Государственное автономное Образовательное

учреждение Астраханской области

высшего образования

«АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ АРХИТЕКТУРНО-СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра систем автоматизированного

проектирования и моделирования

**Лабораторная работа №1**

***Программирование алгоритмов линейной, разветвляющейся и циклической структуры в среде Delphi***

**по дисциплине «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»**

Разработка и внедрение в учебный процесс новой лабораторной работы с использованием нового приобретенного оборудования

и сопроводительных методических рекомендаций

для студентов направления подготовки

09.03.02 «информационные системы и технологии»

направленности (профиля)

«Информационные системы и технологии в строительстве и архитектуре»

очной и заочной форм обучения

Астрахань 2020

Составитель: к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Садчиков П.Н./

(занимаемая должность, (подпись) И. О. Ф.

учёная степень и учёное звание)

Рецензент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_ /

(занимаемая должность, (подпись) И. О. Ф.

учёная степень и учёное звание)

Инструкция по выполнению лабораторной работы по дисциплине «Технологии программирования» рассмотрена и одобрена на заседании кафедры «Системы автоматизированного проектирования и моделирования» ГАОУ АО ВО «АГАСУ»

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_ 2020 г.

Зав. кафедрой  ***\_\_\_\_\_\_\_\_\_\_\_\_\_*** ***/***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***/***

**(**подпись) (инициалы, фамилия)

Согласовано с УМО ГАОУ АО ВО «АГАСУ»

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Специалист УМУ \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (инициалы, фамилия)

Инструкция по выполнению лабораторной работы по дисциплине «Технологии программирования» утверждена и рекомендована к публикации на заседании методического совета направления подготовки 09.03.02 «информационные системы и технологии» ГАОУ АО ВО «АГАСУ»

Протокол № \_\_\_\_ от \_\_\_\_\_\_\_\_\_\_ 2020 г.

Председатель МКН «Информационные системы и технологии»

направленность (профиль)

«Информационные системы и технологии в строительстве и архитектуре»

\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (инициалы, фамилия)

©Садчиков П.Н.

©ГАОУ АО ВО «Астраханский государственный архитектурно-строительный университет»

**ВВЕДЕНИЕ**

**Язык программирования DELPHI**

**DELPHI** – среда разработки программ, ориентированных на работу в Windows.

В основе идеологии DELPHI лежит технология визуального проектирования и методология объектно-ориентированного программирования. Для представления программ используется разработанный фирмой *BORLAND* язык *Object Pascal*, в основе которого *Turbo Pascal*.

В основе DELPHI лежит концепция быстрого создания приложений. Приложение создаётся с использованием компонентов, которые располагаются на форме. При расположении компонентов на форме и задании их свойств одновременно создаются автоматические подпрограммы, характеризующие эти свойства.

Среда DELPHI разработана для профессиональных программистов, но в вузах рассматриваются основные принципы этой среды визуального программирования.

После запуска DELPHI на экране появляется четыре окна интегрированной среды разработки приложений.

*Главные составные части среды программирования:*

1. главное окно;
2. окно Form;
3. окно редактора кода программы;
4. окно инспектора объектов.

***Главное окно*** (располагается в верхней части экрана, имеет заголовок Project1)

В главном окне находится строка главного меню, панели инструментов и палитра компонентов. Главное окно остаётся открытым пока загружено DELPHI. Если вы закрываете окно, значит, заканчиваете работу в DELPHI.

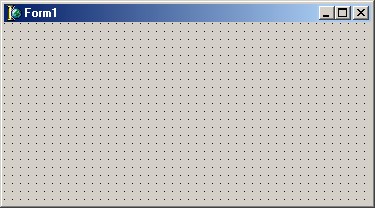
1. строка меню содержит команды для разработки и тестирования приложений и управления ими;
2. панель инструментов содержат кнопки, которые выполняют определённые команды меню;
3. палитра компонентов *Component* (находится правее кнопок панелей инструментов). В ней отражаются компоненты, с помощью которых пользователь создаёт свои приложения.

Компоненты являются основой библиотеки визуальных компонентов. Они позволяют создавать пользовательский интерфейс прикладных программ.

**Компонент** *–* это элемент, имеющий определённые свойства и размещаемый в окне формы. К ним относятся кнопки, надписи, метки и т.д.

Палитра компонентов имеет несколько страниц. Когда мы загружаем DELPHI, активной становится страница *Standart*.





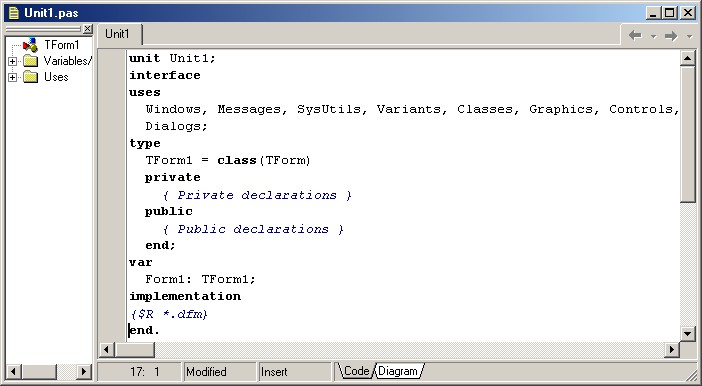
***Окно Form*** - это проект Windows окна будущей программы (если окно приложения на этапе разработки).

Когда мы загружаем DELPHI, на окне формы содержится только кнопка вызова системного меню и три кнопки: минимизация, максимизация и закрытие. Также содержится полоса заголовка и рамка, очерчивающая форму. На форме содержится координатная сетка для упорядочения компонентов на форме.

Работа с формой напоминает игру с конструктором. На форме мы будем располагать различные компоненты.

***Окно редактора кода программы*** – располагается позади окна формы. Оно предназначено для создания и редактирования текста программы. Между содержимым окон формы и программы существует неразрывная связь, которая строго отслеживается системой DELPHI.

Размещение на форме компонентов автоматически изменяет программу. Удаление вставленных фрагментов в программу автоматически ведёт к удалению элементов на форме.



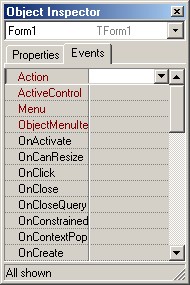
***Окно инспектора объектов*** – это окно создаёт и редактирует свойства компонентов, а также является обработчиком событий компонентов. Инспектор объектов – инструмент, который используется для формирования внешнего вида и функциональных возможностей.

*Окно инспектора объектов содержит две страницы:*

1. свойства Properties
2. события Events

В левой части перечислены все имеющиеся обработчики событий компонентов. События определяют поведение компонента, т.е. будет ли компонент реагировать на щелчок мыши или на нажатие клавиши.

В правых колонках обеих страниц указывается значение свойств или обработчики событий соответствующих компонентов.



1. **Разработка приложения в среде DELPHI**

*Разработка приложений состоит из двух этапов:*

1. этап конструирования формы;
2. кодирование (составление программы).

Например, сконструируем форму, на которой нужно поместить компонент *Label* и в ней поместить надпись: «Я программирую на DELPHI»

* Для того, чтобы поместить компонент на форму, надо выбрать страницу, где находится нужный компонент;
* Выбрать компонент на странице и щёлкнуть по нему мышью;
* Перейти на форму и щёлкнуть по тому месту, где нужно разместить компонент;
* Затем надо установить его свойства.

Создать кнопку на форме так, чтобы после нажатия на эту кнопку появилась надпись: «Нет, только учусь».

1. В компоненте *Button* задаём свойству *Caption* значение *OK*.
2. Перейти на страницу события *Events* и напротив *OnClick* щёлкнуть мышью, открывается процедура
3. В процедуре записываем инструкцию *Label1.Caption:=*’Нет, только учусь’
4. Затем запускаем программу из Меню, команда RUN. Затем щелкаем по кнопке *ОК* Вместо надписи «Я программирую на DELPHI» появится надпись «Нет, только учусь».

**Основные компоненты проекта**

Любая программа на Delphi состоит из файла проекта (файл \*.dpr) и одного или нескольких модулей (файлы с расширением \*.pas). Каждый из таких файлов описывает программную единицу Object Pascal. Файл проекта представляет собой текстовый файл - программу, написанную на языке Object Pascal, предназначенную для обработки компилятором. Текст программы начинается зарезервированным словом **program** и заканчивается словом end. Сочетание end и точки называется терминатором программной единицы, как только в тексте программы встретится такой терминатор, компилятор прекращает анализ программы и игнорирует дальнейший текст программы. Собственно тело программы начинается со слова begin и оканчивается терминатором end. Тело программы или модуля состоит из одного или нескольких операторов языка Object Pascal. В каждом операторе реализуется некоторое действие – изменение значения переменной, анализ результата вычисления, обращение к подпрограмме и т.д. Файл проекта практически полностью формируется самой средой Delphi и редко требует отдельного редактирования.

**Структура модуля**

**Модули** – это программные единицы, предназначенные для размещения фрагментов программ. С помощью содержащегося в них программного кода реализуется вся поведенческая сторона программы. Любой модуль имеет следующую структуру:

Заголовок;

Секция интерфейсных объявлений; Секция реализаций; Терминатор.

Заголовок открывается зарезервированным словом **unit**, за которым следует имя модуля и точка с запятой. Секция интерфейсных объявлений открывается зарезервированным словом **interface,** секция реализаций – словом **implementation.**

В секции интерфейсных объявлений описываются программные элементы (Типы, классы, процедуры и функции), которые будут доступны другим программным модулям, а в секции реализаций раскрывается механизм работы этих элементов. Разделение модуля на две секции обеспечивает удобный механизм обмена алгоритмами между отдельными частями одной программы. В отличие от файла проекта, текст модуля может быть изменен программистом.

**Ввод-вывод данных в DELPHI**

***Ввод:***

Программа может получать результаты из окна ввода, из поля ввода компонентов, а также из файла.

**1. Ввод данных из окна ввода**

Ввод данных производится вызовом функции InputBox. Результатом этой функции является строка, вводимая пользователем.

Вид этой функции:

*Переменная:=InputBox(заголовок, подсказка, значение);* Заголовок – это текст заголовка окна ввода.

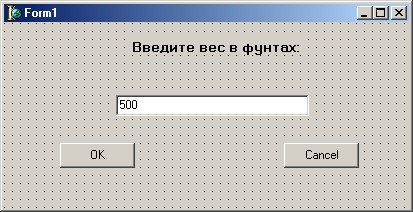
Подсказка – это текст поясняющего сообщения. Он появляется в окне ввода на экране.

Переменная – это имя переменной строкового типа, значение которой должно быть получено из окна ввода.

Например, в программе надо пересчитать вес из фунтов в килограммы.

Инструкция может выглядеть так:

S:=InputBox(‘фунты-килограммы’, 'введите вес в фунтах', ' '); Окно ввода:



В поле ввода вводится число фунтов (например, 500), затем щёлкаем по кнопке OK и переменная S получает значения строкового типа, а чтобы получить число, надо преобразовать введённую строку в число.

**Функции преобразования**

*StrToInt(S)* – функция преобразует строку S в целое число a:=StrToInt(S);

*StrToFloat(S*) – преобразует строку S в вещественное число;

*IntToStr(n*) – преобразует целое n число в строку S;

*FloatToStr(n)* – преобразует вещественное число n в строку S; *FloatToStrF(n,f,l,m*) – преобразует вещественное n в строку;

n – вещественное число; f – формат (способ) изображения; l – точность (общее количество цифр); m – количество цифр после точки.

**2. Ввод данных через компоненты**

Компоненты, в поле ввода которых будем записывать данные, называется компонентами, имеющими фокус ввода. Такими компонентами являются:

**Edit** – однострочный редактор ввода

**Memo** – многострочный редактор ввода

Ввод данных в поле ввода компонентов Edit производится обращением к свойству Text var a: real; a:= StrToFloat (Edit.Text);

Компонент Memo имеет свойства Lines.

***Вывод:***

Вывод результатов может быть выполнен следующим образом:

1. В окне сообщения;
2. в поле ввода компонентов формы;
3. в файл; 4. на принтер.
4. **Вывод в окно сообщения**

Вывод может производиться с помощью процедуры Show Message, функции - MessageDig . Вид:

Show Message (‘сообщение’); Show Message (‘корней нет’);

1. **Вывод результатов в компоненты формы**

Вывод в форму можно сделать в компоненте Label, либо в диалоговом окне других компонентов.

Для вывода информации с помощью Label используется свойство *Caption*. В компоненте Edit используется свойство Text, а в компоненте Memo - свойство Text и Lines.

***Свойства видимых компонентов Left, Top, Width, Height, Visible, Enable, Align, Hint, ShowHint, Cursor, Font***

*Left* – устанавливает отступ слева относительно родительского компонента.

*Top* – устанавливает отступ сверху относительно родительского компонента.

*Width* – устанавливает ширину компонента.

*Height* – устанавливает высоту компонента.

*Visible* – устанавливает видимость компонента.

*Enable* – устанавливает доступность компонента.

*Align* – устанавливает выравнивание компонента относительно границ своего родителя.

*Hint* – текст всплывающей подсказки, которая появляется при наведении на компонент мыши.

*ShowHint* – показывать/не показывать всплывающую подсказку.

*Cursor* – вид курсора при наведении его на компонент.

*Font* – шрифт, которым будет отображать текст на компоненте.

**События видимых компонентов OnClick, OnDblClick,**

**OnMouseDown, OnMouseUp, OnMouseMove, OnKeyPress, OnKeyDown, OnKeyUp, OnEnter, OnExit**

*OnClick* – нажатие и отпускание кнопки мыши на компоненте.

*OnDblClick* – двойное нажатие левой кнопки мыши на компоненте.

*OnMouseDown* – нажатие кнопки мыши на компоненте.

*OnMouseUp* – отпускание кнопки мыши.

*OnMouseMove* – перемещение мыши над компонентом.

*OnKeyPress* – нажатие и отпускание кнопки на клавиатуре.

*OnKeyDown* – нажатие кнопки на клавиатуре.

*OnKeyUp* – отпускание кнопки на клавиатуре.

*OnEnter* – получение фокуса компонентом.

*OnExit* – потеря фокуса компонентом.

*Width* – ширина экрана в пикселах.

***Лабораторная работа***

***Программирование алгоритмов линейной и разветвляющейся структуры***

1.Выполнить задания, приведенные ниже как примеры.

2.Выполнить задание своего варианта.

***Пример******программы алгоритма линейной структуры****:*

**1.Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

Label1 – используется для текста “Введите t, x”.

Label2 – используется для текста “Вычисление у”. Button1 – кнопка, при нажатии которой производятся вычисления.

Код вычислений прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Button2 – кнопка, при нажатии которой форма закрывается.

Edit1 – однострочный редактор для ввода t.

Edit2 – однострочный редактор для ввода x.

Edit3 – однострочный редактор для вывода у.

Для перевода строковых данных в число использовалась функция StrToFloat. Для перевода результатов вычислений в строку была использована функция FormatFloat, позволяющая перевести число с плавающей запятой в требуемый формат в строковом виде.

**2.Условие:**

Вычислить:

y

=

*b*

*t*

*t*

*a*

*t*

*t*

*e*

*at*











4

)

(

при а = 2.3\*

sin

(

t

)

;

b

= 17.6\*

x

3

;

t

=

3;

x

= 1.38.

**3**

**.Форма**



**4. Программа:**

unit Unit1; interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls; type

TForm1 = class(TForm)

Label1: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Button1: TButton;

Label2: TLabel;

Edit3: TEdit; 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 t, x, a, b, y:real;

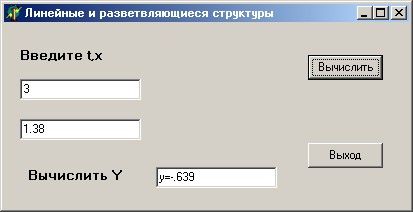
begin t:=strtofloat(edit1.Text); //*Переводит строку в число* x:=strtofloat(edit2.Text);//*Переводит строку в число*  a:=2.3\*sin(t); b:=17.6\*x\*x\*x;

y:=(exp(-a\*t)\*(t+sqrt(abs(t+a))))/(t-(sqrt(abs(4\*t-b)))); edit3.text:=('y='+formatfloat('##.###',y)); end;

procedure TForm1.Button2Click(Sender: TObject); begin

Application.Terminate; end; end.

**5. Результат:**



***Пример******программы алгоритма разветвляющейся структуры****:*

**1. Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

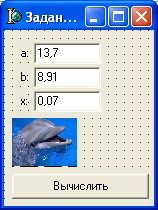
TLabel – используется для создания комментариев и вывода результата вычислений. Для этого использовалось свойство компонента Caption, которое имеет строковый тип. Для перевода результатов вычислений в строку была использована функция FormatFloat, позволяющая перевести число с плавающей запятой в требуемый формат в строковом виде.

TImage – предназначен для просмотра изображения. Для размещения изображения используется свойство компонента Picture.

TButton – кнопка, при нажатии которой производятся вычисления. Код вычислений прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Комментарий на кнопке прописывается в свойстве кнопки Caption.

TEdit – компонент предназначенный для ввода строковых данных. В свойстве Text данного компонента при создании программы были установлены необходимые значения переменных. Для перевода строковых данных в число использовалась функция StrToFloat.

**2.Форма:**



TButton

TLabel

TEdit

TImage

**3.Программа:**

interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, math, jpeg, ExtCtrls; type

TForm1 = class(TForm)

Edit1: TEdit;

Label1: TLabel;

Label2: TLabel;

Edit2: TEdit;

Label5: TLabel;

Button1: TButton;

Label7: TLabel;

Edit3: TEdit; Image1: TImage;

procedure Button1Click(Sender: TObject); private

{ Private declarations } public

{ Public declarations } end; var

Form1: TForm1; implementation {$R \*.dfm}

procedure TForm1.Button1Click(Sender: TObject); var a,b,x,y:real; begin

Label5.Caption:=''; a:=StrToFloat(Edit1.Text); b:=StrToFloat(Edit2.Text); x:=StrToFloat(Edit3.Text); if (sqrt(a-b)<x) then y:=a\*power(x,3)+b\*log10(2\*x); if (sqrt(a-b)=x) then y:=sqrt(abs(a+sin(2\*x))); if (sqrt(a-b)>x) then y:=arctan(5\*x)-b\*power(cos(x),3); Label5.Caption:='y='+FormatFloat('0.000',y); end; end.

**4.Результат:**



***Организация программ циклической структуры***

1.Выполнить задание, приведенное ниже как пример.

2.Выполнить задание своего варианта, из литературы[1] (стр.14-23)

***Пример программы циклической структуры :***

**1.Конструирование формы:**

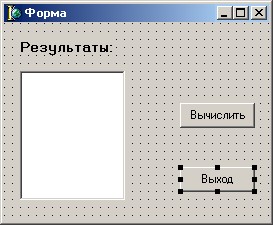
При выполнении задания на форме были размещены следующие компоненты:

TLabel – используется для создания комментариев и вывода результата вычислений. Для этого использовалось свойство компонента Caption, которое имеет строковый тип. Для перевода результатов вычислений в строку была использована функция FormatFloat, позволяющая перевести число с плавающей запятой в требуемый формат в строковом виде.

TButton – кнопка, при нажатии которой производятся вычисления. Код вычислений прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Комментарий на кнопке прописывается в свойстве кнопки Caption.

TListBox – используется для вывода результатов программы.

**2.Форма:**



**3.Программа:**

unit Unit1;

interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls; type

TForm1 = class(TForm)

Label1: TLabel;

ListBox1: TListBox;

Button1: TButton; Button2: TButton; procedure Button1Click(Sender: TObject); procedure Button2Click(Sender: TObject); private

{ Private declarations } public

{ Public declarations } end; var

Form1: TForm1; implementation {$R \*.dfm}

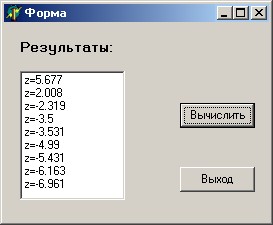
procedure TForm1.Button1Click(Sender: TObject); const a=3.4; var x,y,z:real; begin x:=4; while x<=8 do begin y:=x\*x\*x-exp(x);

if y<1 then z:=cos(y\*y)\*cos(y\*y)\*cos(y\*y)\*cos(y\*y)-ln(abs(y)/ln(10)); if y>=1 then z:=sqrt(abs(a\*y+sin(a)/cos(a))); listbox1.Items.Add('z='+formatfloat('##.###',z)); x:=x+0.5; end; end;

procedure TForm1.Button2Click(Sender: TObject); begin

application.Terminate; end; end.

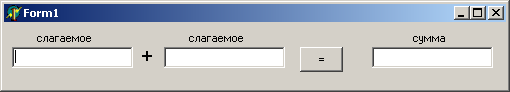
**4.Результат:**



**Пример выполнения**

**Задание №1**

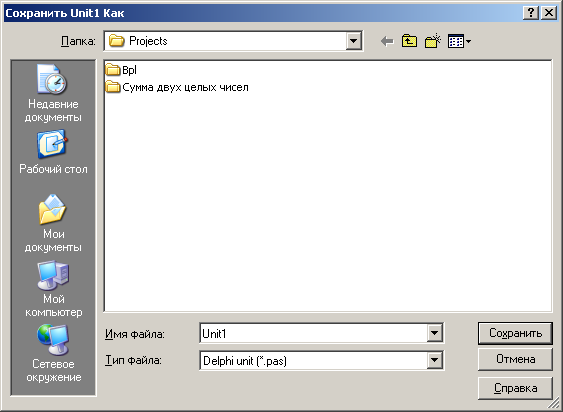
**Создать приложение, которое обеспечивает ввод двух целых чисел, по щелчку на кнопке с символом «=» вычисляет их сумму и выводит значение результата.**



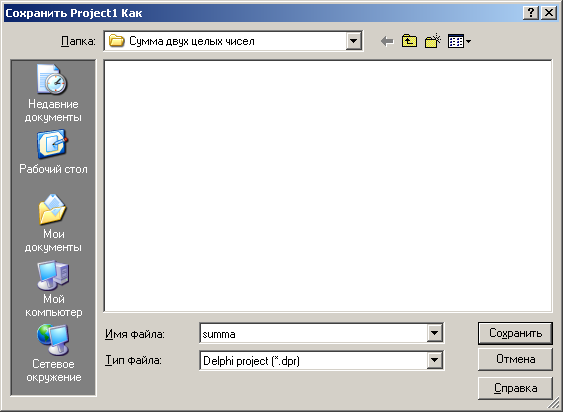
Вид окна приложения

**Алгоритм выполнения работы**

1. После запуска ИСР Delphi создайте новый проект при помощи команды Файл ► Создать ► Приложение. В Delphi 6 при запуске ИСП форма создается автоматически. В результате создания проекта приложения в окне Delphi будет раскрыто окно формы, на которой можно размещать визуальные компоненты проекта. Сохраните новый проект, для чего выберите команду Файл (File) ► Сохранить проект Как (Save Project As). В окне Сохранить Unit1 Как создайте новую папку для файлов создаваемого проекта, например, «Сумма двух целых чисел», как показано на рис. 11.



**Рис. 11.** Создание папки для нового проекта



**Рис. 12**. Сохранение проекта.

Затем в окне Сохранить Unit1 Как откройте созданную папку и нажмите кнопку Сохранить. После сохранения файла модуля Unit1.pas откроется окно Сохранить проект Как (Save Project As) (рис. 12). Задайте имя файла проекта, например, «summa» и нажмите кнопку **Сохранить**.

2. Измените свойства формы. Для изменения размеров формы захватите угол окна формы и, не отпуская левую кнопку мыши, перемещайте мышь, задавая требуемый размер формы.

3. Измените надпись в заголовке формы Form1 с помощью Инспектора объектов. Инспектор объектов обеспечивает простой и удобный интерфейс для изменения свойств объектов Delphi и управления событиями, на которые реагирует объект. Окно Инспектора объектов состоит из двух страниц, каждую из которых можно использовать для определения поведения компонента. Первая страница называется Свойства (Properties), вторая — События (Events). Над ними располагается раскрывающийся список всех компонентов, размещенных на форме.

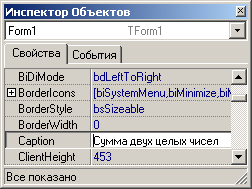
В нем можно выбрать тот компонент, свойства и события которого вас интересуют. Страница Свойства (Properties) Инспектора объектов (рис. 13), показывает свойства того объекта, который в данный момент выделен. Для переключения между страницами свойств и событий используются закладки Свойства (Properties) и События (Events) в верхней части окна Инспектора объектов.

При щелчке на некоторых свойствах, например, на свойстве Color, справа от имени свойства открывается окно раскрывающегося списка. Нажав в нем кнопку со стрелкой вниз, можно увидеть список возможных значений этого свойства. Список значений можно просмотреть с помощью бегунка. Например, если изменить значение свойства Color с принятого по умолчанию clBtnFace (цвет поверхности кнопок) на clWmdow (цвет окна), поверхность формы изменит свой цвет.

Рядом с некоторыми свойствами можно заметить знак плюс (см., например, свойство Font на рис. 16). Это означает, что данное свойство является объектом, который в свою очередь имеет ряд свойств. После щелчка на этом плюсе или двойного щелчка на свойстве Font откроется таблица таких свойств, как Color (цвет), Height (высота), Name (имя шрифта) и др. Среди них есть свойство Style (стиль), около которого тоже имеется знак плюса. Щелчок на этом плюсе или двойной щелчок на этом свойстве раскроет дополнительный список подсвойств, в котором можно, например, установить в значение true для свойства fsBold (жирный). Кстати, для смены true на false и обратно в подобных булевых свойствах не обязательно выбирать значение из раскрывающегося списка. Для его изменения достаточно двойного щелчка на значении свойства.

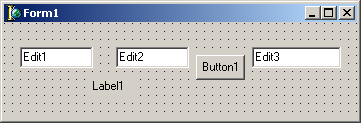
После просмотра или изменения подсвойств можно опять произвести двойной щелчок на головном свойстве или щелчок на знаке минус около него, и список подсвойств будет закрыт.

Для изменения заголовка формы Form1 в окне Инспектора объектов откройте страницу Свойства, затем выберите свойство Caption (Заголовок) и задайте новое значение этого свойства, например, «Сумма двух целых чисел», как показано на рис. 13.



**Рис. 13.** Изменение свойства Caption формы Form1

4. Разместите на форме компоненты Edit1, Edit2, Edit3, Label1, Button1, как показано на рис. 14.



**Рис. 14.** Форма с размещенными на ней компонентами

Для размещения компонентов на форме, щелкнув на вкладке Standard, откройте палитру с требуемыми компонентами, затем, щелкнув на значке соответствующего компонента, например, Edit, щелкните в окне формы. Задайте положение и размер компонента при помощи мыши.

**СОВЕТ**

Если вы забыли, на какой странице палитры расположен конкретный компонент, выберите команду Вид (View) ► Список компонентов (Component List) и на экране появится список компонентов в алфавитном порядке. Выбрав в окне Компоненты (Components) нужный компонент, нажмите кнопку Добавить для размещения компонента на форме.

**ПРИМЕЧАНИЕ**

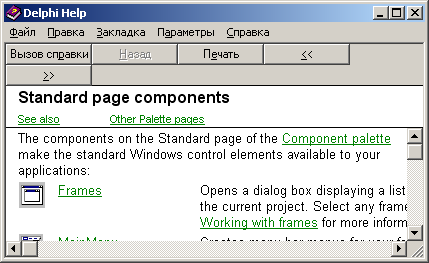
Есть и другой способ поместить компонент на форму — достаточно сделать двойной щелчок на значке компонента, и он будет автоматически помещен в центр формы. Если вы выбрали компонент, а затем решили не размещать его, достаточно нажать кнопку указателя. Это прервет процесс размещения компонента и программа вернется в нормальный режим, в котором можно выбрать другой компонент или выполнить какую-либо команду.

*Получение справки о назначении компонентов*

*Имена компонентов, соответствующих значкам на палитре, можно узнать из ярлычка, появляющегося, если задержать над этим значком курсор мыши.*

***ПРИМЕЧАНИЕ*** *Имена на ярлычках выглядят так: MainMenu, Button и т. д. Однако в Delphi все имена классов в действительности начинаются с символа «Т», например, TMainMenu, TButton. Под такими именами во встроенной в Delphi справочной системе можно найти описания соответствующих компонентов.*

*Назначение компонентов можно уточнить, используя систему контекстной помощи Delphi. Для этого следует выделить нужную страницу в палитре компонентов и нажать клавишу F1. После этого откроется окно Delphi Help со справочной информацией. Например, окно справки о компонентах страницы Standard выглядит следующим образом:*

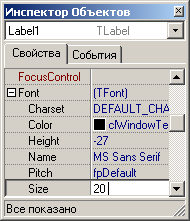
**

***Рис. 15.*** *Окно справочной информации о компонентах страницы Standard*

*Для просмотра информации в окне используется полоса прокрутки. Если нужно просмотреть справку о конкретном компоненте, то следует выбрать нужную ссылку. Если выбрать в палитре компонент и нажать F1, то будет показана справка по типу данного компонента. Нажатие кнопки Закрыть в окне Delphi Help приводит к закрытию окна со справочной информацией.*

5. Задайте свойство Caption компонента Label1 «+».

Измените размер символов компонента Label1, для чего в окне Инспектора объектов выберите в списке объект Label1, затем на странице Свойства (Properties) разверните список свойств Font. В списке свойств Font выберите свойство Size и задайте для него значение 20, как показано на рис. 16.



**Рис. 16.** Изменение значения

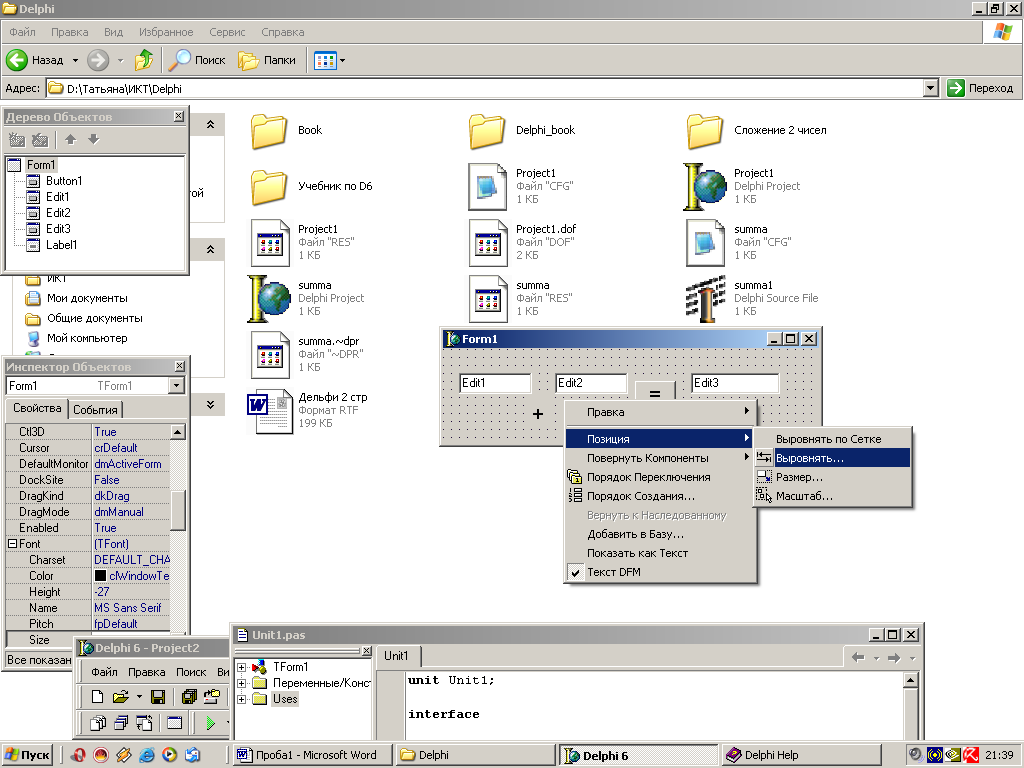
свойства Font.Size

Аналогичным образом задайте свойство Caption компонента Button1 «=» и размер символов, равный 20.

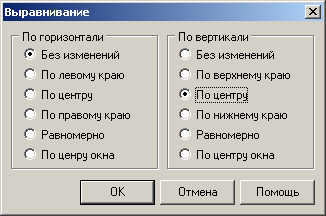
***ПРИМЕЧАНИЕ***

Помимо главного меню в Delphi имеется система контекстных раскрывающихся меню, которые появляются при щелчке правой кнопкой мыши на каком-либо компоненте, как показано на рис. 17. Большинство разделов этих контекстных меню дублируют основные разделы главного меню.

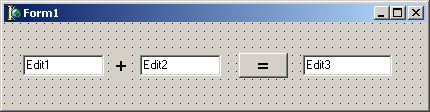
6. Для выравнивания компонентов на форме при нажатой клавише Shift выделите компоненты и правой кнопкой мыши выберите в контекстном меню команду Позиция (Position) ► Выровнять (Align).



**Рис. 17.** Выравнивание компонентов

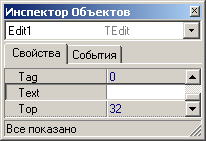


В окне Выравнивание (Alignment) выберите в панели По вертикали (Vertical) вариант По центру (Centers) и нажмите кнопку ОК. Все компоненты, участвующие в операции выравнивания, будут выровнены по вертикали, как показано на рис. 18.



**Рис. 18.** Результат выравнивания компонентов на форме

7. Удалите текст Edit1, Edit2, Edit3 в соответствующих компонентах. Для этого выберите объект в окне Инспектора объектов, на странице Свойства выберите свойство Text и удалите текст, как показано на рис. 19.



**Рис. 19.** Удаление текста в Edit1.Text

***ПРИМЕЧАНИЕ***

При работе с Инспектором объектов можно получить контекстную справку по свойствам или событиям. Для этого следует выделить в окне Инспектора объектов интересующее свойство или событие и нажать клавишу F1, а затем просмотреть в окне Delphi Help справочную информацию.

8. Добавьте на форму три объекта Label, расположите их над объектами Edit1-Edit3 и задайте их свойствам Caption значения «Слагаемое», «Слагаемое» и «Сумма».

9. Нажатием F12 активизируйте окно Редактора кода. Обратите внимание, что в разделе описания программного модуля ИСР были сгенерированы описание формы и размещенных на ней компонентов.

type

TForm1 = class(TForm)

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Label1: TLabel;

Button1: TButton;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

**СОВЕТ**

10. Сохраните изменения, внесенные в проект, нажатием кнопки Сохранить все (Save All) в стандартной панели инструментов.

11. Откомпилируйте созданный проект командой Проект (Project) ► Компилировать summa (Compile summa) (слово «summa» указывает имя проекта).

12. Запустите программу на выполнение при помощи команды Выполнить (Run) ► Выполнить или нажатием кнопки Выполнить (Run) в панели инструментов или клавиши F9.

После этого на экране компьютера появится окно созданной вами формы с компонентами, в которые можно вводить значения. Обратите внимание, что окно формы имеет стандартные атрибуты окна Windows, его можно минимизировать, развернуть на весь экран, изменить его размеры, но приложение не будет выполнять каких-либо вычислений, так как выполняющий вычисления фрагмент программы не был создан. Завершите работу приложения любым из стандартных способов, например, нажатием Alt+F4.

13. Для того чтобы приложение выполняло вычисления при щелчке на кнопке Button1 с изображением символа «=», следует написать код обработки этого события. Для создания кода обработки события следует воспользоваться Инспектором объектов. В окне Инспектора объектов на странице События указаны все события, на которые может реагировать выбранный объект. Страница событий связана с Редактором кода следующим образом: если дважды щелкнуть мышью справа от какого-либо пункта, то соответствующий данному событию код будет автоматически помещен в окно Редактора кода, а окно Редактора кода немедленно получит фокус, и вы сразу же будете иметь возможность отредактировать код обработчика данного события. Например, если требуется выполнить определенные действия при щелчке левой кнопкой мыши по данному объекту, то следует выделить событие OnClick.

Рядом с именем этого события откроется окно с раскрывающимся списком. Если в приложении уже были созданы какие-нибудь обработчики событий и при событии OnClick требуется использовать один из них, можно выбрать необходимый обработчик из раскрывающегося списка.

Так как в приложении еще нет обработчика событий, требуется написать новый обработчик щелчка на кнопке Buttonl с изображением символа «=». Для этого следует выбрать в окне Инспектора объектов объект Buttonl, затем на странице Событий произвести двойной щелчок на пустом поле списка в событии OnClick.

После этого окно Редактора кода немедленно получит фокус. В этом окне в разделе interface находится запись процедуры обработчика события:

procedure TForm1.Button1Click(Sender: TQbject);

В разделе implementation располагается текст заготовки этой процедуры:

procedure TForm1.Button1Click(Sender: TObject);

begin

end;

Курсор будет находиться в пустой строке между ключевыми словами **begin** и **end.** Этот код является заготовкой для обработчика события, которую автоматически создала ИСР Delphi.

Вам остается только разместить необходимые операторы в пространстве между begin и end. Так как в задаче требуется выполнить сложение двух целых чисел, а результат их сложения также является целым числом, то в разделе описания переменных следует ввести следующее описание:

var

a,b,c: integer; {2 слагаемых и сумма - целые числа}

Так как в приложении для ввода чисел-слагаемых используются окна редактирования Edit1**,** Edit2,то необходимо при помощи процедуры StrToInt преобразовать строки из окон Edit1, Edit2в целые числа. Для вывода результата суммирования в окне редактирования Edit3 нужно преобразовать число в строку функцией IntToStr**.** Поэтому в основное тело процедуры обработки события следует ввести следующий текст:

a:=StrToInt(Edit1.text);

b:=StrToInt(Edit2.text);

c:=a+b;

Edit3.text:=IntToStr(c);

Целиком процедура обработки события щелчка на кнопке **Button1** будет выглядеть следующим образом:

procedure TForm1.Button1Click(Sender: TObject);

var

a,b,c: integer; {2 слагаемых и сумма - целые числа}

begin

a:=StrToInt(Edit1.text); {преобразование текстовой строки в целое число}

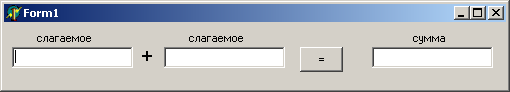
b:=StrToInt(Edit2.text);

c:=a+b;

Edit3.text:=IntToStr(c); {преобразование целого числа в текстовую строку}

end;

14. Запустите приложение на выполнение нажатием кнопки Выполнить в панели инструментов. В окне приложения введите значения слагаемых и, нажав кнопку Button1 с надписью «=», запустите процедуру обработки события — вычисление суммы. Окно созданного приложения будет выглядеть следующим образом:



**Рис. 20.** Вид окна приложения

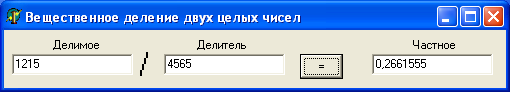
**Дополнительное задание 1.**

Измените приложение таким образом, чтобы выполнялась операция вычитания.

**Пример выполнения**

**Задание №2**

**Создайте приложение, реализующее ввод двух целых чисел, по щелчку на кнопке с символом «=» вычисляющее результат операции вещественного деления и выводящее значение результата на экран.**



Окно приложения, выполняющего операцию вещественного деления

**Алгоритм выполнения работы**

**1.** Запустите ИСР Delphi и создайте новый проект командой Файл ► Создать ► Приложение. В результате создания проекта в окне Delphi будет раскрыто окно формы, на которой вы будете размещать визуальные компоненты проекта. Сохраните новый проект при помощи команды Файл ► Сохранить проект Как. В окне Сохранить Как создайте новую папку для файлов создаваемого проекта, например, «Вещественное деление». Затем в окне Сохранить Как откройте созданную папку, задайте в поле Имя файла имя main и щелкните на кнопке Сохранить. После сохранения файла модуля main.pas откроется окно Сохранить проект Как, в котором следует задать имя файла проекта, например, delenie, а затем нажать кнопку Сохранить.

2. Измените свойства формы. Для изменения размеров формы захватите угол окна формы и, не отпуская левую кнопку мыши, перемещайте мышь, задавая требуемый размер формы.

3. Измените надпись в заголовке формы Form1 с помощью Инспектора объектов. Для этого в окне Инспектора объектов выберите объект Form1, откройте страницу Свойства, затем выберите свойство Caption (Заголовок) и задайте новое значение этого свойства, например, «Вещественное деление двух целых чисел».

4. Разместите на форме компоненты Edit1, Edit2, Edit3, Label1, Label2, Label3, Label4 и Button1, как показано на рис. 22.

5. Удалите текст Edit1, Edit2, Edit3 из соответствующих компонентов, для чего выберите объект в окне Инспектора объектов, на странице Properties выберите свойство Text .

6. Задайте для свойства Caption компонента Label1 значение «/».

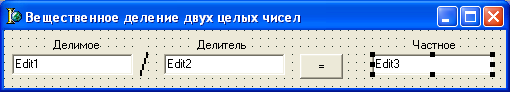
7. Измените размер символов компонента Label1, для чего в окне Инспектора объектов выберите в списке объект Label1, затем на странице Свойства разверните список свойств Font.

В списке свойств Font выберите свойство Size и задайте для него значение 20, как показано на рис. 16. Аналогичным образом задайте для свойства Caption компонента Button1 значение «=» и размер символов 20.

8. Задайте для свойства Caption компонента Label2 значение «Делимое», для Label3 — «Делитель», для Label4 — «Частное».

9. Зафиксируйте положение компонентов на форме, выбрав команду Правка ►Зафиксировать.

После редактирования свойств визуальных компонентов форма приложения будет выглядеть следующим образом:



**Рис. 22**. Вид формы приложения с визуальными компонентами

10. Нажав F12, откройте окно Редактора кода и просмотрите текст модуля, сгенерированный Delphi в процессе визуального проектирования формы. Обратите внимание, что в тексте программы еще не описаны переменные и нет процедуры вычисления операции вещественного деления.

11. Чтобы приложение выполняло вычисления при щелчке мышью на кнопке Button1 с изображением символа «=», следует написать код обработки этого события. Для создания кода обработчика события следует воспользоваться помощью Инспектора объектов. Выберите в его окне объект Button 1, затем на странице События дважды щелкните мышью на пустом поле списка в событии OnClick. После этого окно Редактора кода немедленно получит фокус, в котором в разделе interface появится запись процедуры обработчика события:

procedure TForm1.Button1Click(Sender: TObject);

В разделе implementation появится текст заготовки этой процедуры:

procedure TForm1.Button1Click(Sender: TObject);

begin

...

end;

Курсор будет расположен в пустой строке между ключевыми словами **begin** и **end.** Этот код является заготовкой обработчика события, которое будет выполняться при щелчке мышью на кнопке **Button1**. Остается только разместить в промежутке между **begin** и **end** необходимые операторы. Так как в задаче необходимо выполнить деление двух целых чисел, а результат их вещественного деления всегда будет вещественным числом, то в разделе описания переменных следует ввести следующее описание:

**var**

**a, b : integer**; {2 операнда - целые числа}  
**с : real**; {частное - вещественное число}

В тело процедуры обработки события введите текст:

**a:=StrToInt(Edit1.text);** {преобразование текстовой строки в целое число}

**b:=StrToInt(Edit2.text);**  
**с:=а/b;** {выполнение операции вещественного деления и присваивание

результата переменной с}

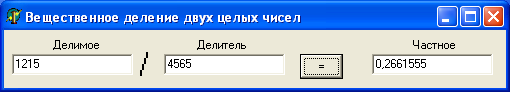
**Edit3.text:=FloatToStrF(c,ffGeneral,7,4);** {преобразование вещественного числа в текстовую строку}

*ПРИМЕЧАНИЕ*

Поскольку свойства text объектов Editl, Edit2 имеют строковые значения, для их пре­образования в целые числа используется стандартная функция Object Pascal — StrToInt. Так как свойству Edit3.text нужно присвоить значение вещественного частного, то для преобразования вещественного числа в текстовую строку используется функция FloatToStrF.

12. Для получения справочной информации о синтаксисе функции FloatToStrF укажите курсором название функции и нажмите F1. В окне справки описано назначение параметров этой функции.

13. Откомпилируйте и запустите программу на выполнение при помощи команды Выполнить ► Выполнить или щелчка мышью на кнопке Выполнитьв панели инструментов или нажатием клавиши F9. После этого на экране компьютера появится окно созданной вами формы с компонентами Edit1, Edit2**,** в которые можно вводить значения, как показано на рис. 23.



**Рис. 23.** Окно приложения, выполняющего операцию вещественного деления

При щелчке на кнопке Button1 «=» обработчик этого события будет выполнять операцию вещественного деления двух чисел и выводить частное в окне Edit3.

14. Завершите работу приложения любым из стандартных способов, например, щелчком мышью на кнопке **Закрыть** в правом верхнем углу окна.

**Дополнительное задание 1.**

Измените программу обработчика события таким образом, чтобы перед операцией деления выполнялась проверка делителя на равенство нулю. В случае равенства делителя нулю вместо выполнения деления в окне Edit3 должно отображаться сообщение «На ноль делить нельзя!».

Вставьте в процедуру обработчика события **TForm1.Button1Click** оператор **if then else**, а в качестве условия выполнения операции вещественного деления задайте b<>0. Фрагмент программы, проверяющий условие отличия делителя от нуля и выполняющий деление, если условие выполняется, может быть записан следующим образом:

**var**

**a,b : integer;** {2 операнда - целые числа}  
**с : real;** {частное - вещественное число}

**begin**

**a:=StrToInt(Edit1.text);** {преобразование текстовой строки в целое число}

**b:=StrToInt(Edit2.text);**

**if b<>0 then** {проверка отличия делителя от 0}

**begin**

**с:=а/b;** {выполнение операции вещественного деления и

присваивание результата переменной с}

**Edit3.text:=F1oatToStrF(c,ffGeneral,7,4);**

{преобразование вещественного числа в текстовую строку}

**end**

**else** {вывод сообщения}

**Edit3.text: = 'На ноль делить нельзя!'**

**end;**

Откомпилируйте модифицированную программу командой Проект (Project) ► Компилировать (Compile). Запустите программу на выполнение при помощи команды Выполнить ► Выполнить или щелчка мышью на кнопке Выполнитьв панели инструментов или нажатием клавиши F9. Проверьте работу приложения, задавая в качестве делителя разные значения.

**Дополнительное задание 2.**

Отредактируйте текст модуля таким образом, чтобы перед выполнением вычислений выполнялась проверка, заданы ли значения операндов. Если значения операндов не заданы, то следует вывести сообщение об этом в отдельном окне.

Поместите перед оператором присваивания **a:=StrToInt(Edit1.text);** строку **if (Edit1.Text<>") and (Edit2.Text<>") then begin**

которая проверяет значения свойств Edit1.Text и Edit2.Text.

Если эти значения не пустые, то выполняется вычисление арифметической операции, в противном случае управление передается на следующий фрагмент программы, который нужно вставить перед последним оператором **end;**

**end;**

**else ShowMessage('He заданы значения');**

Сохраните, откомпилируйте и запустите приложение на выполнение. Проверьте работу приложения для случая, когда не заданы значения операндов и убедитесь, что в этом случае выводится сообщение в отдельном окне. Попробуйте ввести в качестве значений операндов не цифры, а символы, например, «а» и «б». Щелкнув на кнопке Button1, убедитесь в том, что приложение завершается и выдает сообщение об ошибке, которая возникает из-за того, что приложение пытается преобразовывать символы в число.

Для предупреждения данной ошибки введем обработку события нажатия на клавишу в окне Edit1 и Edit2, чтобы запретить ввод любых символов, кроме цифр от 0 до 9 и знаков **-, +**.

Для создания процедуры обработчика события нажатия на клавишу в окне Edit1 выберите в окне Инспектора объектов компонент Edit1 и на странице События дважды щелкните левой кнопкой мыши на пустом поле списка в событии OnKeyPress. После этого окно Редактора кода немедленно получит фокус и в разделе interface появится запись процедуры обработчика события

procedure Edit1KeyPress(Sender: TObject: var Key:Char);

Параметр Key в обработчике этого события соответствует символу нажатой клавиши.

В разделе implementation появится текст заготовки этой процедуры:

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key:Char);

begin

**end;**

Вставьте в тело процедуры следующий оператор:

**if not (Key in ['О'..'9','+','-']) then Key:=#0;**

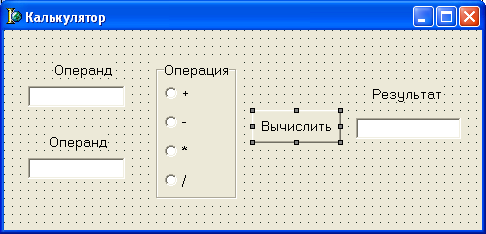
Действие оператора сводится к сравнению значения переменной Key с мно­жеством значений ['0'..'9','+','-']. Если символ нажатой клавиши не входит во множество, то Key присваивается значение нулевого символа (#0). Таким образом, в окне Edit1 будет отображаться текст, состоящий из цифр и знаков «+» и «-». Аналогичным образом создайте процедуру обработчика события нажатия на клавишу в окне Edit2.

Сохраните, откомпилируйте и запустите приложение на выполнение. Попро­буйте ввести в качестве значений операндов не цифры, а символы и убедитесь, что созданные выше обработчики событий. OnKeyPress для Edit1 и Edit2 не позволяют пользователю вести нецифровую информацию. Закройте окно приложения.

**Пример выполнения**

**Задание №3**

**Создайте приложение, обеспечивающее ввод двух целых чисел и выполнения над ними арифметических операций сложения, вычитания, умножения и вещественного деления.**



Вид формы приложения с компонентами

**Алгоритм выполнения работы**

**1.** После запуска ИСР Delphi создайте новый проект при помощи команды Файл ► Создать ► Приложение. В Delphi 6 при запуске ИСП форма создается автоматически. В результате создания проекта приложения в окне Delphi будет раскрыто окно формы, на которой можно размещать визуальные компоненты проекта. Сохраните новый проект, для чего выберите команду Файл (File) ► Сохранить проект Как (Save Project As). В окне создайте новую папку для файлов создаваемого проекта, например, «Калькулятор». Затем в окне Сохранить Unit1 Как откройте созданную папку, - в поле Имя файла задайте имя модуля Main и нажмите кнопку Сохранить. После сохранения файла модуля main.pas откроется окно Сохранить проект Как. В этом окне в поле Имя файла задайте имя файла проекта, например, Calculator и нажмите кнопку Сохранить.

2. Задайте для свойства Form1.Caption значение «Калькулятор».

3. Разместите на форме Form1 компоненты Edit1, Edit2 и Label1, Label2.

4. Задайте для свойства Caption компонентов Label1 и Label2 значения «Операнд». При нажатой клавише Shift выделите компоненты Edit1, Edit2, Label1, Label2 и, вызвав контекстное меню, командой Позиция (Position) ► Выровнять (Align) ► По горизонтали (Horizontal) ► По центру (Centers) выровняйте их по горизонтали.

5. Для выбора одной из четырех арифметических операций над операндами используйте переключатели, размещенные на панели RadioGroup.

*ПРИМЕЧАНИЕ*

Компонент RadioGroup позволяет отображать и редактировать поля с ограниченным множеством возможных значений.

Разместите на форме Form1 компонент RadioGroup1 из палитры Standard.

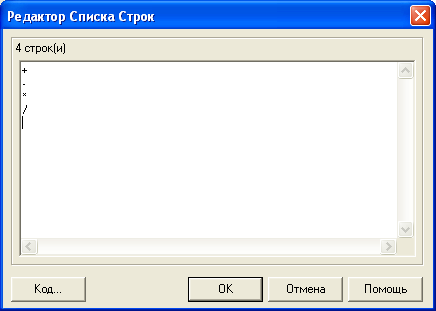
Задайте для свойства Caption компонента RadioGroup1 значение «Операция». Так как количество переключателей в группе и надписи около них определяются свойством **Items**, выберите в Инспекторе объектов компонент RadioGroup1, а на странице свойств выберите свойство Items (список элементов). В окне Редактор Списка Строк введите список элементов — символов арифметических операций: +, - ,\*, /, как показано на рис. 31.

Щелкнув мышью на кнопке ОК, завершите формирование списка арифметических операций.

6. В окне Инспектора объектов задайте для свойства **RadioGroup.Items Index** значение 0, чтобы сделать первую кнопку (сложение) выбранной по умолчанию.

ПРИМЕЧАНИЕ

Определить во время выполнения, какая кнопка выбрана в данный момент, можно по индексу **ItemIndex** (0 означает первую кнопку, -1 — ни одна кнопка не выбрана).



**Рис. 31.** Редактирование списка элементов в панели RadioGroup

7. Задайте размер символов компонента RadioGroup, установив для свойства RadioGroup.Font.Size значение 11 пунктов.

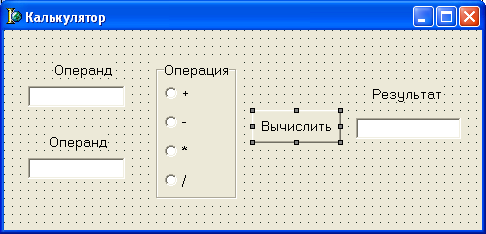
8. Разместите на форме Form1 кнопку Button1 и задайте для нее надпись «Вычислить».

9. Разместите на форме Form1 компоненты Edit3, Label3 и задайте для свойства Label3.Caption значение «Результат».

10. Удалите текст Edit1, Edit2, Edit3 в соответствующих компонентах. Для этого выберите объект в окне Инспекто-ра объектов, на странице Properties выберите свойство Text и удалите текст.

11. Выровняйте компоненты Edit3 и Label3, отцентрировав их по горизонтали.

12. Зафиксируйте положение компонентов на форме, выбрав в меню Delphi команду Правка ► Зафиксировать. Форма будет выглядеть, как показано на рис. 32.



**Рис. 32.** Вид формы приложения с компонентами

13. Для обработки щелчка на кнопке Button1 с изображением символа «=» в окне Инспектора объектов выберите объект Button1, затем на странице События дважды щелкните мышью на пустом поле списка в событии OnClick. После этого в разделе interface модуля Main.pas появится процедура обработчика события procedure TForm1.Button1Click(Sender: TObject);, а в разделе **implementation** появится текст заготовки этой процедуры:

**procedure TForm1.Button1CIick(Sender: TObject);**

**begin**

**end;**

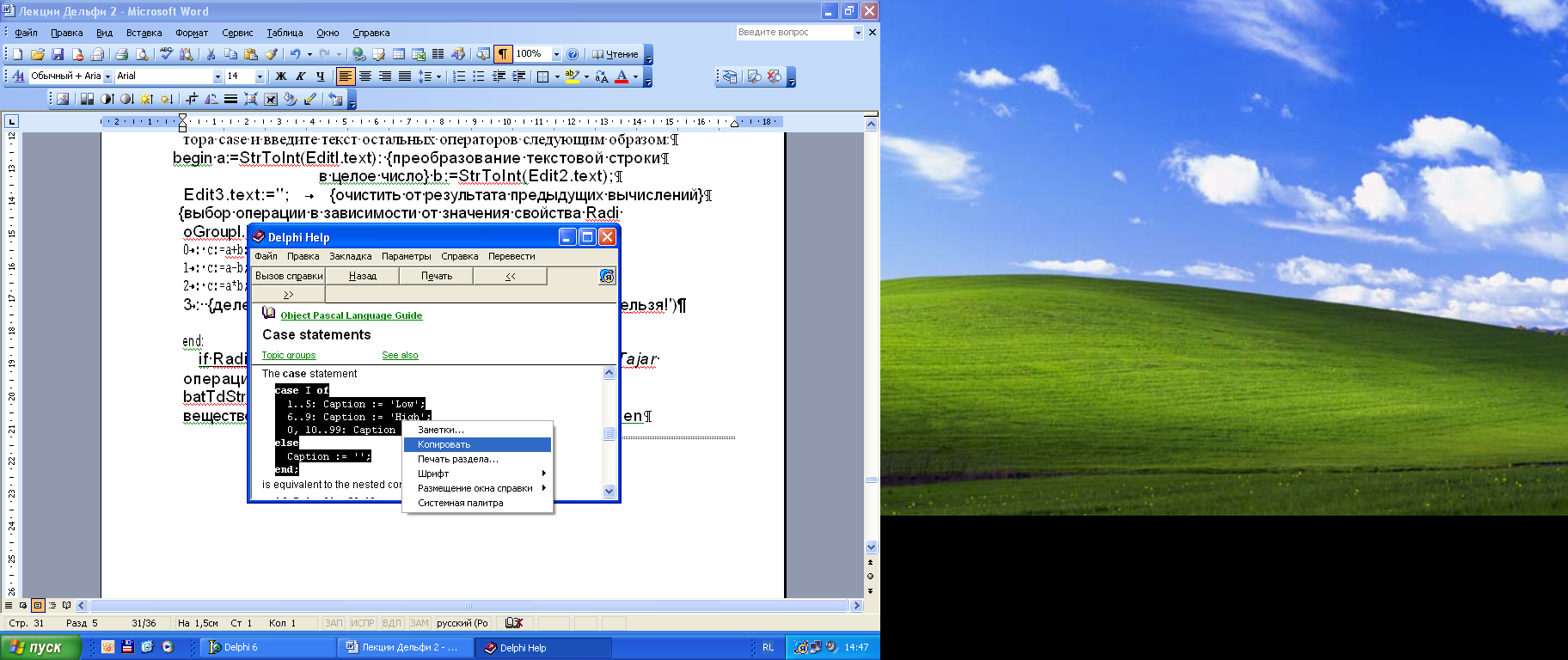
Для решения задачи вычисления арифметических операций над двумя целыми операндами введите в текст процедуры TForm1.Button1Click следующее описание:

**var**

**a,b:integer;** { 2 операнда - целые числа}

**с:real;** {результат арифметических операций}

Так как в процедуре обработки нажатия кнопки Button1 «Вычислить» должно быть 4 варианта реализации, по одному для каждой арифметической операции, то следует записать выбор вычисления с помощью оператора **case**. Для получения подсказки Delphi по синтаксису оператора **case**, введите слово **case,** укажите на него мышью и нажмите F1. В списке найденных разделов выберите раздел **Case statements** и щелкните мышью на кнопке Показать. Для просмотра информации используйте полосу прокрутки в окне **Delphi Help**. Для копирования примера кода программы в окне Delphi Help выделите текст в окне справки и, нажав правую кнопку мыши, вызовите контекстное меню, затем в контекстном меню выберите команду Копировать, как показано на рис. 33



**Рис. 33.** Копирование примера кода программы в окне Delphi Help

В окне Редактора кода укажите место вставки скопированного фрагмента в основном теле процедуры обработки события и, вызвав контекстное меню, выберите команду Paste (Вставить). В основном теле процедуры обработки события отредактируйте текст оператора **case** и введите текст остальных операторов следующим образом:

**begin**

**a:=StrToInt(Edit1.text)**; {преобразование текстовой строки в целое число}

**b:=StrToInt(Edit2.text);**

**Edit3.text:='';** {очистить от результата предыдущих вычислений}

{выбор операции в зависимости от значения свойства RadioGroup1.ItemIndex}

**Case RadioGroup1.ItemIndex of**

1. : **c:=a+b**; {сложение}
2. : **c:=a-b;** {вычитание}
3. : **c:=a\*b;** {умножение}
4. : {деление} **if** b=0 **then** **ShowMessage('На ноль делить нельзя!')**

**else c:=a/b;** **end;**

**if** **RadioGroup1.ItemIndex<> 3** **then** {вывод результата операций}

**Edit3.text:=** **FloatToStrF(c,ffGeneral,10,7)**{преобразование

результата вещественного деления в текстовую строку}

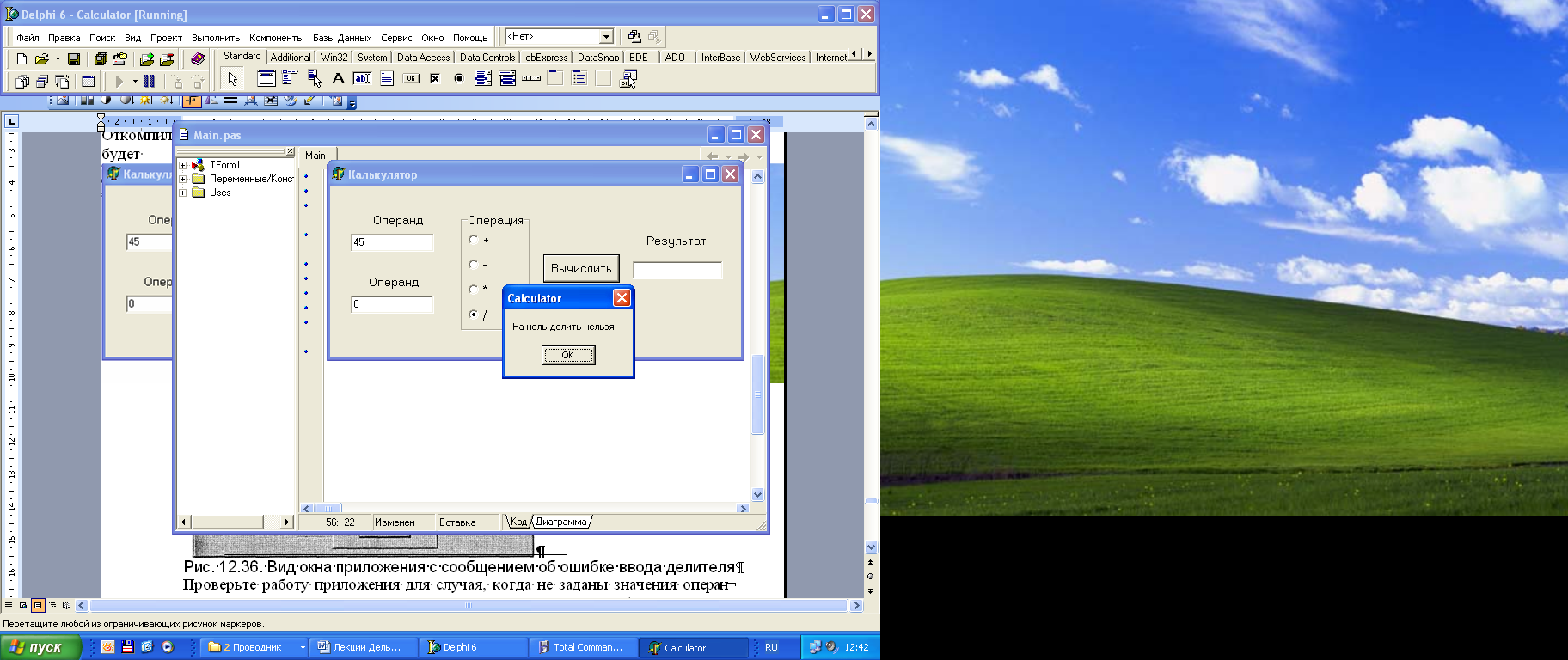
**else if b<>0 then**

**Edit3.text:=FloatToStrF(c,ffGeneral,10,4)**

**end;**

Как видно из текста процедуры, в операторе case выполняется выбор варианта вычислений, а вывод результата вычислений выполняется оператором, следующим после оператора **case,** причем для случая b=0 результат не вычисляется и в окне Edit3 не выводится, а выводится сообщение в окне сообщений.

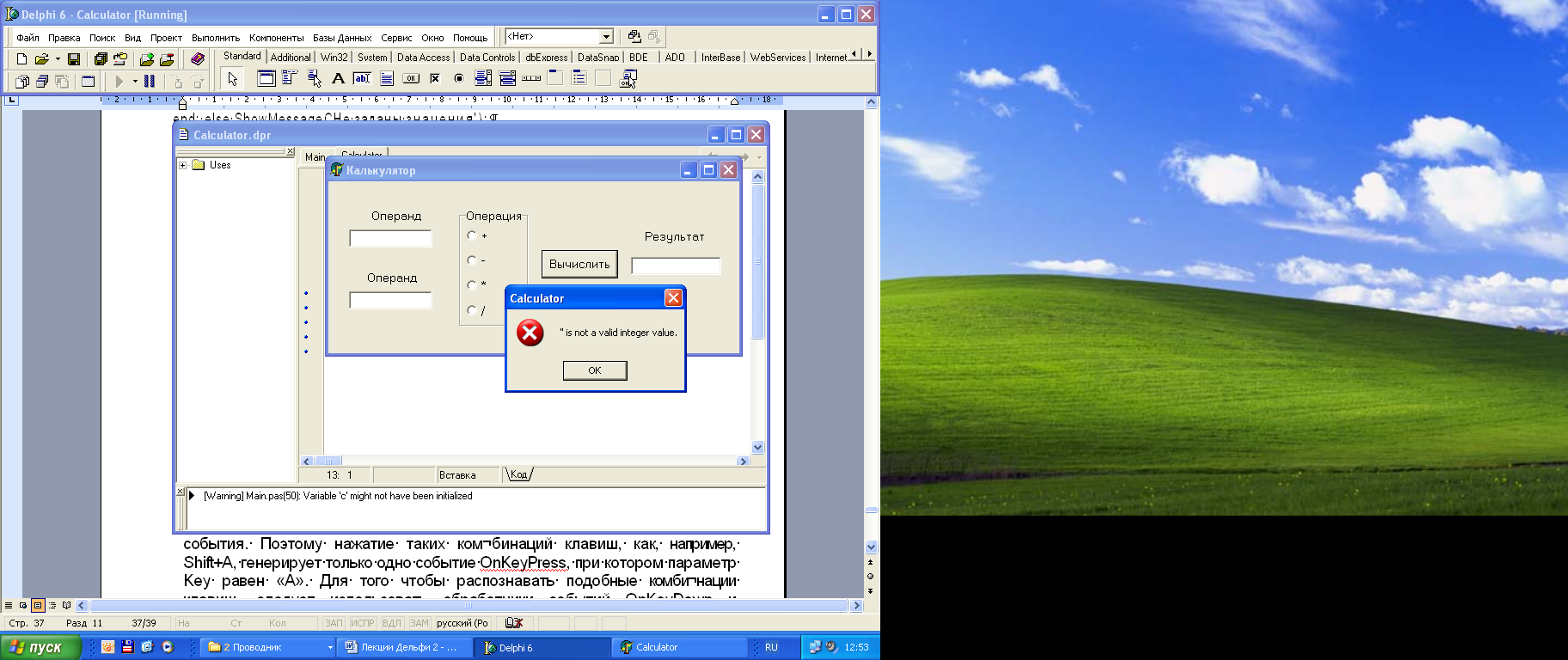
14. Откомпилируйте и запустите приложение на выполнение.



**Рис. 34.** Вид окна приложения с сообщением об ошибке ввода делителя

**Дополнительное задание 1.**

Закройте окно приложения и отредактируйте текст модуля таким образом, чтобы перед выполнением вычислений выполнялась проверка, заданы ли значения операндов. Если значения операндов не заданы, то следует вывести сообщение об этом в отдельном окне.



Окно сообщения о некорректности значения операнда

Поместите перед оператором присваивания **a:=StrToInt(Edit1.text);** строку **if (Edit1.Text<>") and (Edit2.Text<>") then begin**

которая проверяет значения свойств Edit1.Text и Edit2.Text.

Если эти значения не пустые, то выполняется вычисление арифметической операции, в противном случае управление передается на следующий фрагмент программы, который нужно вставить перед последним оператором **end;**

**end;**

**else ShowMessage('He заданы значения');**

Сохраните, откомпилируйте и запустите приложение на выполнение. Проверьте работу приложения для случая, когда не заданы значения операндов и убедитесь, что в этом случае выводится сообщение в отдельном окне. Попробуйте ввести в качестве значений операндов не цифры, а символы, например, «а» и «б». Щелкнув на кнопке Button1, убедитесь в том, что приложение завершается и выдает сообщение об ошибке, которая возникает из-за того, что приложение пытается преобразовывать символы в число.

Для предупреждения данной ошибки введем обработку события нажатия на клавишу в окне Edit1 и Edit2, чтобы запретить ввод любых символов, кроме цифр от 0 до 9 и знаков **-, +**.

Для создания процедуры обработчика события нажатия на клавишу в окне Edit1 выберите в окне Инспектора объектов компонент Edit1 и на странице События дважды щелкните левой кнопкой мыши на пустом поле списка в событии OnKeyPress. После этого окно Редактора кода немедленно получит фокус и в разделе interface появится запись процедуры обработчика события

procedure Edit1KeyPress(Sender: TObject: var Key:Char);

Параметр Key в обработчике этого события соответствует символу нажатой клавиши.

В разделе implementation появится текст заготовки этой процедуры:

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key:Char);

begin

end;

Вставьте в тело процедуры следующий оператор:

**if not (Key in ['О'..'9','+','-']) then Key:=#0;**

Действие оператора сводится к сравнению значения переменной Key с мно­жеством значений ['0'..'9','+','-']. Если символ нажатой клавиши не входит во множество, то Key присваивается значение нулевого символа (#0). Таким образом, в окне Edit1 будет отображаться текст, состоящий из цифр и знаков «+» и «-».

Аналогичным образом создайте процедуру обработчика события нажатия на клавишу в окне Edit2. В итоге получится следующий текст модуля приложения:

**unit Main;**

**interface**

**uses**

**Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms. Dialogs, StdCtrls, ExtCtris;**

**type**

**TForm1 = class(TForm)**

**Edit1: TEdit;**

**Edit2: TEdit;**

**Label1: TLabel;**

**Label2: TLabel;**

**RadioGroup1: TRadioGroup;**

**Button1: TButton;**

**Edit3: TEdit;**

**Label3: TLabel;**

**procedure Button1ClickCSender: TObject);**

**procedure Edit1KeyPress(Sender: TObject: var Key: Char);**

**procedure Edit2KeyPress(Sender: TObject; var Key: Char);**

**private**

{Private declarations}

**public**

{Public declarations}

**end;**

**var**

**Form1: TForm1;**

**implementation** {$R \*.dfm}

**procedure TForm1.Button1Click(Sender: TObject);**

{обработчик события щелчка на кнопке Button1}

**var**

**a,b:integer;** {2 операнда - целые числа.} **с:real;** {результат арифметических операций}

**begin**

**if (Edit1.Text<>’’) and (Edit2.Text<>’’) then**

**begin**

**a:=StrToInt(Edit1.text);** {если значения заданы}

**b:=StrToInt(Edit2.text);** {преобразование текстовой строки

в целое число}

**Edit3.text:='';** {очистить от результата предыдущих вычислений}

{выбор операции в зависимости от значения свойства RadioGroup1.ItemIndex}

**Case RadioGroup1.ItemIndex of**

1. **: c:=a+b;** {сложение}
2. **: c:=a-b;** {вычитание}
3. **: c:=a\*b;** {умножение}
4. **:** {деление} **if b=0 then ShowMessage('Ha ноль делить нельзя!')**

**else c:=a/b;**

**end;**

**if RadioGroup1.ItemIndex <>3 than** {вывод результата опе~~р~~аций}

**Edit3.text:=F1oatToStrF(c,ffGeneral,10,4)**

{преобразование результата вещественного деления в текстовую строку}

**else if b<>0 then**

**Edit3.text:=FloatToStrF(c,ffGeneral,10.4)**

**end**

**else ShowMessage('He заданы значения');**

**end;**

**procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char); begin** {разрешить ввод в Edit1 только цифр}

**if not (Key in ['0'..'9','+','-']) then Key:=#0;**

**end;**

**procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char); begin** {разрешить ввод в Edit2 только цифр}

**if not (Key in ['0'..'9','+','-']) then Key:=#0;**

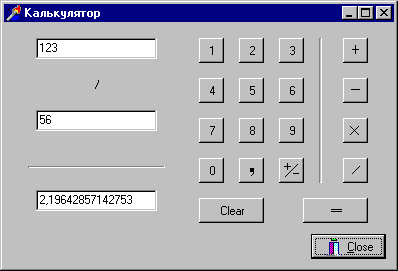
**end;**

**end.**

Сохраните, откомпилируйте и запустите приложение на выполнение. Попро­буйте ввести в качестве значений операндов не цифры, а символы и убедитесь, что созданные выше обработчики событий. OnKeyPress для Edit1 и Edit2 не позволяют пользователю вести нецифровую информацию. Закройте окно приложения.

**Задание 1. Калькулятор**

**Задача:** Создать приложение, выполняющее функции простейшего калькулятора (сложение, вычитание, деление, умножение). Предусмотреть защиту от повторного нажатия на кнопку десятичной запятой и выполнения операции деления на ноль. При повторном нажатии на кнопку знака числа знак вводимого числа должен меняться на противоположный. Кнопка "Clear" очищает все поля ввода. Знак выполняемой операции отражается в окне приложения между полями ввода операндов.

****

Компоненты:

StaticText – окна для ввода чисел и вывода результата;

Label – выполняемая арифметическая операция;

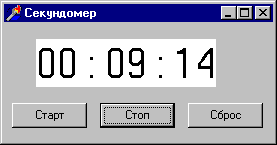
Button – кнопки цифр, операций, очистки (Clear) и получения результата (=);

BitBtn – кнопка закрытия приложения (Close);

Bevel – разделитель для кнопок цифр и кнопок операций, разделитель для операндов и результата.

**Задание 2. Секундомер**

**Задача:** Создать приложение, выполняющее функции секундомера (включить, остановить, очистить показания).

****

Компоненты:

Label – окна для вывода показаний времени (часы, минуты, секунды);

Button – кнопки включения секундомера, остановки и очистки показаний;

Timer – для отсчета времени.

**Лабораторная работа №1**

***Программирование алгоритмов линейной, разветвляющейся и циклической структуры в среде Delphi***

**по дисциплине «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»**

Составитель: Садчиков П.Н.

Разработка и внедрение в учебный процесс новой лабораторной работы с использованием нового приобретенного оборудования и сопроводительных методических рекомендаций для студентов направления подготовки 09.03.02 «информационные системы и технологии» направленности (профиля) «Информационные системы и технологии в строительстве и архитектуре» очной и заочной форм обучения

Печатается в авторской редакции

Подписано к печати \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Уч.-изд. л. \_\_\_\_. Тираж \_\_\_\_\_\_ экз. Заказ №\_\_\_\_\_\_

Отпечатано в Астраханском государственном архитектурно-строительном университете

Государственное автономное Образовательное

учреждение Астраханской области

высшего образования

«АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ АРХИТЕКТУРНО-СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра систем автоматизированного

проектирования и моделирования

**Лабораторная работа №2**

**Визуальное программирование в среде Delphi**

**при обработке текстовой документации**

**по дисциплине «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»**

Разработка и внедрение в учебный процесс новой лабораторной работы с использованием нового приобретенного оборудования

и сопроводительных методических рекомендаций

для студентов направления подготовки

09.03.02 «информационные системы и технологии»

направленности (профиля)

«Информационные системы и технологии в строительстве и архитектуре»

очной и заочной форм обучения

Астрахань 2020

Составитель: к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Садчиков П.Н./

(занимаемая должность, (подпись) И. О. Ф.

учёная степень и учёное звание)

Рецензент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_ /

(занимаемая должность, (подпись) И. О. Ф.

учёная степень и учёное звание)

Инструкция по выполнению лабораторной работы по дисциплине «Технологии программирования» рассмотрена и одобрена на заседании кафедры «Системы автоматизированного проектирования и моделирования» ГАОУ АО ВО «АГАСУ»

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_ 2020 г.

Зав. кафедрой  ***\_\_\_\_\_\_\_\_\_\_\_\_\_*** ***/***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***/***

**(**подпись) (инициалы, фамилия)

Согласовано с УМО ГАОУ АО ВО «АГАСУ»

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Специалист УМУ \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (инициалы, фамилия)

Инструкция по выполнению лабораторной работы по дисциплине «Технологии программирования» утверждена и рекомендована к публикации на заседании методического совета направления подготовки 09.03.02 «информационные системы и технологии» ГАОУ АО ВО «АГАСУ»

Протокол № \_\_\_\_ от \_\_\_\_\_\_\_\_\_\_ 2020 г.

Председатель МКН «Информационные системы и технологии»

направленность (профиль)

«Информационные системы и технологии в строительстве и архитектуре»

\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (инициалы, фамилия)

©Садчиков П.Н.

©ГАОУ АО ВО «Астраханский государственный архитектурно-строительный университет»

**Введение**

**Визуальное программирование в среде Delphi**

Методические указания содержат краткий справочник свойств некоторых наиболее часто используемых в приложениях компонентов среды Delphi, а также комплекс лабораторных работ. Читатель должен быть знаком с принципами объект­но–ориенти­ро­ван­ного програм­ми­ро­ва­ния, языком программирования Object Pascal и основами визуального программирования.

Среда визуального программирования Delphi – это эффективное средство разработки приложений для операционной системы Windows, имеющих стандартный пользовательский интерфейс.

Приложение собирается из готовых элементов, которые называются компонентами. Некоторые из них представляют знакомые вам элементы управления, такие, как кнопка, поле ввода, флажок и т.п. Каждому компоненту соответствует свой класс, являющийся прямым или отдаленным потомком класса TComponent. Библиотека классов компонентов поставляется вместе со средой программирования и носит название библиотеки визуальных компонентов (Visual Component Library, VCL). В ней содержится около сотни стандартных компонентов, обеспечивающих создание самых разнообразных приложений.

**Базовые классы**

Класс **TComponent** является предком всех компонентов VCL. Он реализует основные механизмы, которые обеспечивают функциони­ро­ва­ние любого компонента.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TComponent |
| Название | Тип | Описание |
| Name | String | Имя экземпляра компонента. |
| Tag | Longint | Вспомогательное свойство. |

Класс **TControl** является прямым потомком класса TComponent. Он реализует механизмы отображения компонента на экране и поэтому является предком всех визуальных компонентов.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TControl |
| Название | Тип | Описание |
| Enabled | Boolean | Доступность элемента управления. |
| Top | Integer | Координата левого верхнего угла компонента. |
| Left | Integer | Координата левого верхнего угла компонента. |
| Height | Integer | Высота компонента. |
| Width | Integer | Ширина компонента. |
| ClientHeight | Integer | Высота рабочей области компонента. |
| ClientWidth | Integer | Ширина рабочей области компонента. |
| Align | TAlign | Выравнивание компонента в рабочей области его владельца: alNone – выравнивание отсутствует, alTop, alBottom, alLeft, alRight – выравнивание по соответствующей стороне, alClient – элемент управления изменяет свои размеры до размеров рабочей области владельца. |
| AutoSize | Boolean | Обеспечивает изменение размеров компонента в соответствии с размерами его содержимого. |
| Color | TColor | Цвет (см. ниже). |
| Cursor | TCursor | Изображение указателя мыши при нахождении его над компонентом (см. ниже). |
| Font | TFont | Шрифт (см. ниже). |
| Visible | Boolean | Видимость компонента. |
| Hint | String | Текст ярлыка – подсказки. |
| ShowHint | Boolean | Обеспечивает работу ярлыка. |
| PopupMenu | TPopupMenu | Ссылка на всплывающее контекстное меню. |

Тип данных **TColor** принимает целые значения и определяет цвет.

Цвет можно задавать, используя одну из зарезервированных констант, обозначающих цвет, например clBlack (черный), clRed (красный), или цвета, соответствующие некоторым системным компонентам, например clBtnFace соответствует цвету кнопок.

Для получения оттенков следует воспользоваться функцией RGB, которая имеет три аргумента, каждый из которых задает интенсивность соответственно красного, зеленого и синего цветов, выраженную числом в диапазоне 0–255. Например, RGB(255,255,0) даст желтый цвет.

Можно также задать цвет в виде шестнадцатиричного восьмиразрядного числа. Первые два разряда задают палитру ($00 – системная палитра), следующие два разряда – интенсивность синего цвета, следующие два – зеленого, а последние – красного. (Значение $00FF0000 представляет насыщенный синий цвет, $0000FF00 – насыщенный зеленый, $000000FF – насыщенный красный, $00000000 – черный, $00FFFFFF – белый.)

Класс **TFont** определяет шрифт.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TFont |
| Название | Тип | Описание |
| Charset | Byte | Номер набора символов шрифта: 0 – ANSI, 1 – набор символов, выбираемый по умолчанию, 204 – кириллица. |
| Color | TColor | Цвет шрифта. |
| Height | Integer | Высота шрифта в пикселах. |
| Name | String | Имя шрифта. |
| Size | Integer | Размер шрифта в пунктах. |
| Style | TFontStyle | Стиль шрифта: fsBold – жирный, fsItalic – курсив, fsUnderline – подчеркнутый, fsStrikeOut – перечеркнутый. |

Тип данных **TCursor** определяет изображение указателя мыши, когда он находится над элементом управления и принимает целые значения от   
–32768 до 32767. Для определения изображения можно воспользоваться системными константами. Некоторые формы курсора и соответствующие им константы представлены ниже.

|  |  |
| --- | --- |
| crDefault | изображение курсора, принятое по умолчанию; |
| crNone | отсутствие курсора; |
| crArrow | cursor1 |
| crCross | cursor2 |
| crIBeam | cursor3 |
| crSize | cursor4 |
| crHourGlass | cursor5 |
| crDrag | cursor6 |
| crNoDrop | cursor7 |

Класс **TWinControl** является прямым потомком класса TControl и реализует оконные элементы управления. Это означает, что потомки этого класса смогут выполнять определенные функции, такие, как прием и передача фокуса, отклик на действия мышью, ввод с клавиатуры.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TWinControl |
| Название | Тип | Описание |
| TabOrder | TTabOrder | Принимает значения от –1 до 32767. Определяет порядок передачи фокуса между элементами управления при нажатии клавиши <Tab>. |
| TabStop | Boolean | Определяет реакцию на нажатие клавиши <Tab>. |
| Controls[Index] | TControl | Указатель на другие компоненты (до­чер­ние компоненты), которые содержит дан­ный компонент. |
| ControlCount | Integer | Количество дочерних компонентов. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TWinControl |
| Название | Тип | Описание |
| SetFocus | procedure | Передает фокус элементу управления. |

Класс **TCustomControl** является прямым потомком класса TWinControl и предназначен для создания на его основе нестандартных оконных элементов управления, которые должны уметь нарисовать себя и отобразить какую-либо информацию в своей рабочей области. Эти возможности реализуются с помощью свойства Canvas, имеющего тип TCanvas.

Класс **TCanvas** объединяет в себе "холст", "рабочие инструменты" – перо, кисть, шрифт и "подмастерьев" – набор методов для рисования типовых геометрических фигур.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TCanvas |
| Название | Тип | Описание |
| Font | TFont | Шрифт для текста. |
| Pen | TPen | Перо (см. ниже). |
| Brush | TBrush | Кисть (см. ниже). |
| Pixels[X, Y] | TColor | Цвет пиксела с координатами (X,Y). |
| PenPos | TPoint | Позиция пера:  type TPoint = record  X: Longint;  Y: Longint;  end;. |

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TPen |
| Название | Тип | Описание |
| Color | TColor | Цвет линии. |
| Style | TPenStyle | Стиль линии: psSolid – сплошная, psDash – черточки, psDot – точки, psDashDot – черточка-точка. |
| Width | Integer | Толщина линии в пикселах. |

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TBrush |
| Название | Тип | Описание |
| Color | Tcolor | Цвет заливки. |
| Style | TBrushStyle | Фактура заливки: bsSolid – сплошная, bsHorizontal – горизонтальная штриховка, bsVertical – вертикальная штриховка, bsDiagonal – диагональная штриховка, bsCross – штриховка клеткой. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TCanvas |
| Название | Тип | Описание |
| Ellipse(X1, Y1, X2, Y2: Integer) | procedure | Рисует и закрашивает эллипс, вписанный в прямоугольник, левый верхний угол которого находится в точке с координатами (X1, Y1), а правый нижний – (X2, Y2). |
| Polygon(Points: array of TPoints) | procedure | Строит и закрашивает мно­гоуголь­ник, используя массив Points коор­динат точек, являющихся вер­ши­на­ми многоугольника. |
| Rectangle(X1, Y1, X2, Y2: Integer) | procedure | Рисует и закрашивает пря­моуголь­ник, левый верхний угол которого находится в точке с координатами (X1, Y1), а правый нижний – (X2, Y2). |
| TextOut(X, Y: Integer; Text: String) | procedure | Выводит строку Text, левый верхний угол строки находится в точке с координатами (X, Y). |

Кроме графических примитивов, таких, как линии и фигуры, на канве можно располагать готовые изображения из графических файлов. Класс **TGraphic** является родительским классом для всех видов графики. Его потомки – классы **TMetafile**, **TIcon**, **Tbitmap** соответствуют отдельным видам графических объектов. Класс **TPicture** представляет графический объект любого типа.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TGraphic |
| Название | Тип | Описание |
| Height | Integer | Высота графического объекта. |
| Width | Integer | Ширина графического объекта. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TGraphic |
| Название | Тип | Описание |
| LoadFromFile(FileName:String) | procedure | Загружает рисунок из файла с именем FileName. |
| SaveToFile(FileName:String) | procedure | Сохраняет рисунок в файле с именем FileName. |

Класс **TStrings** предназначен для работы со строковыми списками.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TStrings |
| Название | Тип | Описание |
| Strings[Index] | String | Строка с индексом Index. |
| Objects[Index] | TObject | Объект, связанный со строкой с индексом Index. |
| Count | Integer | Количество строк в списке. |
| Duplicates | TDuplicates | Задает реакцию списка на попытку внести повторяющиеся строки: dupIgnore – повторяющиеся строки игнорируются, dupAccept – ввод повторяющихся строк разрешается, dupError – при вводе повторяющейся строки генерируется исключительная ситуация. |
| Sorted | Boolean | Сортирует строки списка. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TStrings |
| Название | Тип | Описание |
| Add(S:String):Integer | function | Добавляет строку S в конец списка. Возвращает номер новой строки. |
| AddObject(S:String, AObject:TObject): Integer | function | Добавляет строку S в конец списка и соотносит ей объект AObject. Возвращает номер новой строки и соответствующего ей объекта. |
| Delete(Index:Integer) | procedure | Удаляет из списка строку с номером Index. |
| Clear | procedure | Удаляет все строки из списка. |
| IndexOf(S:String): Integer | function | Ищет в списке строку S. Возвращает номер найденной строки или –1, если такой строки в списке нет. |
| Exchange(Index1, Index2:Integer) | procedure | Меняет местами две строки с номерами Index1 и Index2. |
| LoadFromFile(FileName: String) | procedure | Заполняет список строками из файла с именем FileName. |
| SaveToFile(FileName: String) | procedure | Сохраняет строки списка в файле с именем FileName. |

Стандартное приложение содержит одно или несколько окон, которые на этапе разработки представляют собой формы. Формы служат контейнерами для других компонентов. Класс **TForm** является от­дален­ным потомком класса TWinControl и реализует форму.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TForm |
| Название | Тип | Описание |
| Active | Boolean | Активность формы. |
| ActiveControl | TWinControl | Элемент управления, который имеет фокус. |
| BorderIcons | TBorderIcons | Комбинация системных кнопок в заголовке формы: biSystemMenu – кнопка системного меню, biMinimize – кнопка сворачивания, biMaximize – кнопка разворачивания, biHelp – кнопка помощи. |
| Menu | TMainMenu | Главное меню окна. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TForm |
| Название | Тип | Описание |
| Show | procedure | Выводит форму на экран. |
| ShowModal:Integer | function | Выводит форму на экран в модальном режиме. |
| Hide | procedure | Убирает форму с экрана, не унич­то­жая ее. |
| Close | procedure | Закрывает форму. |
| FocusControl(Control:TWinControl) | procedure | Передает фокус элементу управления Control. |
| SetFocus | procedure | Передает фокус форме. |

**Элементы управления**

button **TButton** (отдаленный потомок TWinControl)

Компонент представляет собой кнопку.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TButton |
| Название | Тип | Описание |
| Caption | String | Текст на поверхности кнопки. |

bitbutton **TBitBtn** (отдаленный потомок TWinControl)

Компонент представляет собой кнопку, на поверхности которой может находиться рисунок. Установка свойства Kind компонента приписывает кнопке одну из стандартных операций, таких, как OK, Cancel, Help, Close и т.д., что означает, что кнопка будет иметь стандартный вид и при ее нажатии будет выполнено соответствующее действие.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TBitBtn |
| Название | Тип | Описание |
| Glyph | TBitMap | Рисунок, который появится на поверхности кнопки. |
| Kind | TBitBtnKind | Тип кнопки: bkCustom, bkOK, bkCancel, bkHelp, bkYes, bkNo, bkClose, bkAbort, bkRetry, bkIgnore, bkAll. |
| Layout | TButtonLayout | Расположение рисунка: blGlyphLeft – слева от заголовка, blGlyphRight – справа от заголовка, blGlyphTop – сверху, blGlyphBottom – снизу. |

label1 **TLabel** (отдаленный потомок TControl)

Компонент предназначен для отображения текста.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TLabel |
| Название | Тип | Описание |
| Caption | String | Отображаемый текст. |
| WordWrap | Boolean | Возможность переноса слов, если длина текста превышает ширину компонента. |

edit1 **TEdit** (отдаленный потомок TWinControl)

Компонент предназначен для ввода и редактирования одной строки текста.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TEdit |
| Название | Тип | Описание |
| MaxLength | Integer | Ограничивает максимально допустимое коли­чест­во символов в вводимой строке. |
| ReadOnly | Boolean | Возможность редактирования. |
| Text | String | Строка текста. |
| SelText | String | Выделенный текст. |
| SelStart | Integer | Номер первого символа в выделенном тексте. |
| SelLength | Integer | Длина выделенного текста. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TEdit |
| Название | Тип | Описание |
| Clear | procedure | Удаляет строку из поля ввода. |
| CopyToClipBoard | procedure | Копирует выделенный текст в буфер. |
| PasteFromClipBoard | procedure | Вставляет текст из буфера в позицию курсора или вместо выделенного текста. |
| CutToClipBoard | procedure | Копирует выделенный текст в буфер, а затем удаляет его из поля ввода. |

maskedit **TMaskEdit** (отдаленный потомок TWinControl)

Компонент предназначен для ввода и редактирования текста в соответствии с шаблоном.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TMaskEdit |
| Название | Тип | Описание |
| EditMask | String | Шаблон. |

Шаблон состоит из трех частей, разделяемых точкой с запятой. Первая часть представляет собой маску ввода, каждый символ которой определяет, что можно ввести на этом месте. Слово "должен" означает, что в данную позицию непременно нужно внести соответствующий символ, а слово "может" означает, что в данную позицию указанный символ либо вносится, либо нет.

|  |  |
| --- | --- |
|  | Символы маски |
| Символ | Описание |
| L | Должна быть буква. |
| I | Может быть буква. |
| A | Должна быть цифра или буква. |
| a | Может быть цифра или буква. |
| C | Должен быть любой символ ASCII. |
| c | Может быть любой символ ASCII. |
| 0 | Должна быть любая цифра. |
| 9 | Может быть любая цифра. |
| # | Может быть любая цифра или знаки "+", "–". |

Кроме перечисленных в таблице символов в маске могут быть форматирующие символы.

|  |  |
| --- | --- |
|  | Форматирующие символы маски |
| Символ | Описание |
| : | Разделитель для часов, минут и секунд при вводе времени. |
| / | Разделитель для дней, месяцев и дет при вводе даты. |
| ! | Подавление пробелов перед текстом. |
| > | Переводит символы в верхний регистр. |
| < | Переводит символы в нижний регистр. |
| <> | Отменяет преобразование регистра. |
| \ | Символ, предшествующий литералу. После этого символа можно записать любой символ, и он будет включен в вводимый текст. |

Вторая часть шаблона определяет, следует ли сохранять разделительные символы в указанном в маске виде. "1" означает, что символы сохраняются, "0" – не сохраняются.

Третья часть шаблона содержит символ, который будет находиться в пустых позициях ввода, обычно используют символ подчеркивания или пробел.

listbox1 **TListBox** (отдаленный потомок TWinControl)

Компонент представляет собой список.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TListBox |
| Название | Тип | Описание |
| Items | TStrings | Строки списка. |
| ItemIndex | Integer | Порядковый номер выделенной строки в списке. |
| MultiSelect | Boolean | Возможность выделения нескольких строк сразу. |
| SelCount | Integer | Количество выделенных строк в списке. |
| Selected[Index] | Boolean | Указывает, выделена или нет, строка с номером Index. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TListBox |
| Название | Тип | Описание |
| Clear | procedure | Удаляет все строки из списка. |

combobox **TComboBox** (отдаленный потомок TWinControl)

Компонент представляет собой раскрывающийся список.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TComboBox |
| Название | Тип | Описание |
| Items | TStrings | Строки списка. |
| ItemIndex | Integer | Номер выбранной строки. |
| Style | TComboBoxStyle | Стиль: csDropDown – раскрывающийся список и поле ввода, csSimple – поле ввода с обычным списком, csDropDownList, – раскрывающийся список без поля ввода. |

stringgrid **TStringGrid** (отдаленный потомок TCustomControl)

Компонент предназначен для представления данных в табличном виде. В каждую ячейку таблицы можно записать строку текста. В верхней и в левой части таблицы имеется фиксированная область, которая не участвует в прокрутке.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TStringGrid |
| Название | Тип | Описание |
| Col | LongInt | Индекс текущего столбца. |
| Row | LongInt | Индекс текущей строки. |
| ColCount | LongInt | Количество столбцов в таблице. |
| ColWidths[Index] | Integer | Ширина столбца с индексом Index. |
| EditorMode | Boolean | Возможность редактирования ячеек. |
| FixedColor | TColor | Цвет фиксированной области. |
| FixedCols | Integer | Количество зафиксированных столбцов. |
| FixedRows | Integer | Количество зафиксированных строк. |
| RowCount | LongInt | Количество строк в таблице. |
| RowHeight[Index] | LongInt | Высота строки с индексом Index. |
| Cells[ACol, ARow] | String | Содержимое ячейки с координатами (ACol, ARow). |
| Cols[Index] | TStrings | Столбец с индексом Index. |
| Rows[Index] | TStrings | Строка с индексом Index. |
| Objects[ACol, ARow] | TObject | Объект, связанный с ячейкой (ACol, ARow). |
| DefaultColWidth | Integer | Значение ширины столбца по умолчанию. |
| DefaultRowHeight | Integer | Значение высоты строки по умолчанию. |
| ScrollBars | TScrollStyle | Полосы прокрутки:  ssNone – отсутствуют, ssHorizontal – горизонтальная, ssVertical – вертикальная, ssBoth – обе. |

radiogroup **TRadioGroup** (отдаленный потомок TCustomControl)

Компонент представляет собой панель с расположенными на ней радиокнопками. Только одна из радиокнопок может быть выделена.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Свойства класса TRadioGroup |
| Название |  | Тип | Описание |
| Items |  | TStrings | Список названий радиокнопок. |
| ItemIndex |  | Integer | Порядковый номер выделенной радиокнопки. |

mainmenu **TMainMenu** (отдаленный потомок TComponent)

Компонент представляет собой меню, расположенное в верхней строке окна.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TMainMenu |
| Название | Тип | Описание |
| Items | TMenuItem | Элемент меню (см. ниже). |

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TMenuItem |
| Название | Тип | Описание |
| Caption | String | Название пункта меню.  Символ "&" перед каким-либо символом названия пункта меню выделяет этот символ подчеркиванием. При нажатии Alt-подчеркнутый символ вызывается данный пункт меню.  Для создания разделительной полосы между пунктами меню введите символ "-". |
| Checked | Boolean | Возможность появления галочки или точки (если входит в состав радиогруппы) слева от названия пункта меню. |
| Enabled | Boolean | Доступность пункта меню. |
| Items[Index] | TMenuItem | Подменю. |
| GroupIndex | Byte | Номер радиогруппы, к которой принадле­жит пункт меню. |
| RadioItem | Boolean | Принадлежность радиогруппе. |
| Visible | Boolean | Видимость пункта в меню. |
| ShortCut | Word | Соотносит пункту меню комбинацию клавиш. |

pagecontrol **TPageControl** (отдаленный потомок TWinControl)

Контейнер для объектов TTabSheet, представляющих страницы блокнота.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TPageControl |
| Название | Тип | Описание |
| Pages[Index] | TTabSheet | Страница блокнота (см. ниже). |
| PageCount | Integer | Количество страниц в блокноте. |
| ActivePage | TTabSheet | Активная страница. |
| TabPosition | TTabPosition | Расположение закладок: tpTop – сверху, tpBottom – снизу, tpLeft – слева, tpRight – справа. |
| MultiLine | Boolean | Возможность расположения закладок в несколько рядов. |
| TabIndex | Integer | Номер активной страницы. |

|  |  |  |
| --- | --- | --- |
| Свойства класса TTabSheet (прямой потомок TWinControl) | | |
| Название | Тип | Описание |
| PageIndex | Integer | Расположение страницы в блокноте. |
| TabVisible | Boolean | Видимость страницы. |

panel **TPanel** (отдаленный потомок TCustomControl)

Компонент представляет собой панель, на которой можно расположить другие элементы управления.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TPanel |
| Название | Тип | Описание |
| Alignment | TAlignment | Выравнивание заголовка панели. |
| BevelInner | TPanelBevel | Стиль внутренней рамки панели: bvNone – рамка отсутствует, bvLowered – погружена, bvRaised – выступает. |
| BevelOuter | TPanelBevel | Стиль внешней рамки панели. |
| BevelWidth | Integer | Толщина внутренней и внешней рамок в пикселах. |
| BorderWidth | Integer | Расстояние между внутренней и внешней рамками в пикселах. |

timer **TTimer** (прямой потомок TComponent)

Компонент реализует функции системного таймера. Это невизуальный компонент. После того как в свойстве Enabled будет установлено значение true, компонент переходит в режим ожидания сообщений от системного таймера. Сообщения от системного таймера приходят через промежутки времени, равные значению, установленному в свойстве Interval. После получения сообщения вызывается событие OnTimer, после обработки которого компонент опять переходит в режим ожидания следующего сообщения от системы. Этот процесс продолжается, пока в свойстве Enabled не будет установлено значение false.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TTimer |
| Название | Тип | Описание |
| Enabled | Boolean | Позволяет получать сообщения от системного таймера. |
| Interval | Integer | Интервал времени, через который вызывается событие OnTimer. Значение 1000 равняется одной секунде. |

updown **TUpDown** (отдаленный потомок TWinControl)

Компонент предназначен для изменения числовых значений щелчком мыши по кнопкам со стрелками или нажатием клавиши управления курсором. Изменяемое числовое значение находится в свойстве Position. При нажатии на одну из кнопок со стрелками значение свойства Position уменьшается или увеличивается на величину, заданную в свойстве Increment в диапазоне, определяемом свойствами Max и Min.

При попытке увеличить значение, когда оно уже достигло максимума, увеличения не происходит либо устанавливается минимальное значение, в зависимости от значения свойства Wrap. Аналогичные действия выполняются и для минимального значения.

Компонент можно связать с другим элементом управления, например TEdit. Для этого необходимо установить соответствующее значение в свойстве Associate. После связывания компонент автоматически располагается на форме рядом со связанным элементом управления, слева или справа от него, в зависимости от значения свойства AlignButton, и принимает соответствующие размеры. Для элемента управления TEdit при нажатии на кнопки компонента происходит изменение значения в окне ввода.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TUpDown |
| Название | Тип | Описание |
| Associate | TWinControl | Ссылка на связанный с компонентом элемент управления. |
| AlignButton | TUDAlignButton | Расположение компоненнта отно­си­тель­но связанного с ним элемента управления: udLeft – слева, udRight – справа. |
| Orientation | TUDOrientation | Ориентация стрелок: udHorizontal – горизонтальная, стрелки: влево и вправо, udVertical – вертикальная, стрелки: вверх и вниз. |
| Increment | Integer | Приращение за одно нажатие на одну из кнопок. |
| Max | SmallInt | Максимальное значение диапазона изменения. |
| Min | SmallInt | Минимальное значение диапазона изменения. |
| Position | SmallInt | Текущее значение. |
| Wrap | Boolean | Определяет поведение компонента при достижении границ диапазона из­ме­не­ния: true – циклический переход к минимальному и максимальному значениям. |

progress **TProgressBar** (прямой потомок TWinControl)

Компонент предназначен для отображения состояния процесса выполнения каких-либо действий. Изменение положения происходит после вызова одного из методов StepIt или StepBy.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TProgressBar |
| Название | Тип | Описание |
| Step | Integer | Величина приращения значения. |
| Max | Integer | Максимальное значение диапазона изменения. |
| Min | Integer | Минимальное значение диапазона изменения. |
| Position | Integer | Текущее значение. |

|  |  |  |
| --- | --- | --- |
|  |  | Методы класса TProgressBar |
| Название | Тип | Описание |
| StepIt | procedure | Изменяет положение на величину приращения, определенную в свой­стве Step. |
| StepBy(Delta: Integer) | procedure | Изменяет положение на величину, заданную параметром Delta. |

bar **TTrackBar** (прямой потомок TWinControl)

Компонент представляет собой бегунок. Пользователь может перемещать бегунок по шкале с помощью мыши или клавиш управления курсором.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TTrackBar |
| Название | Тип | Описание |
| Frequency | Integer | Расстояние между делениями шкалы. |
| LineSise | Integer | Величина приращения при нажатии клавиши управления курсором. |
| PageSize | Integer | Величина приращения при нажатии клавиш PgUp, PgDown. |
| Orientation | TTrackBar Orientation | Положение: trHorizontal – горизонтальное, trVertical – вертикальное. |
| Max | Integer | Максимальное значение диапазона из­ме­не­ния. |
| Min | Integer | Минимальное значение диапазона изменения. |
| Position | Integer | Текущая позиция бегунка. |

bevel **TBevel** (отдаленный потомок TControl)

Компонент представляет собой рамку.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TBevel |
| Название | Тип | Описание |
| Shape | TBevelShape | Форма рамки: bsBox – панель, bsFrame – рамка, bsTopLine, bsBottomLine, bsLeftLine, bsRightLine – линия. |
| Style | TBevelStyle | Стиль рамки: bsLowered – погруженная, bsRaised – приподнятая. |

checkbox **TCheckBox** (отдаленный потомок TWinControl)

Компонент реализует элемент управления – флажок.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TCheckBox |
| Название | Тип | Описание |
| Checked | Boolean | Определяет, установлен флажок или сброшен. |
| AllowedGrey | Boolean | Определяет, может ли флажок находиться в третьем состоянии – закрашенным в серый цвет. |
| State | TCheckBox State | Состояние флажка: cbUnchecked – сброшен, cbChecked – установлен, cbGrayed – закрашен в серый цвет. |

image **TImage** (отдаленный потомок TControl)

Компонент представляет собой картинку.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TImage |
| Название | Тип | Описание |
| Autosize | Boolean | Определяет, будут ли размеры компонента изменяться в соответствии с размерами рисунка. |
| Picture | TPicture | Рисунок. |
| Stretch | Boolean | Определяет, будут ли размеры рисунка изменяться в соответствии с размерами компонента. |
| Transparent | Boolean | Определяет, будет ли фон рисунка прозрачным. |

**Диалоги**

Выполнение некоторых операций, например, таких, как открытие и сохранение документов, обеспечивается стандартными окнами, функционирующими в модальном режиме и не позволяющими менять свои размеры. Такие окна называются диалогами.

opendialog **TOpenDialog**, savedialog **TSaveDialog**

(отдаленные потомки TComponent)

Диалоги открытия и сохранения файлов.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства классов TOpenDialog и TSaveDialog |
| Название | Тип | Описание |
| DefaultExt | String | Расширение по умолчанию, добавляется к выб­ран­но­му файлу. |
| FileName | String | Имя последнего файла из списка выбранных в диалоге. |
| Files | TString | Список файлов, выбранных в диалоге. |
| Filter | String | Маска отбора файлов в диалоге. |
| FilterIndex | Integer | Фильтр отбора файлов по умолчанию |
| InitialDir | String | Каталог, к которому диалог обращается при открытии. |
| Title | String | Заголовок диалога. |

Маска фильтра состоит из двух частей, разделенных вертикальной чертой. Первая часть содержит название фильтра, а вторая – маску файла. Например,

'Модули проекта | \*.pas'.

Несколько фильтров разделяются вертикальной чертой. В одном фильтре можно отбирать файлы с различными расширениями. Пример более сложного фильтра:

'Графические файлы | \*.bmp; \*.jpg; \*.gif; \*.pcx | PCX–файлы | \*.pcx'.

colord **TColorDialog** (отдаленный потомок TComponent)

Диалог выбора цвета.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TColorDialog |
| Название | Тип | Описание |
| Color | TColor | Определяет выбранный цвет. |

fontd **TFontDialog** (отдаленный потомок TComponent)

Диалог установки параметра шрифта.

|  |  |  |
| --- | --- | --- |
|  |  | Свойства класса TFontDialog |
| Название | Тип | Описание |
| Font | TFont | Содержит параметры шрифта. |

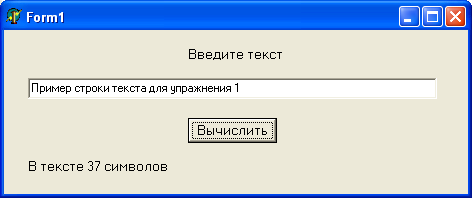
Компоненты диалогов не являются визуальными. Для того чтобы вызвать диалоговое окно, необходимо вызвать метод Execute, после вызова которого появляется диалоговая форма. Если пользователь выбрал один или несколько файлов, или выбрал цвет, или выбрал параметры шрифтов и затем нажал кнопку "OK", то функция возвращает значение true. Если пользователь нажал кнопку "Cancel", то функция возвращает значение false.

|  |  |  |
| --- | --- | --- |
|  |  | Методы классов диалогов |
| Название | Тип | Описание |
| Execute: Boolean | function | Вызывает окно диалога. |

**Пример выполнения**

**Задание №1**

**Создайте приложение, предлагающее пользователю ввести строку, определяющее количество символов в ней и выводящее результат, согласуя окончание слова «символов» с числом символов, например, «В тексте 1 символ», «В тексте 32 символа», «В тексте 47 символов».**



Видокна приложения, подсчитывающего число символов в тексте

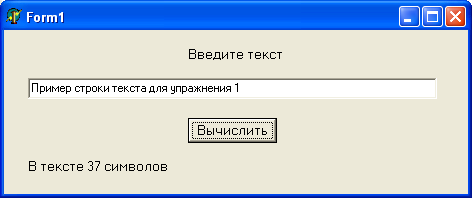
**Алгоритм выполнения работы**

1. Создайте форму и разместите на ней компоненты: Edit1, Label1, Label2, Button1, как показано на рис. 41.

2. Задайте значения свойств Label1.Caption — «Введите текст», Label2.Caption — «В тексте», Button1.Caption — «Вычислить».

3. Удалите текст Edit1 из соответствующего компонента. Выровняйте компоненты на форме.

4. Для вычисления количества символов во введенной строке и вывода результатов создайте процедуру обработчика события щелчка мышью на кнопке Button1. Выберите в окне Инспектора объектов объект Button1, затем на странице Событий произведите двойной щелчок мышью на пустом поле списка в событии OnClick. После этого в окне Редактора кода будет сгенерирована заготовка процедуры обработчика события procedure TForm1.Button1Click (Sender: TObject);.



**Рис.41.** Видокна приложения, подсчитывающего число символов в тексте

Введите в текст процедуры следующее описание переменных:

**var**

**N: integer;** {последняя цифра}

**S: string;** {изменяемое слово «символ»}

Введите в окне Редактора кода текст тела процедуры обработки текста:

**begin**

**N:=Length(Edit1.Text);** {определить число символов}

**Label2.Caption: = 'В тексте ';**

**if N>20 then N:=N mod 10;** {выделить последнюю цифру}

**case N of** {в зависимости от последней цифры}

**1 : S:=' символ';** {изменять окончание слова}

**2..4 : S:=' символа';**

**0,5..20 : S:=' символов';**

**end;**

**Label2.Caption:=Label2.Caption+**

**IntToStr(Length(Edit1.Text))+ S**; {вывести результат}

**end;**

5. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.

6. Проверьте работу приложения, задавая текст с разным количеством символов и проверяя правильность вывода результатов работы.

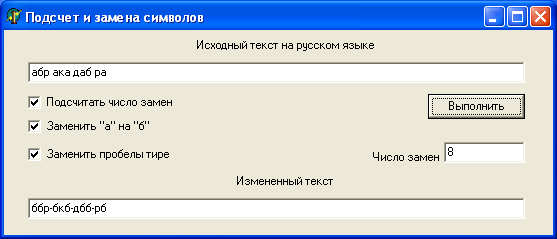
**Дополнительное задание 1.**

Измените цветовое решение формы.

**Пример выполнения**

**Задание №2**

**Создайте приложение, предлагающее пользователю ввести строку текста, а затем заменяющее символы в тексте и подсчитывающее количество замененных символов.**

****

**Алгоритм выполнения работы.**

Вариант замены символов должен определяться по положению соответствующего флажка CheckBox, как показано на рис. 5.1.

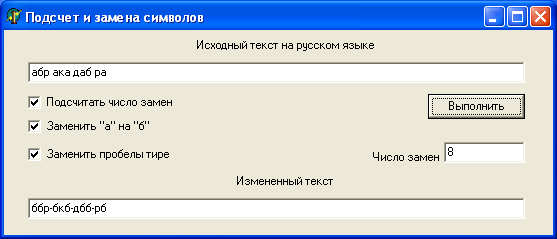
1. Создайте форму, для свойства Caption задайте значение «Подсчет и замена символов».

2. На форме разместите компоненты: Edit1, Edit2, Edit3, CheckBox1, CheckBox2, СпескВохЗ, Label1, Label2, Label3, Button1.

3. Задайте значения свойств Label1.Caption — «Исходный текст на русском языке», Label2.Caption — «Измененный текст», Label3.Caption — «Число замен», CheckBox1.Caption — «Подсчитать число замен», CheckBox2.Caption — «Заменить "а" на "б"», CheckВохЗ. Caption — «Заменить пробелы на тире», Button1.Caption — «Выполнить».

4. Удалите текст Edit1, Edit2, Edit3 из соответствующих компонентов.

5. Выровняйте компоненты на форме, как показано на рис. 5.1.

****

**Рис. 5.1.** Вид окна приложения, выполняющего замену символов в тексте

6. Замену символов в тексте и подсчет количества замененных символов опишите в процедуре обработчика события щелчка мышью на кнопке Button1. Для создания процедуры обработчика события выберите в окне Инспектора объектов объект Button1, затем на странице События произведите двойной щелчок на пустом поле списка в событии OnClick. После этого в окне Редактора кода в заготовку процедуры обработчика события procedure TForm1.Button1Click(Sender: TObject); введите следующее описание переменных:

**var**

**S : string;** {строка текста}

**N : byte;** {количество замен символов}

Если флажок CheckBox1 установлен, то свойство CheckBox1.Checked приобретает значение True.

С учетом вышесказанного тело процедуры может быть записано следующим образом:

**N:=0;** {обнулить число букв а}

**Edit3.Text:=' ';**

**Edit2.Text:=' ';**

**S:=Edit1.Text;** {присвоить S значение текстовой строки}

**if CheckBox2.Checked then** {если включен флажок CheckBox2}

**while Pos('a', S) > 0 do** {если в строке найдена буква а}

**begin**

**N:=N+1;** {увеличить счетчик замен на 1}

**S[Pos('a', S)]:='б';** {заменить букву а буквой б}

**end;**

**if CheckBox3.Checked then**

**while Pos(' ', S) > 0 do** {если в строке найден пробел}

**begin**

**N:=N+1;** {увеличить счетчик замен на 1}

**S[Pos(' ', S)]:='-';** {заменить пробел символом -}

**end;**

**if CheckBox1.Checked then Edit3.Text:=IntToStr(N);** {вывод числа замен символов}

**Edit2.Text:=S;** {вывод измененного текста}

**end;**

7. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.

8. Проверьте работу приложения, задавая различный текст на русском языке с разным количеством букв «а» и пробелов.

9. Проверьте правильность вывода результатов работы и закройте окно приложения.

**Пример выполнения**

**Задание №3**

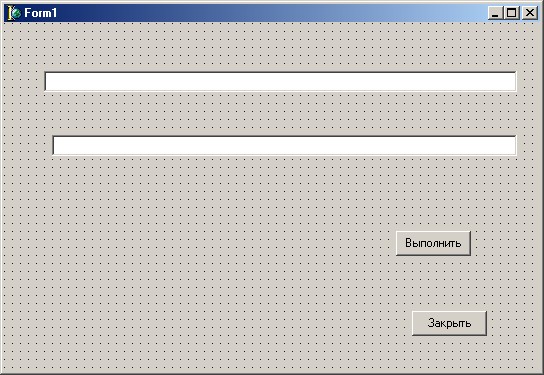
**Дан текст: Наступила зима (температура -5 градусов). Выпал снег (10см). Исключить из текста группы символов, расположенных между скобками ” (” и ” )”. Сами скобки должны быть исключены. Предполагается, что внутри каждой пары скобок нет других скобок.**

**1. Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

Button1 – кнопка, имеющая заголовок **Выполнить,** при нажатии которой первоначальный текст из программы заносится в компонент Edit1, а преобразованный текст заносится в компонент Edit2 . Button2 – кнопка, имеющая заголовок **Закрыть,** при нажатии которой форма закрывается.

Edit1 – компонент предназначенный для отображения первоначального текста до изменения его. Edit2 – компонент предназначенный для отображения текста после изменения его.



**Программа**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type

TForm1 = class(TForm)

Edit1: TEdit;

Button1: TButton;

Button2: TButton; Edit2: TEdit;

procedure Button1Click(Sender: TObject); procedure Button2Click(Sender: TObject); private

{ Private declarations } public

{ Public declarations } end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

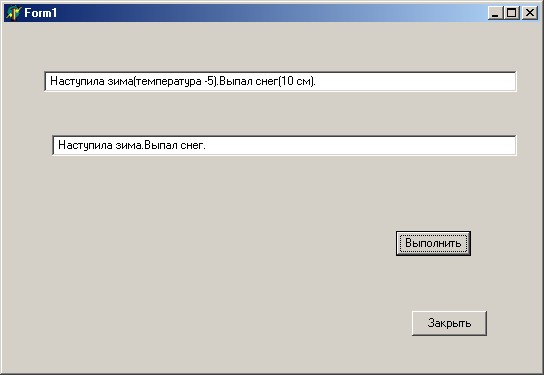
procedure TForm1.Button1Click(Sender: TObject); Var s:string; n,nn,n1,n2,i:integer; begin

s:=' Наступила зима(температура -5).Выпал снег(10 см).'; edit1.Text:=s; n:=Length(s); for i:=1 to n do begin n1:=pos('(',s); n2:=pos(')',s) ; nn:=n2-n1+1;

if (i>=n1) and(i<=n2) then delete(s,n1,nn); end; edit2.Text:=s; end;

procedure TForm1.Button2Click(Sender: TObject); begin

Application.Terminate; end; end.



**Пример выполнения**

**Задание №4**

**Пример программы обработки записей**

**1.Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

TLabel – используется для создания комментариев. Для этого использовалось свойство компонента Caption, которое имеет строковый тип.

TEdit – данный компонент был использован для ввода исходных данных о студентах.

TButton – кнопка, при нажатии которой производятся действия по запоминанию данных о студентах. Код прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Комментарий на кнопке прописывается в свойстве кнопки Caption.

TGroupBox – компонент, предназначенный для группировки внутри себя других компонентов. Заголовок компонента прописывается в его свойстве Caption.

TMemo – компонент, позволяющий отображать текстовую информацию. Для внесения дополнительной информации используется метод Append() свойства Lines.

**2.Условие:**

Ввести информацию о 10 студентах группы. Информация содержит фамилию, имя, отчество студента, год рождения, четыре оценки за экзамены последней сессии. Распечатать анкетные данные студентов, получивших в последнюю сессию оценку 2.

**3.Форма:**



TButton

TMemo

TEdit

TLabel

TGroupBox

**4.Программа:**

unit Unit1; interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls; type

TForm1 = class(TForm)

GroupBox1: TGroupBox;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Edit4: TEdit;

Edit5: TEdit;

Button1: TButton;

Button2: TButton;

Edit6: TEdit;

Edit7: TEdit;

Edit8: TEdit; Memo1: TMemo;

procedure Button1Click(Sender: TObject); procedure Button2Click(Sender: TObject);

private

{ Private declarations } public

{ Public declarations } end;

Student=record

Surname:string;

Name:string;

Otch: string;

YearBirth:integer; Mark:array[1..4]of byte;

end; var

. Form1: TForm1;

Stud:array [1..10] of Student; n:byte=0; implementation {$R \*.dfm}

procedure TForm1.Button1Click(Sender: TObject); begin n:=n+1; with Stud[n] do begin

Surname:=Edit1.Text;

Name:=Edit2.Text;

Otch:=Edit3.Text; YearBirth:=StrToInt(Edit4.Text); mark[1]:=StrToInt(Edit5.Text); mark[2]:=StrToInt(Edit6.Text); mark[3]:=StrToInt(Edit7.Text); mark[4]:=StrToInt(Edit8.Text); end;

Memo1.Lines.Append(Edit1.Text+' '+Edit2.Text+' '+Edit3.Text+' '+Edit4.Text);

Memo1.Lines.Append('Оценки: '+Edit5.Text+' '+Edit6.Text+' '+Edit7.Text+'

'+Edit8.Text);

Edit1.Clear;

Edit2.Clear;

Edit3.Clear;

Edit4.Clear;

Edit5.Clear;

Edit6.Clear;

Edit7.Clear; Edit8.Clear;

if n=10 then GroupBox1.Enabled:=false; end;

procedure TForm1.Button2Click(Sender: TObject); var i,j:integer; begin Memo1.Clear;

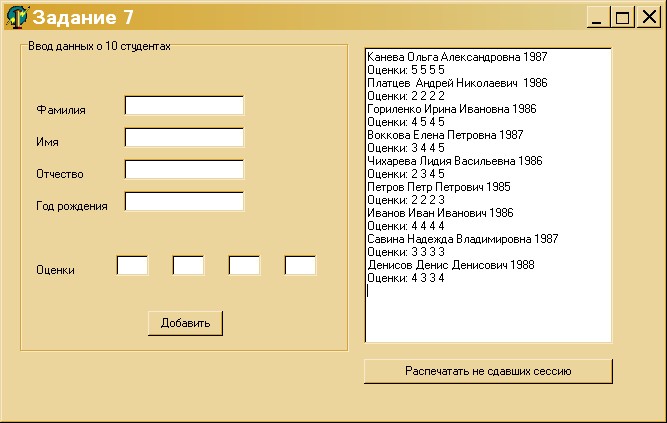
for i:=1 to n do with stud[i] do for j:=1 to 4 do begin

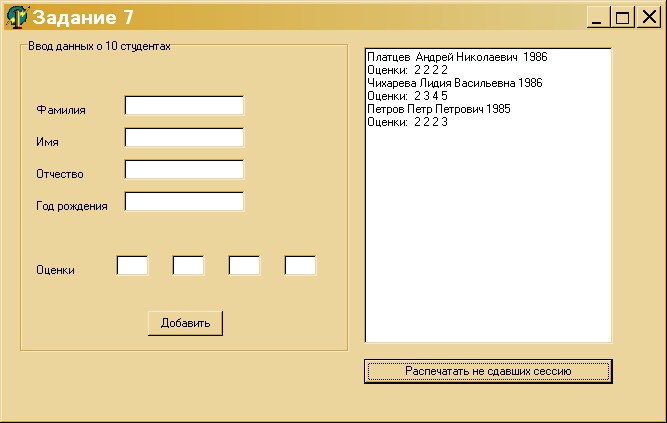
if Mark[j]=2 then begin

Memo1.Lines.Append(Surname+' '+Name+' '+Otch+' '+IntToStr(YearBirth));

Memo1.Lines.Append('Оценки: '+' '+IntToStr(Mark[1])+' '+IntToStr(Mark[2])+' '+IntToStr(Mark[3])+' '+IntToStr(Mark[4])); break; end; end; end; end.

**5.Результат:**





**Пример выполнения**

**Задание №5**

**Пример программы обработки файлов**:

**1.Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

TLabel – используется для создания комментариев. Для этого использовалось свойство компонента Caption, которое имеет строковый тип.

TEdit – компонент, используемый для ввода критериев поиска.

TButton – кнопка, при нажатии которой производятся вычисления. Код для вывода списка по запросу прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Комментарий на кнопке прописывается в свойстве кнопки Caption.

TStringGrid – данный компонент был использован для вывода информации о пациентах.

TGroupBox – компонент, предназначенный для группировки внутри себя других компонентов. Заголовок компонента прописывается в его свойстве Caption.

**2.Условие:**

Создать файл, содержащий сведения о пациентах глазной клиники. Информация такова: фамилия пациента, пол, возраст, место проживания (город), диагноз.

Написать программу, выдающую следующую информацию: - количество иногородних, прибывших в клинику;

- список пациентов старше X лет с диагнозом Y.

**:**

**.Форма**

**3**

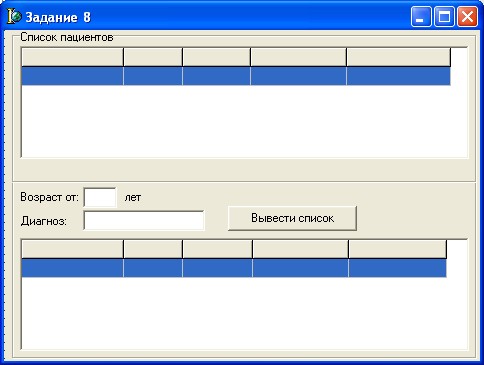
Лабораторная работа № 9.

***Графика в***

***DELPHI***

**4.**

**Программа**



TStringGrid

TEdit

TGroupBox

TButton

unit Unit1; interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, Grids, StdCtrls; type

TForm1 = class(TForm)

GroupBox1: TGroupBox;

StringGrid1: TStringGrid;

GroupBox3: TGroupBox;

Edit2: TEdit;

Label2: TLabel;

Edit1: TEdit;

Label3: TLabel;

Label1: TLabel;

Button1: TButton;

StringGrid2: TStringGrid; Label4: TLabel; procedure Button1Click(Sender: TObject); procedure FormCreate(Sender: TObject);

private

{ Private declarations } public

{ Public declarations } end; var

Form1: TForm1; implementation {$R \*.dfm}

procedure TForm1.FormCreate(Sender: TObject); var f: TextFile;

Fm,S,A,C,D:string;

n:integer; begin

StringGrid1.Cells[0,0]:='Фамилия';

StringGrid1.Cells[1,0]:='Пол';

StringGrid1.Cells[2,0]:='Возраст';

StringGrid1.Cells[3,0]:='Город';

StringGrid1.Cells[4,0]:='Диагноз';

StringGrid2.Cells[0,0]:='Фамилия';

StringGrid2.Cells[1,0]:='Пол';

StringGrid2.Cells[2,0]:='Возраст';

StringGrid2.Cells[3,0]:='Город'; StringGrid2.Cells[4,0]:='Диагноз';

AssignFile(f,'Pacient.txt');

Reset(F); n:=0; while not(Eof(f)) do begin Readln(f,Fm);

Readln(f,S);

Readln(f,A);

Readln(f,C); Readln(f,D); if C<>'Ухта' then n:=n+1; with StringGrid1 do

begin

Cells[0,RowCount-1]:=Fm;

Cells[1,RowCount-1]:=S;

Cells[2,RowCount-1]:=A; Cells[3,RowCount-1]:=C;

Cells[4,RowCount-1]:=D; RowCount:=RowCount+1;

end; end; CloseFile(f);

StringGrid1.RowCount:=StringGrid1.RowCount-1;

Label4.Caption:='Количество иногородних пациентов: '+IntToStr(n); end;

procedure TForm1.Button1Click(Sender: TObject); var f: TextFile;

Fm,S,A,C,D:string; begin

AssignFile(f,'Pacient.txt');

Reset(F);

StringGrid2.RowCount:=2; while not(Eof(f)) do begin Readln(f,Fm);

Readln(f,S);

Readln(f,A);

Readln(f,C); Readln(f,D);

if (StrToInt(A)>=StrToInt(Edit1.Text)) and (D=Edit2.Text) then with StringGrid2 do

begin

Cells[0,RowCount-1]:=Fm;

Cells[1,RowCount-1]:=S;

Cells[2,RowCount-1]:=A; Cells[3,RowCount-1]:=C;

Cells[4,RowCount-1]:=D;

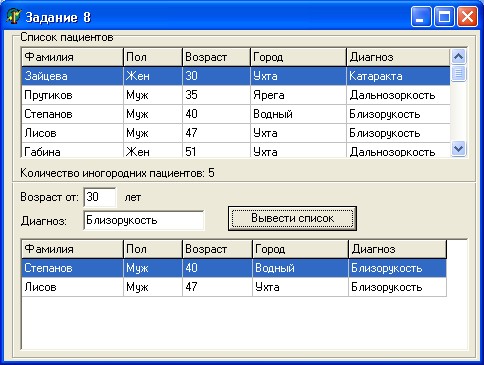
RowCount:=RowCount+1;

end; end;

CloseFile(f);

StringGrid2.RowCount:=StringGrid2.RowCount-1; end; end.

**5.Результат:**

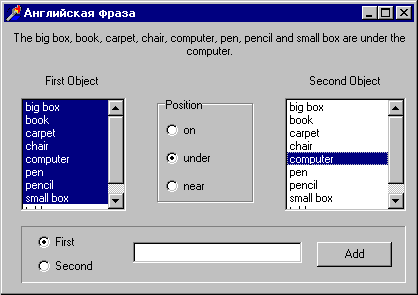


**Задание 1. Генератор английских фраз**

**Задача:** Создать приложение для генерации английской фразы.

После запуска программы появляется предложение, состоящее из слов, стоящих первыми в каждом списке (слова должны быть выделены). После выбора в первом списке одного слова или группы из нескольких необязательно подряд идущих слов предложение должно мгновенно измениться. Аналогично, после выбора одного слова из радиогруппы или одного слова из второго списка, предложение должно мгновенно измениться. При этом должны соблюдаться правила построения английских предложений: глагол *is* – в случае единственного подлежащего, глагол *are* – во множественном числе, а также расстановка запятых при перечислении дополнений и союза *and* перед последним словом в перечислении. Если предложение не помещается на одной строке, оно должно продолжаться на следующей.

Пользователь должен иметь возможность добавить новое слово в первый или во второй список. При нажатии на кнопку "Add" слово, введенное в окне ввода, добавляется в тот список, который выбран в радиогруппе слева от окна ввода. При этом должна проводиться проверка на наличие введенного слова в списке, и если это слово уже содержится в списке, то оно не добавляется.



Компоненты:

Label – сгенерированная фраза, названия списков;

RadioGroup – первая группа содержит список предлогов, вторая – номера списков;

ListBox – содержит список подлежащих и список дополнений, в первом списке разрешен множественный выбор;

Edit – окно ввода нового слова в один из списков в зависимости от выбора соответствующей радиокнопки слева от окна ввода;

RadioButton – радиокнопка выбора списка: First – список подлежащих, Second – список дополнений;

Button – кнопка добавления слова в один из списков;

Bevel – рамка, объединяющая поле ввода, две радиокнопки и кнопку "Add".

**Задание 2. Множества**

**Задача:** Создать приложение, выполняющее логические операции над множествами. Пользователь щелчком мыши включает или исключает элемент одного из множеств SetA или SetB, результат одной из операций над множествами (объединение, пересечение, разность) мгновенно отражается в множестве SetC. Одновременно обновляется информация об отношениях между множествами SetA и SetB: A=B – множества совпадают, A<=B – первое множество содержится во втором, A>=B - второе множество содержится в первом. Выбор операции над множествами осуществляется из раскрывающегося списка.

Компоненты:

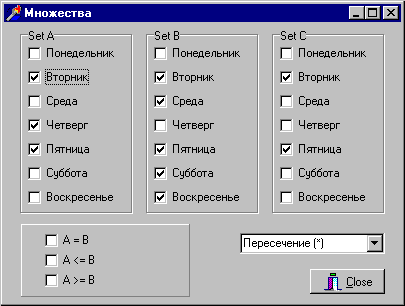
GroupBox – содержит элементы каждого из трех множеств;

CheckBox – элемент множества (каждый компонент принадлежит одному из компонентов GroupBox), а также индикатор отношения между множествами (все индикаторы принадлежат панели Panel);

Panel – содержит три индикатора отношений между множествами;

ComboBox – раскрывающийся список для выбора операции над множествами, содержит три операции: объединение, пересечение, разность;

BitBtn – кнопка завершения работы программы (Close).

****

Замечание: Приложение должно содержать единственную процедуру – отклик на событие "щелчок мыши", которая должна быть назначена всем компонентам CheckBox из первого и второго множеств. В этой процедуре должна обновляться информация о составе третьего множества и информация об отношениях между первым и вторым множествами. Для ссылки на элементы множеств следует воспользоваться свойством Controls компонентов GroupBox, которым принадлежат компоненты CheckBox. Изменение состава третьего множества и состояния индикаторов отношения между множествами должно осуществляться только программно, щелчок мыши по этим элементам не должен иметь обычного эффекта – выделение или снятие выделения.

**Задание 3. Два списка**

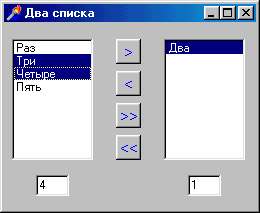
**Задача:** Создать приложение, выполняющее функцию перемещения строк между двумя списками. Два списка содержат несколько различных строк. Оба списка допускают множественный выбор. При нажатии на кнопку ">" выделенная строка или несколько выделенных строк переносятся из левого списка в правый. Это означает, что они удаляются из левого списка и добавляются в правый список. Аналогично работает кнопка "<", при нажатии на которую выделенные строки из правого списка переносятся в левый. При нажатии на кнопку ">>" все строки левого списка переносятся в правый список. При нажатии на кнопку "<<" все строки правого списка переносятся в левый список. В двух окошках под списками отражается текущее количество записей в каждом списке. При переносах эти значения должны мгновенно обновляться.

Компоненты:

ListBox – окна со списками;

Button – кнопки ">", "<", ">>", "<<";

Edit – окна с количеством элементов в каждом списке.



**Задание 4. Блокнот**

**Задача:** Создать приложение, выполняющее функции записной книжки с вкладками для каждого месяца одного года. Окно приложения содержит блокнот, на каждой странице которого находится список заметок для одного месяца. Окно ввода предназначено для отображения текущей выделенной строки в списке и для ввода новой строки. Меню имеет два пункта: "Файл" – включает пункты "Загрузить", "Сохранить" и "Выход" – для сохранения и загрузки всех записей блокнота в файле, и "Редактирование" – включает пункты "Добавить", "Изменить", "Удалить" – для работы с записями на одной странице блокнота.

Компоненты:

Label – название месяца на каждой странице;

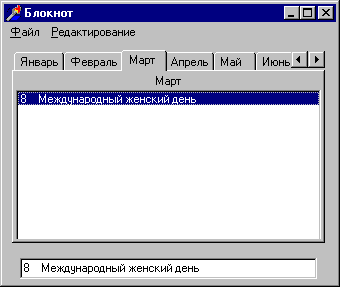
ListBox – список заметок на каждой странице;

PageControl – страницы блокнота;

MainMenu – основное меню программы;

Edit – окно отображения выделенной записи в блокноте, а также используется для ввода новой записи;

OpenDialog, SaveDialog – сохранение и загрузка файла с листами блокнота.

****

**Лабораторная работа №2**

**Визуальное программирование в среде Delphi**

**при обработке текстовой документации**

**по дисциплине «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»**

Составитель: Садчиков П.Н.

Разработка и внедрение в учебный процесс новой лабораторной работы с использованием нового приобретенного оборудования и сопроводительных методических рекомендаций для студентов направления подготовки 09.03.02 «информационные системы и технологии» направленности (профиля) «Информационные системы и технологии в строительстве и архитектуре» очной и заочной форм обучения

Печатается в авторской редакции

Подписано к печати \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Уч.-изд. л. \_\_\_\_. Тираж \_\_\_\_\_\_ экз. Заказ №\_\_\_\_\_\_

Отпечатано в Астраханском государственном архитектурно-строительном университете

Государственное автономное Образовательное

учреждение Астраханской области

высшего образования

«АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ АРХИТЕКТУРНО-СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра систем автоматизированного

проектирования и моделирования

**Лабораторная работа №3**

***Визуальное программирование при обработке массивов данных***

**в среде Delphi**

**по дисциплине «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»**

Разработка и внедрение в учебный процесс новой лабораторной работы с использованием нового приобретенного оборудования

и сопроводительных методических рекомендаций

для студентов направления подготовки

09.03.02 «информационные системы и технологии»

направленности (профиля)

«Информационные системы и технологии в строительстве и архитектуре»

очной и заочной форм обучения

Астрахань 2020

Составитель: к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Садчиков П.Н./

(занимаемая должность, (подпись) И. О. Ф.

учёная степень и учёное звание)

Рецензент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_ /

(занимаемая должность, (подпись) И. О. Ф.

учёная степень и учёное звание)

Инструкция по выполнению лабораторной работы по дисциплине «Технологии программирования» рассмотрена и одобрена на заседании кафедры «Системы автоматизированного проектирования и моделирования» ГАОУ АО ВО «АГАСУ»

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_ 2020 г.

Зав. кафедрой  ***\_\_\_\_\_\_\_\_\_\_\_\_\_*** ***/***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***/***

**(**подпись) (инициалы, фамилия)

Согласовано с УМО ГАОУ АО ВО «АГАСУ»

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Специалист УМУ \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (инициалы, фамилия)

Инструкция по выполнению лабораторной работы по дисциплине «Технологии программирования» утверждена и рекомендована к публикации на заседании методического совета направления подготовки 09.03.02 «информационные системы и технологии» ГАОУ АО ВО «АГАСУ»

Протокол № \_\_\_\_ от \_\_\_\_\_\_\_\_\_\_ 2020 г.

Председатель МКН «Информационные системы и технологии»

направленность (профиль)

«Информационные системы и технологии в строительстве и архитектуре»

\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (инициалы, фамилия)

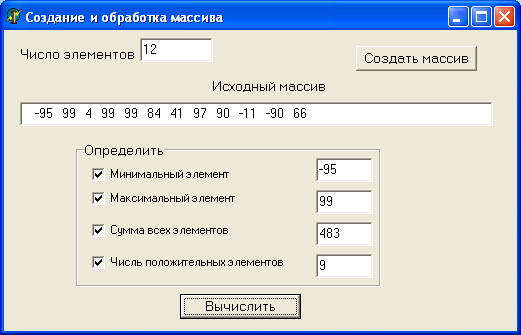
©Садчиков П.Н.

©ГАОУ АО ВО «Астраханский государственный архитектурно-строительный университет»

**Пример выполнения**

**Задание №1**

**Создайте приложение, которое предлагает пользователю задать размер линейного размера, заполняет этот массив случайными целыми числами, выводит список элементов массива, а затем по выбору пользователя определяет минимальный и максимальный элементы массива, сумму всех элементов и количество положительных элементов.**



Окно приложения создания и обработки линейного массива

**Алгоритм выполнения работы.**

1. Создайте форму, для свойства Caption задайте значение «Создание и обработка массива».

2. На форме разместите компоненты Edit1 (число элементов массива) и Edit2 (исходный массив), кнопку Button1 (Создать массив), для свойства Caption кнопки задайте значение «Создать массив».

3. Разместите на форме панель GroupBox1, для свойства Caption которой задайте значение «Определить».

ПРИМЕЧАНИЕ

Панель GroupBox — это контейнер с рамкой и надписью, объединяющий группу свя­занных органов управления, таких как переключатели RadioButton, флажки CheckBox и т. д. GroupBox имеет встроенную рамку с надписью, которая обычно используется для выделения на форме группы функционально объединенных компонентов.

4. В панели GroupBox1 разместите компоненты CheckBox1, CheckBox2, CheckBox3 и CheckBox4, для свойств Caption которых задайте значения: «Минимальный элемент», «Максимальный элемент», «Сумма всех элементов», «Число положительных элементов», соответственно.

5. Напротив них разместите компоненты Edit3, Edit4, Edit5 и Edit6. Если компоненты CheckBox1, CheckBox2, СпескВохЗ, CheckBox4 окажутся размещенными под панелью GroupBox1 и не будут отображаться, то следует выделить панель GroupBox1 и выбрать в контекстном меню команду Control ► Send to Back (Порядок ► На задний план).

6. В нижней части формы разместите кнопку Button2 и задайте значение свойства Button2.Caption — «Вычислить».

7. Удалите текст Edit1, Edit2, Edit3, Edit4, Edit5, Edit6 из соответствующих компонентов.

8. Выровняйте компоненты на форме, как показано на рис.6.1. Зафиксируйте положение компонентов на форме, выбрав в меню Delphi команду Вид ► Зафиксировать.

9. Сохраните файл проекта и программного модуля.

10. Прежде чем создавать обработчики событий щелчка мышью по кнопкам Button1 и Button2, в разделе описания переменных опишите переменные целого типа N и I, где N — размер массива, а I — порядковый номер элемента массива, а также М - динамический массив целых чисел.

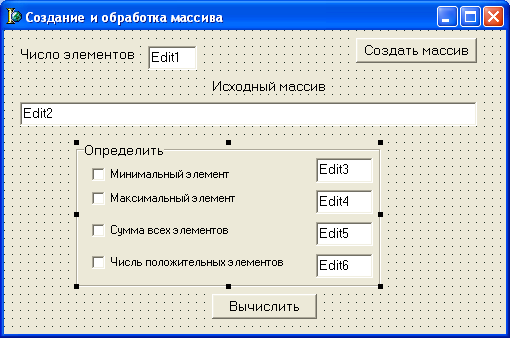
Создание и обработка линейного массива

**var**

**Form1: TForm1;**

**N,I: integer;**

**М: array of integer;** {описание динамического массива целых чисел}



**Рис. 6.1.** Окно формы с компонентами

11. Для предупреждения ввода в окно Edit1 нечислового значения реализуем обработку события нажатия на клавишу в окне Edit1, чтобы запретить ввод любых символов, кроме цифр от 0 до 9.

Для создания процедуры обработчика события нажатия на клавишу в окне Edit1 выберите в окне Инспектора объ­ектов компонент Edit1 и на странице События дважды щелкните левой кнопкой мыши на пустом поле списка в событии On Key Press. После этого окно Редактора кода немедленно получит фокус и в разделе interface появится запись процедуры обработчика события:

**procedure Edit1KeyPress(Sender: TObject; var Key:Char);**

а в разделе implementation — текст заготовки этой процедуры:

**procedure Edit1KeyPress(Sender: TObject; var Key: Char);**

**begin**

**end;**

Вставьте в тело процедуры следующий оператор:

**if not (Key in ['0'..'9']) then Key:=#0;**

Действие этого оператора сводится к сравнению значения переменной Key с множеством значений ['0'..'9']. Если символ нажатой клавиши не входит в это множество, то Key присваивается значение нулевого символа (#0).

12. Таким образом, в окне Edit2 будет отображаться текст, состоящий только из цифр.

13. Создание массива целых чисел опишите в процедуре обработчика события щелчка мышью на кнопке Button1. Для создания процедуры обработчика события выберите в окне Инспектора объектов объект Button1, затем на странице События сделайте двойной щелчок на пустом поле списка в событии OnClick. После этого отредактируйте заготовку процедуры обработчика события procedure TForm1.Button1Click(Sender: TObject); в окне Редактора кода следующим образом:

**procedure TForm1.Button1Click(Sender: TObject)**;

**begin**

**Randomize;**

**N:=StrToInt(Edit1.Text);** {число элементов массива}

**SetLength(M, N);** {задать массиву М длину N}

**Edit2.Text:=' ';** {очистить окно Edit2}

**for I:= 0 to N-1 do** {заполнить массив случайными значениями целых чисел}

**begin**

**M[I]:= Round(Sin(Random(100))\*100);**

{присвоить элементу массива случайное значение}

**Edit2.Text:=Edit2.Text+' '+IntToStr(M[I]);**

{вывести элементы массива}

**end;**

**end;**

**ПРИМЕЧАНИЕ**

Оператор Randomize; задает случайное начальное значение для функции Random.

14. Обработку массива опишите в процедуре обработчика события щелчка мышью на кнопке Button2. Для этого выберите в окне Инспектора объектов объект Button2, затем на странице События сделайте двойной щелчок на пустом поле списка в событии OnClick. После этого в окне Редактора кода в заготовку процедуры обработчика события procedure TForm1.Button2Click(Sender: TObject); в раздел описания локальных переменных поместите следующее.описание:

**var**

**Max, Min, Sum, CountPlus:integer;** {результаты обработки массива}

где **Мах** — максимальный элемент массива,

**Min** — минимальный элемент массива,

**Sum** — сумма всех элементов массива,

**CountPlus** — количество положительных элементов массива.

Обработку массива можно реализовать с помощью цикла For, в котором вы­числение значения каждой из этих переменных записывается при помощи оператора if then, проверяющим условие CheckBox.Checked. Если свойство Checked имеет значение True, то выполняется вычисление соответствующей переменной. В заключительной части процедуры можно разместить вывод результатов обработки массива. Текст процедуры обработки массива может быть записан следующим образом:

**procedure TForm1.Button2Click(Sender: TObject);** {обработка массива} **var**

**Max, Min, Sum, CountPlus:integer;** {результаты обработки массива}

**begin**

**if CheckBox1.Checked then Min:=M[0];** {пусть 0-й элемент - Min}

**Edit3.Text:=' ';**

**if CheckBox2.Checked then Max:=M[0];** {пусть 0-й элемент - Max}

**Edit4.Text:=' ';**

**Sum:=0;**  {обнулить значения Sum}

**Edit5.Text:=' ';**

**CountPlus:=0;** {обнулить значения суммы положительных элементов}

**Edit6.Text:=' ';**

**for I:=0 to N-l do**

**begin**

**if CheckBox1.Checked then** {определить минимальный элемент массива}

**if Min>M[I] then Min:=M[I];**

**if CheckBox2.Checked then** {определить максимальный элемент массива}

**if Max<M[I] then Max:=M[I];**

**if CheckBox3.Checked then** {суммировать элементы массива}

**Sum:=Sum+M[I];**

**if CheckBox4.Checked then** {суммировать положительные

элементы массива}

**if M[I]>0 then CountPlus:=CountPlus+1;**

**end;**

{вывести результаты обработки массива}

**if CheckBox1.Checked then Edit3.Text:=IntToStr(Min);**

**if CheckBox2.Checked then Edit4.Text:=IntToStr(Max);**

**if CheckBox3.Checked then Edit5.Text:=IntToStr(Sum);**

**if CheckBox4.Checked then Edit6.Text:=IntToStr(CountPlus);**

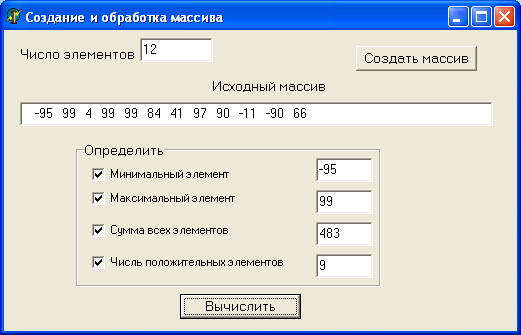
**end;**

15. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.

16. Задавая различные значения числа элементов массива и щелкая мышью на кнопке Создать массив, убедитесь в правильной работе процедуры защиты от ввода нечисловых данных в Edit1 и генерации массива случайных целых чисел.

17. Выбирая варианты обработки массива установкой соответствующих флажков CheckBox, как показано на рис. 6.1, и щелкая мышью на кнопке Вычислить, убедитесь в правильности работы процедуры обработки массива.

18. Закройте окно приложения.

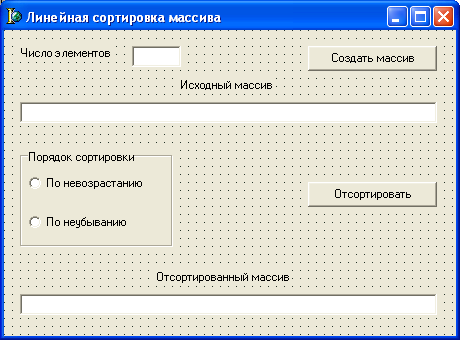


**Рис. 6.1.** Окно приложения создания и обработки линейного массива

**Пример выполнения**

**Задание №2**

**Создайте приложение, которое предлагает пользователю ввести размер массива и создать массив случайных целых чисел, а затем, используя переключатели, указать порядок сортировки (по неубыванию, по невозрастанию), выполнить сортировку и просмотреть отсортированный массив.**



Окно формы приложения линейной сортировки массива

**Алгоритм выполнения работы**

1. Создайте форму, для свойства Caption задайте значение «Линейная сортировка массива».

2. На форме разместите компоненты Edit1, Edit2 и Edit3, кнопку Button1, для свойства Caption которой задайте значение «Создать массив».

3. Удалите текст Edit1, Edit2, Edit3 из соответствующих компонентов.

4. Разместите на форме компоненты Label1, Label2 и задайте для их свойств Caption значения «Число элементов» и «Исходный массив» соответственно.

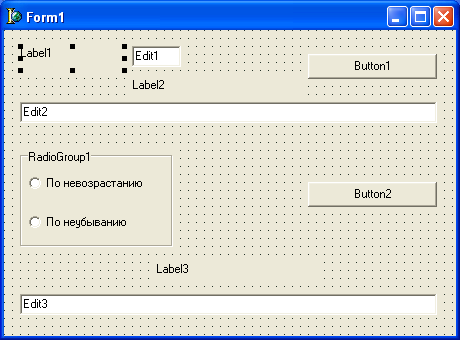
5. Ниже объекта Edit3 на форме разместите панель RadioGroup1, для свойства Caption которой задайте значение «Порядок сортировки».

6. Для выбора порядка сортировки задайте два переключателя в панели RadioGroup1 и подписи к ним. Выбрав в Инспекторе объектов компонент RadioGroup1, на странице свойств выберите свойство Items, затем в окне String List Editor введите список элементов: По невозрастанию, По неубыванию и нажмите ОК.

7. Справа от панели RadioGroup1 разместите кнопку Button2, для свойства Caption которой задайте значение «Отсортировать».

8. В нижней части формы разместите Edit3 для вывода отсортированного массива. Над объектом Edit3 разместите объект Label3, для свойства Caption которого задайте значение «Отсортированный массив».

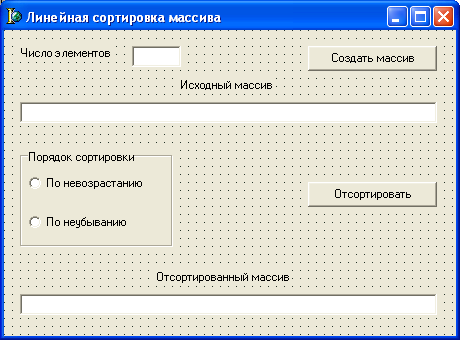
9. Выровняйте компоненты на форме, как показано на рис. 7.1.



**Рис. 7.1.** Окно формы приложения линейной сортировки массива

10. Зафиксируйте положение компонентов на форме, выбрав в меню Delphi команду Правка ► Зафиксировать.

11. Сохраните файл проекта и программного модуля.



**Рис. 7.2.** Окно формы приложения линейной сортировки массива

12. Аналогично предыдущему примеру, прежде чем создавать обработчики событий щелчка мышью по кнопкам Button1 и Button2, опишите глобальные переменные целого типа N и I, где N — размер массива, а I — порядковый номер элемента массива, а также М — динамический массив целых чисел.

**var**

**Form1: TForm1;**

**N, I : integer;**

**M : array of integer;** {описание динамического массива целых чисел}

13. Для предупреждения ошибки ввода в окно Edit1 нечислового значения введите обработку события нажатия клавиши в окне Edit1, чтобы запретить ввод любых символов, кроме цифр от 0 до 9. Для создания процедуры обработчика события нажатия клавиши в окне Edit1 выберите в окне Инспектора объектов компонент Edit1 и на странице События дважды щелкните левой кнопкой мыши на пустом поле списка в событии On Key Press. После этого в текст процедуры обработчика события добавьте следующий оператор: if not (Key in ['0'..'9']) then Key:=#O;. Полный текст процедуры обработчика события будет выглядеть следующим образом:

**procedure Edit1KeyPress(Sender: TObject; var Key: Char);**

**begin**

**if not (Key in ['0'..'9']) then Key:=#0;**

**end;**

14. Создание массива целых чисел опишите в процедуре обработчика события щелчка мышью на кнопке Button1. Для создания процедуры обработчика события выберите в окне Инспектора объектов объект Button1, затем на странице События произведите двойной щелчок на пустом поле списка в событии OnClick. После этого в окне Редактора кода в заготовку процедуры обработчика события введите следующий текст:

**procedure TForm1.Button1Click(Sender: TObject);**

**begin**

**Randomize;**

**N:=StrToInt(Edit1.Text);**  {число элементов массива}  
**SetLength(M, N);** {задать динамическому массиву М длину N}

**Edit2.Text:=' ';**

**for I:= 0 to N-1 do** {заполнить массив случайными значениями целых чисел}

**begin**

**M[I]:= Round(Sin(Random(100))\*100);**

{присвоить элементу массива случайное значение}

**Edit2.Text:=Edit2.Text+' '+IntToStr(M[I]);**

{вывести элементы массива}

**end;**

**end;**

15. Обработка события нажатия кнопки Button2 «Вычислить» начинается с сортировки массива, которую можно записать с помощью оператора цикла for.

Для хранения номера элемента массива, сравниваемого с текущим (имеющим номер I) введем переменную целого типа J. Для запоминания элемента массива при обмене в процессе сортировки введем переменную Тmр.

Линейная сортировка массива выполняется путем перестановки элементов в массиве при соблюдении следующих условий: если в неотсортированной части массива найден элемент с номером J, больший, чем элемент с номером I (для сортировки по невозрастанию), или меньший, чем элемент с номером I (для сортировки по неубыванию). Поэтому можно записать следующий оператор if then else с составным условием:

**if (RadioGroup1.ItemIndex=0) and (M[I]<M[J])**

**or (RadioGroup1.ItemIndex=1) and (M[I]>M[J]) then**

Перестановка элементов массива осуществляется с использованием третьей переменной:

**Тmр:= М[I];** {запомнить на время значение М[1]}

**М[I]:= M[J];**

**M[J]:= Tmp;**

Вывод отсортированного массива в окне Edit3 можно записать оператором:

**for I:=0 to N-1 do**

**Edit3.Text:=Edit3.Text+' '+IntToStr(M[I]);**

В целом текст процедуры сортировки и вывода отсортированного массива будет выглядеть следующим образом:

**procedure TForm1.Button2Click(Sender: TObject);**

**var**

**J, Tmp: integer;**

**begin**

**Edit3.Text: = ' ';**

**for I:= 0 to N - 2 do** {изменять размер неотсортированной части массива}

**for J:=I+1 to N-1 do** {сравниваем поочередно I-й элемент неотсортированной части массива со всеми от I+1-го до конца}

**begin** {выбор операции в зависимости от значения свойства RadioGroup1.ItemIndex}

**if (RadioGroup1.ItemIndex=O) and (M[I]<M[J])**

**or (RadioGroup1.ItemIndex=1) and (M[I]>M[J]) then**

{если в неотсортированной части массива нашли J-й элемент, больший чем I-й (для сортировки по невозрастанию) или меньший чем 1-й(для сортировки по неубыванию)}  
**begin** {обменять местами элементы массива}

**Тmр:= М[I];** {запомнить на время значение М[I]}

**M[I]:= M[J];**

**M[J]:= Tmp;**

**end;**

**end;**

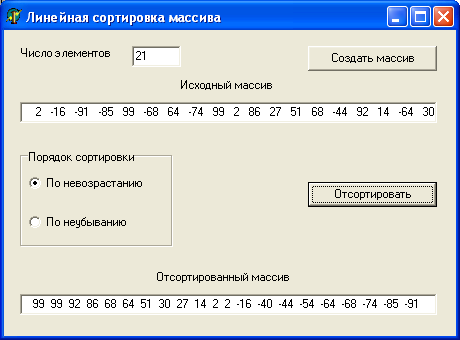
**for I:=0 to N-l do** {вывести отсортированный массив}

**Edit3.Text:=Edit3.Text+' '+IntToStr(M[I]);**

**end;**

16. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.

17. Задавая различные значения числа элементов массива и щелкая мышью на кнопке Создать массив, создавайте линейный массив целых чисел. Выбирая при помощи переключателей в панели RadioGroup1 вариант сортировки и щелкая мышью на кнопке Отсортировать, как показано на рис. 7.3, убедитесь в правильной работе процедуры сортировки и вывода отсортированного массива.



**Рис. 7.3.** Окно приложения линейной сортировки массива

18. После окончания проверки работы приложения закройте его окно.

**Дополнительное задание 1.**

Отредактируйте текст модуля таким образом, чтобы линейная сортировка выполнялась по возрастанию и по убыванию. Что может произойти в этом случае.

**Пример выполнения**

**Задание №3**

***Пример программы обработки одномерных массивов****:*

**1.Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

TLabel – используется для создания комментариев. Для этого использовалось свойство компонента Caption, которое имеет строковый тип.

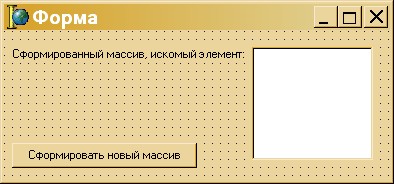
TButton – кнопка, при нажатии которой производятся вычисления. Код вычислений прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Комментарий на кнопке прописывается в свойстве кнопки Caption.

TListBox – используется для вывода результатов программы.

**2.Условие:**

Дан массив xi={3,5;7,2;4,5;2,1;7,6;9,3}. Вычислить *y i*  *x i* \* cos *x i*  *x*4*i*2 **,** найти max yi и его номер k.

**3.Форма:**



**4.Программа:**

unit Unit1; interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls; type

TForm1 = class(TForm)

Button1: TButton;

ListBox1: TListBox; Label1: TLabel;

procedure Button1Click(Sender: TObject); private

{ Private declarations } public

{ Public declarations } end; var

Form1: TForm1; implementation {$R \*.dfm} procedure TForm1.Button1Click(Sender: TObject); const x:array[1..6] of real=(3.5,7.2,4.5,2.1,7.6,9.3); var y:array[1..6] of real; max:real; i,k:integer;

begin max:=-1E5; k:=0;

For i:=1 to 6 do begin

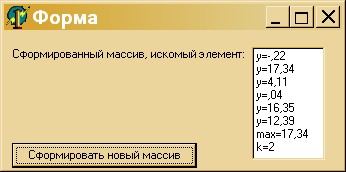
y[i]:=x[i]\*cos(x[i])+x[i]\*x[i]/4;

ListBox1.Items.Add('y='+FormatFloat('###.##',y[i])); if y[i]>max then begin max:=y[i];k:=i; end; end;

ListBox1.Items.Add('max='+FormatFloat('###.##',max));

ListBox1.Items.Add('k='+IntToStr(k)); end; end.

**5.Результат:**



**Пример выполнения**

**Задание №4**

***Пример программы обработки матриц (двумерных массивов)****:*

**1.Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

TLabel – используется для создания комментариев. Для этого использовалось свойство компонента Caption, которое имеет строковый тип.

TButton – кнопка, при нажатии которой производятся вычисления. Код вычислений прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Комментарий на кнопке прописывается в свойстве кнопки Caption.

TStringGrid – данный компонент был использован для вывода исходногй матрицы и модифицированной матрицы. Для вывода элементов матриц использовалось свойство компонента Cells (ячейка), которое характеризуется индексами размещения в столбце и в строке. Количество строк и столбцов прописываются в свойствах компонента RowCount и ColCount. А для перевода числовых значений в строку – функция IntToStr. Задаём следующие свойства:

ColCount– число столбцов (в данной работе 6)

RowCount– число строк (в работе 5)

FixedCols = 0 - фиксированное число столбцов

FixedRows = 0 – фиксированное число строк

StringGrid1 – здесь записывается исходная матрица. В свойствах Options установить свойству редактирования go Editing значение true.

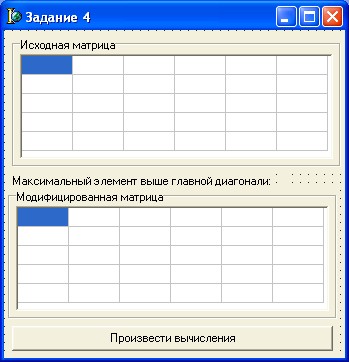
StringGrid2 – здесь записывается полученная матрица.

TGroupBox – компонент, предназначенный для группировки внутри себя других компонентов. Заголовок компонента прописывается в его свойстве Caption.

**2.Условие:**

Определить наибольшее значение среди элементов, расположенных выше главной диагонали матрицы A(5,6). Напечатать новую матрицу A, в которой все элементы выше главной диагонали заменены нулями, кроме наибольшего элемента. Элементы исходной матрицы выбрать самостоятельно.

**3.Форма:**



TLabel

TGroupBox

TStringGrid

TButton

**4.Программа:**

unit Unit1; interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Grids; type

TForm1 = class(TForm)

GroupBox1: TGroupBox;

StringGrid1: TStringGrid;

Button1: TButton;

Label1: TLabel;

Label2: TLabel;

GroupBox2: TGroupBox; StringGrid2: TStringGrid; procedure FormCreate(Sender: TObject); procedure Button1Click(Sender: TObject); private

{ Private declarations } public

{ Public declarations }

end; var

Form1: TForm1;

A:array[1..5,1..6]of integer;

implementation {$R \*.dfm}

procedure TForm1.FormCreate(Sender: TObject); var i,j:integer; begin for j:=1 to 6 do for i:=1 to 5 do

begin

A[i,j]:=random(100);//*Создание случайного числа*

StringGrid1.Cells[j-1,i-1]:=IntToStr(A[i,j]);//*Помещение A[i,j] в ячейку*

*StringRead*

end; end;

procedure TForm1.Button1Click(Sender: TObject); var

i,j:integer;

Max:integer; begin

Max:=A[1,2]; for i:=1 to 5 do for j:=i+1 to 6 do if (A[i,j]>Max)then Max:=A[i,j]; Label2.Caption:=IntToStr(Max);

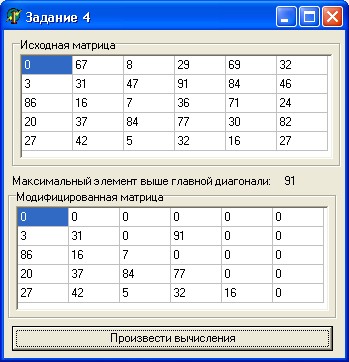
for i:=1 to 5 do for j:=i+1 to 6 do

if (A[i,j]<Max)then A[i,j]:=0; for j:=1 to 6 do for i:=1 to 5 do

begin

StringGrid2.Cells[j-1,i-1]:=IntToStr(A[i,j]); end; end; end.

**5.Результат:**



**Пример выполнения**

**Задание №5**

***Пример программы на программирование с использованием функций и процедур***

**1.Конструирование формы:**

При выполнении задания на форме были размещены следующие компоненты:

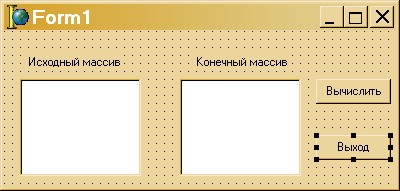
TLabel – используется для создания комментариев. Для этого использовалось свойство компонента Caption, которое имеет строковый тип. TButton – кнопка, при нажатии которой производятся вычисления. Код вычислений прописан в методе OnClick, который срабатывает при нажатии кнопки во время исполнения программы. Комментарий на кнопке прописывается в свойстве кнопки Caption.

TListBox – используется для вывода результатов программы.

**2. Условие.**

Задана последовательность чисел А={A1,A2,…,A6}. Выбрать из них числа, принадлежащие отрезку [x,y]. Из выбранных чисел образовать массив. Решение оформить в виде подпрограммы. Исходные данные: А={-4.1,-1,1.2,3,5,0.75}. X=-2.5, Y=1.2.

**3.Форма:**



**4. Программа.**

unit Unit1; interface uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls; type

TForm1 = class(TForm)

ListBox1: TListBox;

ListBox2: TListBox;

Label1: TLabel;

Label2: TLabel;

Button1: TButton; 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); const A:array [1..6] of real=(-4.1,-1,1.2,3,5,0.75); type mas=array [1..6] of real; var A1,B:mas;i,k:integer;

Procedure mas2(Z:mas;n:integer;var k:integer;var B:mas); var i:integer; const x=-2.5; y=1.2; begin k:=0;

For i:=1 to n do

if (z[i]>=x) and (z[i]<=y) then begin

k:=k+1; B[k]:=z[i]; end;

end; begin for i:=1 to 6 do A1[i]:=A[i]; mas2(A1,6,k,B); for i:=1 to k do

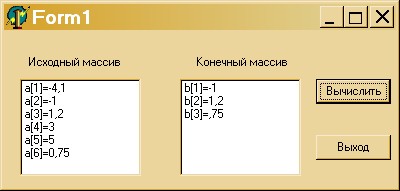
Listbox2.Items.Add('b['+inttostr(i)+']='+formatfloat('##.###',b[i])); for i:=1 to 6 do

Listbox1.items.add('a['+inttostr(i)+']='+floattostr(a[i])); end;

procedure TForm1.Button2Click(Sender: TObject); begin

Application.Terminate; end; end.

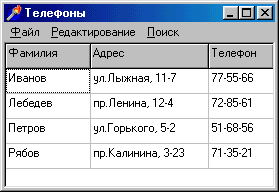
**5.Результат:**

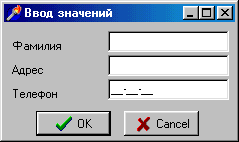


**Задание 1. Телефонная книжка**

**Задача:** Создать приложение для ведения телефонной книжки.

Таблица состоит из трех столбцов: фамилия, адрес, телефон, строки добавляются по мере ввода новых записей. Меню состоит из трех пунктов: "Файл" – включает пункты "Загрузить", "Сохранить", "Выход", "Редактирование" – включает пункты "Добавить", "Удалить", "Отсортировать" и третий пункт "Поиск". При выборе пункта "Добавить" вызывается вторая форма с полями ввода для каждого столбца таблицы. Внешний вид формы представлен на рисунке. При нажатии кнопки "OK" в таблицу добавляется новая строка с введенными во второй форме значениями. При выборе пункта "Удалить" удаляется выделенная запись из таблицы, остальные строки поднимаются вверх. При выборе пункта "Отсортировать" строки таблицы сортируются по фамилии в лексико-графическом порядке. При выборе пункта "Поиск" вызывается вторая форма с одним доступным полем для ввода фамилии. После нажатия кнопки "OK" в таблице выделяется строка, соответствующая введенной фамилии, а если такой фамилии в таблице нет, то выдается сообщение "Объект не найден!".





Компоненты:

StringGrid – таблица, состоящая из трех столбцов, строки добавляются по мере ввода новых записей, одна строка зафиксирована;

MainMenu – основное меню для первой формы;

Edit – окна для ввода фамилии и адреса;

MaskEdit – окно для форматированного ввода номера телефона;

BitBtn – кнопки "OK" и "Cancel" на второй форме;

OpenDialog,

SaveDialog – сохранение и загрузка файла телефонной книги.

**Лабораторная работа №3**

***Визуальное программирование при обработке массивов данных***

**в среде Delphi**

**по дисциплине «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»**

Составитель: Садчиков П.Н.

Разработка и внедрение в учебный процесс новой лабораторной работы с использованием нового приобретенного оборудования и сопроводительных методических рекомендаций для студентов направления подготовки 09.03.02 «информационные системы и технологии» направленности (профиля) «Информационные системы и технологии в строительстве и архитектуре» очной и заочной форм обучения

Печатается в авторской редакции

Подписано к печати \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Уч.-изд. л. \_\_\_\_. Тираж \_\_\_\_\_\_ экз. Заказ №\_\_\_\_\_\_

Отпечатано в Астраханском государственном архитектурно-строительном университете