True, если результат диалога с пользователем был успешным.

Проанализировав результат вызова метода Execute, программа может приступить к выполнению третьего этапа – использовать введенные с помощью диалогового окна данные – имя файла, параметры принтера, выбранный шрифт.

Далее представлена программа для просмотра содержимого текстового файла. Для этого на пустую форму необходимо поместить компонент OpenDialog, кнопку Button и многострочное окно Memo. При работе программы щелчок на кнопке будет сигналом о

необходимости загрузить в поле новый файл. Обработчик события Onclick будет иметь вид:

```
procedure TForm2.Button1Click(Sender: TObject);
var S: String;
F:TextFile;
begin
 if OpenDialog1.Execute and FileExists(openDialog1/Filename)
  then begin
       AssignFile(F, OpenDialog1.FileName);
       Reset(f);
       Memo1.Lines.Clear;
       while not eof(f) do
          begin
            readln(F, S);
            Memo1.Lines.Add(S);
          end;
       CloseFile(F);
      end;

end;
```

Страница Dialogs содержит следующие компоненты OpenDialog SaveDialog – окна открытия и сохранения файлов, OpenPictureDialog, SavePictureDialog – окна открытия и сохранения изображений, FontDialog – выбор шрифта, ColorDialog – выбор цвета, PrintDialog – настройка параметров печати, PrinterSetupDialog – настройка параметров принтера, ReplaceDialog – окно поиска и замены.

Задание

1. Написать программу, которая предусматривает следующие действия:
   - приложение должно иметь три формы;
   - первая форма должна включать в себя компоненты страницы Standart, вторая – компоненты страницы Additidnal, третья – компоненты страниц System и Dialogs.
   - каждая страница должна содержать кнопки перехода с одной формы на другую в обе стороны.
2. Меню и кнопки страницы Standart должны позволять очищать и менять свойства форм и компонентов (цвет, размеры, вид, размер, цвет шрифта).

3. Вторая форма должна быть спроектирована для решения задачи вариант, которой приведен ниже. Элементы вводимой и результатной матрицы должны быть отображены в компоненте StringGrid.
4. Третья форма должна обязательно содержать окно диалога «открыть файл» для выбора файла, содержимое которого должно выводиться в Memo, а также одно из окон диалога «Шрифты», «Цвет» или «Найти».

Вариант 1. В матрице A(6X5) удалить все строки, которые начинаются с 0.

Вариант 2. В матрице A(4X6) найти максимальные элементы и заменить их среднеарифметическим всех элементов матрицы

Вариант 3. В матрице A(6X6) найти минимальные элементы и заменить последний из них суммой элементов строки, в которой он находится.

Вариант 4. В матрице A(3X6) найти нулевые элементы и заменить их максимальным элементом матрицы.

Вариант 5. В матрице A(5X5) найти максимальный и минимальный элементы среди элементов кратных 5, найти их сумму и заменить ею нулевые элементы матрицы.

Вариант 6. В матрице A(4X5) найти среднеарифметическое максимального и минимального элементов матрицы и заменить найденным числом все первые элементы строк.

Вариант 7. В матрице A(5X6) найти количество нулевых элементов в каждой строке. Матрица B(5X7) должна содержать все элементы матрицы A, а первый столбец информировать о количестве нулевых элементов в матрице A.

Вариант 8. В матрице A(6X4) все максимальные и минимальные элементы и заменить 0.

Вариант 9. В матрице A(3X7) среди минимальных элементов столбцов найти максимальный и заменить его нулем.

Вариант 10. В матрице A(4X6) найти максимальные элементы строк и заменить их нулями

Вариант 11. В матрице A(4X4) найти максимальные и минимальные элементы строк, дополнить матрицу одним столбцом, куда поместить сумму найденных элементов.

Вариант 12. В матрице A(4X6) найти среднеарифметическое положительных элементов каждого столбца. Дополнить матрицу строкой, элементы которой будут равны найденным среднеарифметическим.

Контрольные вопросы

1. По какому принципу сформированы страницы визуальных компонент?
2. Какие визуальные компоненты можно использовать в качестве кнопок выполнения заданий?
3. Какие визуальные компоненты собраны на странице Standart?
4. Какой из визуальных компонент удобно использовать для работы с двумерными массивами?
5. Компоненты какой страницы позволяют открывать типовые диалоговые окна(выбор загружаемого файла, настройки принтера и т.д.)?

Лабораторная работа №3

ОБРАБОТКА ИСКЛЮЧЕНИЙ

Исключение – механизм, созданный для упрощения процесса сообщения об ошибке пользователю или для обработки ошибок, возникающих при выполнении программы.

Исключения позволяют выделить определенные области программного кода для обработки ошибок. В частности, можно защитить целый раздел кода таким образом, что если внутри него возникнет ошибка, то проблема будет обработана в отдельном участке кода с помощью операторов, предназначенных для этих целей. Эта технология охватывает так же вложенные вызовы функций, так что можно начать выполнение защищенного блока, выполнить несколько вложенных вызовов, а при возникновении ошибки возвратиться назад в область кода, предназначенную для обработки ошибки.

Базовый синтаксис исключений

```
try
<оператор 1>
<оператор 2>
……
<оператор n>
except
    on  {тип ислючений} do <оператор>
end;
```

В следующем простом примере показано, как создать блок try……except, позволяющий обрабатывать возникающую ошибку и сообщить о ней пользователю.

```
procedure TForm1.UserExceptionClick(Sender: Tobject);
var
    i, j, k: Integer;
begin
    j:=0;
    try
        k:=i div j;
         ShowMessage(IntToStr(k));
    except
         on EdivByZero do
              MessageDlg (‘произошло деление на ноль’, mtError,
               [mbOk], 0);
     end;
end;
```

В данном примере происходит ошибка деления на ноль. После возникновения ошибки выполняется код, расположенный после слова except. В блоке try……except выполняется явная обработка исключения EdivByZero и пользователь получает сообщение об ошибке.

Оператор ShowMessage(IntToStr(k)); никогда не будет выполен. Размер блока кода, который должен быть обойден, может быть любого размера. Глубина вызовов функций в коде тоже не имеет значения. Если любая из функций в блоке try……except возбудит ошибку, обрабатываемую в данном блоке, управление автоматически перейдет к оператору выдачи сообщения.

Delphi поставляется с большим набором встроенных классов исключений, предназначенных для обработки широкого диапазона состояний ошибок. Пример классов исключений для ошибок деления на ноль, ошибок файлового ввода/вывода, неправильного приведения типов и других ситуаций:

```
EdivByZero = class(Exception);
EZeroDivide = class(Exception);
EConvertError = class(Exception);
EInvalidPointer = class(Exception);
EIntOverflow = class(Exception);
EOverflow = class(Exception);
EStackOverflow = class(Exception);
EInvalidCast = class(Exception);
EPrivelege = class(Exception);
EPropReadOnly = class(Exception);
EPropWriteOnly = class(Exception);
```

Этот список далеко не полон. Чтобы ознакомиться с классами исключений более подробно, можно использовать оперативную справку или просмотреть исходные файлы в каталоге Include\VCL.

Блок try……except дает возможность обрабатывать нескольких классов исключений. Это наглядно показано в следующем примере.

```
Procedure Tform1.Button1Click(Sender:Tobject);
  var i : integer;
  begin
   try
       i:=StrToInt(Edit1.Text);
       x:=1/i;
   ……………
    except
        on  C:EdatabaseError do
           ShowMessage( C.Message);
        on  E:EDivByZero do
           ShowMessage( E.Message);
        on  A:Exception do
           ShowMessage( C.Message);
        on  C:EConvertError do
           ShowMessage( B.Message);
     end;
end;
```

Одна из возможных ошибок, возникающих при работе программы, ошибка типа EDivByZero. При этом будет вызвано исключение, которое не принадлежит типу EDatabaseError . Выполнится второй оператор except, а остальные будут пропущены.

Этот же пример демонстрирует перехват исключений в общем операторе except. При вводе символьной информации в окно Edit1 функция StrToInt возбудит исключение EConvertError, но управление будет передано к блоку on A:Exception, так как Exception – общий вид исключений.

Принимая во внимание эту схему, логичнее расположить блоки обработки исключений в методе в следующем порядке:

EDatabaseError
EDivByZero
EConvertError
Exception

Сначала будет предпринята попытка найти совпадение в первых трех операторах except и только, если она закончится неудачей, управление перейдет к оставшемуся оператору Exception.

Еще один вопрос, возникающий при рассмотрении обработки исключений – создание и возбуждение собственных исключений. Чтобы создать собственный класс исключений, необходимо записать следующий оператор

type <имя_исключения>=class (Exception);

Затем необходим способ сигнализировать, что возникла исключительная ситуация. Этот процесс называется возбуждением исключения. Возбуждение исключения происходит при использовании ключевого слова raise и затем создается новый объект исключения.

Raise <имя_исключения>.Create (‘сообщение’);

Рассмотрим пример создания и возбуждения собственных исключений

```
Unit person;
interface
uses SysUtils, WinTypes, Winprocess,
      Messages, Classes, Graphics,
       Controls, Forms, Dialogs,
       StdCtrls, Buttons;
type
  Error1=class(Exception);
  Error2=class(Exception);
  TPersonForm=class(TForm)
            Name: Tedit1;
            Adress : Tedit2;
            Den: Tedit3;
            Mes: Tedit4;
            God: Tedit5;
            Procedure ProverkaDatClick(Sender:Tobject);
  end;
 var PersonForm: TpersonForm;

implementation
   {$R *.DFM}
    procedure  PersonForm.ProverkaDatClick(sender:Tobject);
    var  S:string;
    begin
      if StrToInt (Den.Text) > 31 then
          raise Error1.Create (‘неверно введен день рождения’);
      try
```

```
        StrToInt (God.Text);
      except
        on EConvertError do
            raise Error2.Create('вместо года введена символьная
строка');
       end;
      S:=Mes.Text;
      If  S<>'январь' or  S<>'февраль'…… or s<>'декабрь' then
            Raise error3.create ('неверно введено название месяца');
   end;
end.
```

Следующий фрагмент программного кода показывает, как можно повторно возбудить исключение:

```
procedure TForm1.Button1click (Sender: Tobject);
var
  k : integer;
begin
   try
     k:=StrToInt('sam');
      SchowMessage(IntToStr(k));
   except
       on  E:Exception do  begin
            SchowMessage('Что-то не так ');
            raise;
          end;
    end;
end;
```

Для повторного возбуждения используется одно ключевое слово raise. В примере умышленно создается ошибка EConvertError, отображается пользователю сообщение и повторно возбуждается исключение. При запуске программы сначала будет показано пользовательское сообщение об ошибке, а затем стандартное сообщение об ошибке EConvertError. Повторное возбуждение исключений позволяет объединить преимущества пользовательской и встроенной обработки.

Чтобы при отладке приложения, в котором выполняется обработка исключений, можно было видеть сообщение об ошибке, необходимо выбрать меню Tools| Debugger Oprions| Language Exceptins и снять флажок Step on Delphi Exception. Затем повторно запустить приложение. При запуске приложений с обработкой исключений

желательно так же отключить оптимизацию. Для э того выбрать меню Project| Options| Compiler.

Задание

1. Проанализировать задачу одного из вариантов и выявить те типы ошибок, которые могут возникнуть при реализации на ЭВМ.
2. Создать приложение для решения задачи, выполнив обработку исключений. Для обработки ошибок деления на ноль, неправильного приведения типов использовать встроенные исключения.
3. Для обработки остальных ошибок создать и возбудить собственные классы исключений.
4. Предусмотреть перехват исключений в общем операторе except.

Вариант 1. Выполнить табулирование функции

$$F(x, y, z)= \begin{cases} \ln(x/2.5)+y^2/(z-1) & \text{если } 0<=x<7,\ -5<=y<0,\ z=1 \\ \sqrt{x/y}+z & \text{если } x<0,\ y<-5,\ z<1 \\ (x+5.8*y)/(z-3) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 2. Выполнить табулирование функции

$$F(x, y, z)= \begin{cases} \exp(x/z)+y^2*(x-1.5) & \text{если } 1<=x<5,\ -5<=y<1,\ z<1 \\ x/y+\sqrt{z} & \text{если } x<0,\ y<-5,\ 2>z>1 \\ (1/x-5.8*y)/\sqrt{(z-3)} & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 3. Выполнить табулирование функции

$$F(x, y, z)= \begin{cases} \sin(x*y)/(z-1) & \text{если } -1<=x<3,\ -5<=y<5,\ z>=3 \\ (x+y)/z & \text{если } x<-2,\ y<-5,\ z<1 \\ \ln(z-3/(x-1)) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 4. Выполнить табулирование функции

$$F(x,y,z)=\begin{cases} \ln(1/x)+\sin(1/(z-y)) & \text{если } 0\le x<7,\ -5\le y<0,\ z=1 \\ \sqrt{z}+1/(x*y) & \text{если } x<0,\ y<-5,\ z<1 \\ x/(z-y) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 5. Выполнить табулирование функции

$$F(x,y,z)=\begin{cases} 1/x+x/\sin(z-y) & \text{если } 0\le x,\ -5\le y<0,\ z=1 \\ \sqrt{z}+4.5*x+1/(x*y) & \text{если } x<-3.5,\ y<-5,\ z<1 \\ x/\ln(z-y) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 6. Выполнить табулирование функции

$$F(x,y,z)=\begin{cases} z/x+(2-y)/\sin(z/y) & \text{если } 0\le x,\ -3\le y<0,\ z>1 \\ \sqrt{z}+4.5*x+1/(x*y) & \text{если } x<-1,\ y<-5,\ z<5 \\ x/\ln(z-y) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 7. Выполнить табулирование функции

$$F(x,y,z)=\begin{cases} 1/(x-y)/x/\sin(z-y) & \text{если } -2\le x,\ -1\le y<0,\ z<1 \\ \sqrt{z}-5*x+1/(x*y) & \text{если } x<0,\ -5<y<-1,\ z>5 \\ x/(z-y) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 8. Выполнить табулирование функции

$$F(x,y,z)=\begin{cases} \sqrt{1/x+x/\sin(z-y)} & \text{если } 0\le x,\ -3\le y<-1,\ z>1.5 \\ z/(4.5-x)+1/(z*x-y) & \text{если } x<-1,\ y<-5,\ z<1 \\ (x-1)/\sin(z/y) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 9. Выполнить табулирование функции

$$F(x, y, z)= \begin{cases} 1/(x-1/y)+x/(z-y) & \text{если } -1<=x,\ -5<=y<1,\ z>3 \\ \sqrt{\sin(z+4.5*x*y)} & \text{если } x<-7,\ y<-5,\ z<1 \\ \mathrm{Ln}(x/(z-y)) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 10. Выполнить табулирование функции

$$F(x, y, z)= \begin{cases} \mathrm{Exp}(1/x)+x/(1-y) & \text{если } 0<=x,\ 0<=y,\ z>3 \\ \sqrt{z+4.5/x}+1/\mathrm{Ln}(x*y) & \text{если } -3<x<-1,\ y<-5,\ z<1 \\ \sqrt{x/(z-y)} & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 11. Выполнить табулирование функции

$$F(x, y, z)= \begin{cases} 1/x+x/(1-y) & \text{если } 0<=x,\ 0<=y<2,\ z>3 \\ \sqrt{y-4.5/x}+1/\sin(x-y) & \text{если } x<-1,\ y<-5,\ z<0 \\ \mathrm{Ln}(\sqrt{x/(z-y)}-5.84) & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Вариант 12. Выполнить табулирование функции

$$F(x, y, z)= \begin{cases} \sin(1/x)-x/(z-y) & \text{если } 0<=x,\ 0<=y<2,\ z>3 \\ 4.5/x+1/\mathrm{Ln}(x-y) & \text{если } x<-5,\ y<-5,\ z<1 \\ \sqrt{(x+5/y)/(z-y)} & \text{иначе} \end{cases}$$

Задать область значений x, y, z и шаги табулирования по x, y, z.

Контрольные вопросы

1. Что такое исключение? Как работает механизм исключений?
2. Представить базовый синтаксис исключений .
3. Как создать собственный класс исключений ?
4. Для чего используется повторное возбуждение исключений?
5. Как выполнить обработку нескольких классов исключений?

Лабораторная работа №4

ОРГАНИЗАЦИЯ ПОТОКОВ

Процесс – это термин Win32, обозначающий загруженную в данный момент программу. Выполняемый файл на дискеявляется просто файлом. Когда он запускается, он становится процессом.

Процессы ничего не делают. Они просто существуют. Та сущность, которая что-то делает в программе называется потоком (thread). Каждая программа содержит как минимум один поток.

Когда Windows загружает процесс, она просто открывает отображаемый в памяти файл и загружает содержимое выполняемого файла или Dll в память. Процессы сами по себе инертны. Они ничего не делают. Всю работу выполняют потоки.

С каждым процессом связано один или несколько потоков. Потоки – это часть исполняющейся программы. Windows 98 и Windows NT являются истинными многозадачными операционными системами. В любой момент времени они выполняют несколько потоков. Каждому из этих потоков Windows назначает интервал времени, который называется квантом.

С каждым потоком связан свой контекст. Запись TContext – это структура данных, содержащая информацию о состоянии потока. Каждому из потоков в определенный момент времени выделяется период для реального выполнения потока. Когда подходит очередь какого-либо потока , загружается его контекст. То есть значения регистров процессора из структуры TContext физического процессора компьютера. Затем поток начинает выполняться с текущей позиции указателя инструкций. Посл е выполнения нескольких циклов подходит очередь следующего потока. Перед прерыванием своего выполнения запись TContext обновляется текущим состоянием процессора. Далее в память загружаются значения TContext для следующего потока. На протяжении следующего интервала времени выполняется уже другой поток.

Никаких два потока никогда не могут выполняться в одно и то же время. Однако пользователю кажется, что все потоки выполняются одновременно. Это связано с высокой скоростью работы компьютера.

При работе с потоками необходимо выполнить две основные задачи:

- создать поток
- создать функцию, служащую точкой входа в поток.

Поток создается с помощью следующей функции:

```
function CreateThread (
LpThreadAttributes: Pointer;          //адрес атрибутов безопасности
DwStackSize:  Dword;                  //размер  стек для потока
LpStart Address: TFNThreadStart;      // адрес функции потока
LpParameter: Pointer ;                //аргументы  нового потока
dwGreationFlags : Dword;              //флаги создания
var lpThreadId):                      // идентификатор потока
Thandle                               //результат функции-дескриптор
```

Ниже рассмотрен пример создания простого потока.

```
unit Unit1;
interface
  uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
Function ThreadFunc(P: pointer): LongInt; stdcall;
var i: Integer;
begin
 for i:=1 to 10000 do
     Form1.Edit1.text:=IntToStr(i);
end;
Function ThreadFunc1(P: pointer): LongInt; stdcall;
```

```
var i,y: Integer;
begin
 for i:=1 to 100 do begin
   Form1.Edit2.text := IntToStr(i);
   y:=i*i*i;
   Form1.Edit3.text := IntToStr(y);
   end;
end;
procedure TForm1.Button1Click(Sender: TObject);
var hTread,hTread1: THandle;
ThreadID, ThreadID1: DWord;
begin
   hTread:=CreateThread(nil,
                        0,
                        @ThreadFunc,
                        nil,
                        0,
                        ThreadID);
   if hTread=0 then
      MessageBox(Handle, 'no Thread', nil, mb_ok);
   hTread1:=CreateThread(nil,
                        0,
                        @ThreadFunc1,
                        nil,
                        0,
                        ThreadID1);
   if hTread1=0 then
      MessageBox(Handle, 'no Thread1', nil, mb_ok);
end;
end.
```

При запуске эта программа открывает простое окно, содержащеее три поля редактирования и одну командную кнопку. Необходимо Button1. При этом в поле Edit1 будут выводиться числа от 1 до 10000, а в поле Edit2 появятся числа от 1до 100, для каждого из которых вычисляется третья степень числа, результат выводится в поле Edit3. Пока выполняются описанные действия пользователь может манипулировать окном: передвигать его, изменять размеры.

Структура программы очень проста. Она содержит метод, который запускает главный поток. Метод запускает две функции ThreadFunc и ThreadFunc1 как два отдельных потока.

Поток создается в следующем программном коде :

```
hTread:=CreateThread(nil,
                     0,
                     @ThreadFunc,
                     nil,
                     0,
                     ThreadID);
if hTread=0 then
   MessageBox(Handle, 'no Thread', nil, mb_ok);
```

Этот фрагмент кода просто пытается создать поток. Если что-то выполняется не так, то вызывается окно, сообщающее пользователю о возникновении ошибки.

Кроме описанного способа организации нескольких потоков, в Delphi существуют другие возможности. Например, использование объектов класса Tthread. С этим способом можно ознакомится в книге В. Фараонова Delphi 6.

Задание

1. Написать программу в которой выполняется 2 потока. Один из которых предназначен для открытия большого текстового файла и вывода его, например в компонент Memo, а второй поток – для решения одного из вариантов первой лабораторной работы.

Контрольные вопросы

1. Что такое процесс и поток, описать принципы их работы.
2. Какие способы существуют для определения нескольких потоков?
3. Какие формальные параметры используется в функции CreateThread?
4. Что обычно используется в качестве фактических параметров в этой функции?
5. Какие операционные системы являются многозадачными?

Лабораторная работа №5

ПОСТРОЕНИЕ ДИНАМИЧЕСКИ ПОДКЛЮЧАЕМЫЕ БИБЛИОТЕКИ (DLL)

Динамически поключаемые библиотеки (DLL) предоставляют универсальный механизм интегрирования в программу процедур и

функций, написанных другими программистами и на других, нежели Object Pascal, языках программирования.

DLL реализуются в виде исполняемых модулей, содержащих готовые к работе процедуры, функции и ресурсы. Принципиальное отличие DLL от Object Pascal модулей состоит во-первых в том, что DLL не в состоянии поставлять в программу переменные, константы и типы. Во-вторых в том, что модули связываются с программой на этапе компоновки, то есть статически, а DLL – в момент ее исполнения, то есть динамически. В-третьих изменение любой библиотеки DLL не требует перекомпиляции использующей ее программы.

Для создания DLL вObject Pascal введено рarезtрвированное слово Library, которым должен начинаться текст библиотеки. За словом Library следует правильный идентификатор, который не обязательно совпадает с именем файла : имя DLL определяется именем DLL-файла, а не имдентификатором следующим за словом Library.

Структура текста DLL повторяет структуру обычной программы с тем исключением , что раздел исполняемых опреаторов в DLL играет ту же роль, что и инициализирующая часть модуля: операторы этой части исполняются только один раз в момент загрузки библиотеки в память. Каждое очередное обращение с требованием загрузить библиотеку наращивает на единицу ее счетчик ссылок, но не приводит к выполнению операторов исполняемой части.

В разделе описаний DLL могут объявляться типы , классы, константы и переменные, но они остаются скрытыми от вызывающей программы и могут использоваться только внутри DLL. В разделе описаний помимо стандартных для обычной программы объявлений используется специальный раздел объявления экспортируемых подпрограмм. Этот раздел начинается зарезервированным словом Exports, за которым следует имена экспортируемых программ, например.

```
library MyLibrary;
uses
  SysUtils,
  Classes;

{$R *.res}
Function MyFunc(...);
begin
....
end;
Procedure MyProc;
```

```
begin
...
end;
Exports MuyFync, MyProc;
begin
end.
```

Помимо имени подпрограммы в заголовок DLL помещается ее порядковый номер, точнее, присвоенный ей целочисленный индекс. Это позволяет вызывающей программе ссылаться не на имя, а на индекс подпрограммы, что уменьшает затраты времени на установление с ней связи. Индекс присваивается подпрограмме в соответствии с порядком ее появления в списке Exports: первая подпрограмма получает индекс 0. Программист может изменить такой порядок индексации и явно указать индекс подпрограммы, добавив за ее именем в списке Exports слово index и целое число без знаков в диапазоне от 0 до 32767:

```
Exports MuyFync index 1, MyProc index 2;
```

Можно определить новое внешнее имя экспортируемой подпрограммы, которое будет отличаться от ее настоящего имени. Для этого в списке Exports добавляется слово name и внешнее имя в апострофах:

```
Exports MuyFync name ‘NewF’, MyProc name ‘NewP’;
```

При вызове по имени программа просматривает имена в таблице имен в поисках нужного. Так как имена могут состоять из длинных наборов символов, а самих имен в таблице может быть много, процесс поиска медленнее, чем процесс поиска индекса.

Для создания заготовки библиотечного модуля выбирается команда File| New| Other и в окне хранилища выбирается значок с надписью DLL. В ответ Delphi откроет специальное окно проекта с длинным комментарием, в котором указывается на необходимость вставить ссылку на модуль ShareMem, усли библиотека экспортирует длинные строки в параметрах обращения к подпрограммам или как результат функции. Эта ссылка должна быть первой как в предложении Uses библиотеки, так и в предложении Uses файла проекта программы, которая использует эту библиотеку. Проект сразу необходимо сохранить под именем , предложенным пользователем, чтобы среда Delphi автоматически исправила имя библиотеки в преложении Library. В примере создается DLL для сложения и вычитания комплексных чисел.

```
library Cmpl;

uses
  SysUtils,
  Classes;

{$R *.res}
type TComplex=record
     Re, Im  :  Real;
     end;
Function AddC(x,y:TComplex): TComplex;stdcall;
begin
  Result.Re:=x.Re+ y.Re;
  Result.Im:=x.Im+ y.Im;
end;

Function SubC(x,y:TComplex): TComplex;stdcall;
begin
  Result.Re:=x.Re - y.Re;
  Result.Im:=x.Im - y.Im;
end;

Exports AddC  index 1 name 'AC',
        SubC  index 2 name 'SC';
begin
end.
```

Все функции этой библиотеки используют директиву stdcall, которая обеспечивает совместимость новых функций с функциями API 32-разрядных версий Windows. Если не указывать эту директиву, то обращение к DLL из программ, написанных на других языках, стало бы невозможным.

Для использования подпрограмм из DLL, необходимо описать их как внешние, добавив за словом External имя библиотеки в апострофах:

Procedure MyProc;External ‘MyDll’;

Как уже отмечалосьподпрограмма вызывается по имени или по индексу. В примере из библиотеки MyDll вызывается подпрограмма с внешним именем MyProc. Если нужно сослаться на индекс за именем библиотеки указывается слово index и сам индекс:

Procedure MyProc;External ‘MyDll’ index 2;

В этом случае имя, под которым программа будет известна программе, может не совпадать с ее внешним именем, определенным в DLL. Можно и явно переопределить имя подпрограммы, даже если имеется ссылка на ее внешнее имя:

```
Procedure MyProc;External ‘MyDll’  Name ‘NewName’;
```

В этом варианте предполагается, что экспортируется процедура с внешним именем ‘NewName’.

После любого из указанных выше объявлений экспортируемая подпрограмма становится доступной программе и может вызываться в ней как обычная подпрограмма.

В следующем примере используется библиотека Cmpl.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type TComplex=record
     re, Im  : real
     end;
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    run: TButton;
    Label1: TLabel;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
  procedure runClick(Sender: TObject);
  private
    { Private declarations }
```

```
    public
      { Public declarations }
    end;
function AddC(x,y :TComplex):TComplex;stdcall; External ‘Cmpl.dll';

function SubC(x,y :TComplex):TComplex;stdcall;External  'Cmpl.dll';
    var
     Form1: TForm1;

    implementation

    {$R *.dfm}

    procedure TForm1.Edit1Change(Sender: TObject);
    begin

    end;

    procedure TForm1.Label2Click(Sender: TObject);
    begin

    end;

    procedure TForm1.runClick(Sender: TObject);
    var x, y,z, f1,f:  TComplex;
    begin
    x.Re:=strToFloat(Edit1.Text);
    x.Im:=strToFloat(Edit2.Text);
    y.Re:=strToFloat(Edit4.Text);
    y.Im:=strToFloat(Edit5.Text);
    z.Re:=strToFloat(Edit6.Text);
    z.Im:=strToFloat(Edit7.Text);
    f1:=AddC(x,y);
    f:=SubC(f1,z);
    edit3.Text:= edit3.Text+ FloatToSTR(f.re)+FloatToSTR(f.im);
    end;

    end.
```

Задание

1. Построить DLL для выполнения одного из следующих вариантов.

2. Привести пример использования DLL.

Вариант 1. Работа со строками:

- подсчет количества заданных символов;
- удаление всех вхождений подстроки;
- нахождение процента вхождения в строку гласных букв;
- нахождение процента вхождения в строку согласных букв.

Вариант 2. Работа с матрицами:

- нахождение суммы двух матриц;
- нахождение произведения двух матриц;
- сравнение на равенство двух матриц.

Вариант 3. Работа с комплексными числами.

Для двух комплексных чисел (a, b) и (c, d) справедливы следующие арифметические операции:

- (a, b)+(c,d) = (a+c, b+d);
- (a, b)-(c,d) = (a-c, b-d)
- (a, b)*(c,d) = (a*c+b*d, a*d+b*c)
- (a, b)/(c,d) = ((a*c+b*d)/(c^2+d^2), (a*d+b*c)/ (c^2+d^2))
- нахождение модуля комплексного числа |(a,b)|=SQRT(a^2+b^2)

Вариант 4. Расчет треугольника:

- определение вида треугольника по трем сторонам
- нахождение площади треугольника;
- нахождение стороны треугольника по стороне и углу прилежащему к этой стороне;
- нахождение высоты.

Вариант 5. Нахождение площади следующих фигур:

- треугольника;
- прямоугольника;
- квадрата;
- круга;
- параллелограмма;
- трапеции;

Вариант 6.Проведение статистического анализа ряда чисел:

- нахождение средне арифметического;
- нахождение максимального значения;
- нахождение минимального значения;
- нахождение средне квадратичного отклонения.

Контрольные вопросы

1. Что представляют собой динамически присоединяемые библиотеки?

2. Чем отличаются DLL от модулей?

3. Какова структура текста DLL?

4. Для каких целей используется раздел Exports и какие параметры включаются в его состав?

5. Что необходимо указать в программе для использования в ней подпрограмм из DLL?

ЛИТЕРАТУРА

1. В.Фараонов Delphi 6: учебный курс. СПб.:Питер, 2002.
2. Ч. Калверт Delphi 4. Самоучитель:Пер. с англ./ Ч. Калверт – К.:Издательство «ДиаСофт», 1999.
3. Н. Б. Культин. Программирование в Turbo Pascal 7.0 Delphi. СПб.: BHV– Санкт-Петербург, 1998.
4. Э. Возневич. Delphi. Освой самостоятельно. – М.: Восточная книжная компания., 1996.
5. В.А.Острейковский Информатика. -М.:Высш.шк.,1999

СОДЕРЖАНИЕ