МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Колледж экономики, управления и права

Методические указания по организации практических занятий и самостоятельной работы студентов

по дисциплине «Основы алгоритмизации и программирования»

Специальность

09.02.04 Информационные системы (по отраслям)
09.02.05 Прикладная информатика (по отраслям)

Ростов-на-Дону
2018

УДК 004 (075.32)

Методические указания по организации практических занятий и самостоятельной работы студентов по дисциплине «Основы алгоритмизации и программирования»

В данных методических указаниях представлены задания и пошаговые инструкции по разработке приложений в среде Delphi, а также даны задания для самостоятельной работы студентов и вопросы для самоконтроля.

Определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных заданий и образцы решения задач, а также список рекомендуемой литературы.

Разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.04 Информационные системы (по отраслям), предназначены для студентов и преподавателей колледжа.

Автор-составитель: С.В.Шинакова

Одобрены решением учебно-методического совета колледжа и рекомендованы к практическому применению в образовательном процессе.

Протокол № 1 от «31» августа 2018 г

Председатель учебно-методического совета колледжа
личная подпись С.В.Шинакова

Ответственный за выпуск к.п.н., заместитель директора колледжа И.И.Джужук

заказ 1087 от 30.10.18

СОДЕРЖАНИЕ

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1

Понятие объектов и классов. Их свойства и методы

Цель занятия: формирование навыка работы с объектами и классами, создание программ с использованием объектов.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание, указанное ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Объекты-переменные (объекты)

В дальнейшем введённый таким образом тип Point (или любой другой класс) можно использовать в программе обычным образом: определять переменные этого типа (как статически - посредством описателя var, - так и динамически, создавая экземпляр переменной этого типа с помощью стандартной процедуры New), работать с полями и т.д.

Методы объекта

Одна из основных идей объектно-ориентированного подхода к программированию: предполагается, что объект содержит не только информацию, но и правила работы с этой информацией, оформленные в виде выполняемых фрагментов.

Такими правилами могут быть, например, операция создания точки (установка значений координат), а также операции "включения" и "выключения" точки и перемещения её в другое место экрана. Подпрограммы, определённые в объектовом типе, называются методами объекта.

Технически определение подпрограмм в объектовых типах делается так: непосредственно в объектовом типе задаются только заголовки подпрограмм-методов, а полные их описания должны быть заданы отдельно, причём имя подпрограммы-метода формируется из имени объектового типа - "хозяина", символа "точка" и имени подпрограммы. Эти методы рассмотрим ниже.

Задание 1

Создать программу, в которой применяется описанный класс Point. Светящаяся точка перемещается по экрану с помощью клавиш со стрелками. Клавишей F1 включается и выключается её видимость.

```
USES Crt, GraphABC;
TYPE
Point = object            {объектовый тип, класс  "точка" }
X, Y: integer;                    {координаты}
Visible: boolean;                 {видимость}
procedure Create (a, b: integer);     {определение координат}
procedure SwithOn;                {вкл. видимость}
procedure SwithOff;               {выкл. видимость}
```

```
procedure Move (dx, dy: integer);      {переместить}
function GetX: integer;          {вернуть координату x}
function GetY: integer;          {вернуть координату y}
function GetV: boolean;          {вернуть видимость}
end;

procedure Point.Create (a,b: integer);
begin
      X:=a; Y:=b;
      Visible := false;
end;

Procedure Point.SwithOn;
begin
      Visible := true;
      PutPixel(X,Y,GetColor);
end;

Procedure Point.SwithOff;
begin
      Visible := false;
      PutPixel(X,Y,GetBkColor);
end;

Procedure Point.Move (dx, dy: integer);
var a,b :integer;
begin
      a:=x+dx;
      b:=y+dy;
      if visible then putpixel(x,y,GetBkColor);
      if a>639 then x:=639 else
      if a<0   then x:=0 else x:=a;
      if b>479 then y:=479 else
      if b<0   then y:=0 else y:=b;
      if visible then putpixel(x,y,GetColor);
end;
function Point.GetX: integer;
 begin
      GetX:=X;
end;

function Point.GetY: integer;
begin
      GetY:=Y;
end;

function Point.GetV: boolean;
begin
      GetV:=Visible;
end;
```

```
VAR
Gd,Gm: integer;
ch: char;
Pt: Point;

BEGIN
       Gd:=Vga;
       Gm:=VgaHi;
       InitGraph(Gd,Gm,'');
       If GraphResult=GrOk then
       Begin
              Pt.Create(320,240);
              Pt.SwithOn;
              Repeat
              ch := readkey;
              if ch=#0 then begin
              ch := readkey;
              case ch of
       {вверх}  #72: Pt.Move(0,-1);
       {вниз}   #80: Pt.Move(0,1);
       {влево}  #75: Pt.Move(-1,0);
{      вправо} #77: Pt.Move(1,0);
       {F1}     #59: if Pt.GetV then Pt.SwithOff else Pt.SwithOn;
       end; {case}
end;
Until ch=#13;
End; Closegraph; END.
```

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Напишите программу с объектом треугольник, передвигающимся из левой части экрана в правую.
2. Напишите программу с объектом "пульсирующая" окружность?
3. Напишите программу с объектом "прямоугольник", размеры которого можно изменять в ходе работы программы.
4. Напишите программу с объектом "линия", передвигающимся с нижней части экрана в верхнюю часть экрана.
5. Напишите программу с объектом "бегущая строка", содержимое которой можно изменять в ходе работы программы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что называют объектовым типом (классом)?
2. Что называют объектом?
3. Что такое инкапсуляция?
4. Как объявляются объектовые типы (классы) в PascalABC?
5. Что называют методами объекта?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2

Основополагающие принципы объектно-ориентированного программирования

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание 1.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Наследование

Следующее важное свойство объектовых типов позволяет при построении нового объектового типа использовать некоторый ранее определённый объектовый тип. Пусть, например, необходимо построить объектовый тип, управляющий простой геометрической фигурой - кругом - на экране дисплея.

Ясно, что структура информация для определения круга очень похожа на описанную в предыдущем разделе структуру для точки: здесь также необходимы поля X и Y для фиксации центра круга в текущий момент. Нужно ещё добавить поле для хранения величины радиуса круга. Традиционный стиль программирования допускает два решения. Во-первых, можно ввести для круга совершенно новую структуру, повторив в ней (может быть, под другими именами) те же поля X, Y и Visible и добавив новое поле Radius. Во-вторых, можно сконструировать структуру для круга, использовав в ней поле с типом, ранее определённым для точки (сделать "структуру в структуре"). Оба подхода вполне приемлемы, однако объектно-ориентированное программирование предполагает иной подход, который по ряду причин является гораздо более предпочтительным.

Объектно-ориентированный стиль позволяет определить новый объект как потомок другого ранее определённого типа. Это означает, что новый тип автоматически получает все поля и методы ранее введённого типа, который в этом случае называется предком или родительским типом.

В этом случае в определении типа-потомка должно быть указано (в круглых скобках после служебного слова object) имя родительского типа:

```
Circle = object (Point)
Radius: integer;
end;
```

Задание родительского типа означает, что в объектовом типе Circle неявно присутствуют все поля из типа Point; аналогично, для переменной нового типа доступны все методы из Point:

```
var
OneCircle: Circle;
begin
OneCircle.Init(100,200);
```

```
OneCircle.Radius := 30;
```

Описанное свойство наследования характеристик одного объекта другим называется *наследованием* является одним из основных свойств ООП и широко используется в объектно-ориентированном программировании. Один тип может являться предком для произвольного числа типов-потомков, в то время как любой объектовый тип может наследовать поля и методы только одного типа-родителя, который указывается в круглых скобках.

Тип потомок может, в свою очередь, выступать как предок по отношению к другому объектовому типу (типам). Так, можно определить фигуру "кольцо", состоящую из двух концентрических кругов:

```
type
Ring = object (Circle)
Radius2: integer;
end;
```

Полиморфизм

Тип Ring наследует поле Radius из своего непосредственного родителя Circle, а также поля и методы из типа Point, который также считается (косвенным) предком для Ring. Длина такой цепочки наследования в языке никак не ограничивается.

По правилу наследования тип Circle имеет в своём составе методы объекта-предка Point. Однако, легко видеть, что методы SwithOn, SwithOff не подходят для рисования круга. Поэтому, полное описание типа Circle должно содержать также собственные методы для рисования и удаления круга и его передвижения по экрану. Самым простым решением было бы ввести в новый тип такие методы, дав им некоторые новые имена.

Но объектно-ориентированный подход позволяет определить новые методы СО СТАРЫМИ именами, ПЕРЕОПРЕДЕЛИВ тем самым методы типа-родителя.

Сразу отметим, что переопределять можно только методы; поля, указанные в родительском типе, безусловно наследуются типом-потомком и не могут быть в нём переопределены (то есть имена полей потомка не должны совпадать с именами полей типа-предка). Кроме того, новый метод в типе потомке может иметь совершенно другие параметры, нежели одноимённый метод из типа-предка.

Возможность иметь в одной программе несколько подпрограмм-методов для разных объектов называется в ООП полиморфизмом, и является одним из основных свойств ООП.

Компилятор, не обнаружив вызываемого метода, просматривает определение родительского типа; если данный метод и здесь не определён, то аналогично просматривается родитель родителя и т.д. Если очередной объектовый тип не имеет предка, а метод не обнаружен, то компилятор фиксирует ошибку.

Механизм наследования, являясь достаточно простым для понимания и использования, предоставляет широкие возможности при разработке программ. Имея несколько "базовых" объектовых типов (например, в интерфейсном разделе модуля), можно на их основе конструировать новые объекты, добавляя в них новые поля и расширяя и/или переопределяя соответствующие методы.

Правило совместимости типов по присваиванию, действующее для случая объектовых типов, формулируется достаточно просто: совместимыми по присваиванию являются, кроме эквивалентных типов (то есть объявленных в виде T1=T2), объектовые

типы, состоящие в отношении наследования, причём присваивание может происходить в направлении ОТ типа-потомка. К родительскому типу, НО НЕ НАОБОРОТ.

В подобных случаях копируются (присваиваются) только те поля, которые являются общими для обоих типов. Такое же правило действует и для указателей на объектовые типы.

Задание 1. Написать программу с объектом "линия" - потомком объекта "точка".

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Напишите программу с объектом "прямоугольный треугольник" - потомком объекта точка.
2. Напишите программу с объектом "прямоугольник" - потомком объекта "точка"
3. Напишите программу с объектом "пульсирующая окружность" - потомком объекта "точка".

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какое свойство ООП называют наследованием?
2. Какое свойство ООП называют полиморфизмом?
3. Объясните понятия объект-предок и объект-потомок.
4. Можно ли переопределять поля объекта-предка?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 3

Знакомство со средой Delphi

Цель занятия: формирование навыка работы со средой программирования Delphi, разработка приложений с линейным алгоритмом.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание, указанное ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Delphi – это высокоэффективная графическая среда для визуальной разработки программ.

Интерфейс Delphi

1 - Главное окно Delphi (оно имеет заголовок «Borland Delphi for Microsoft Windows »).

2 - Окно Формы (заголовок «Form1»).

Окно Формы представляет собой своеобразную макетницу для сборки на ней окна разрабатываемого приложения. Сама Форма и все то, что находится в её поле, с точки

зрения Delphi, представляют собой *объекты*. Каждому объекту Delphi присваивает своё уникальное имя, которое, при необходимости, можно изменить.

3 - Окно Кода Программы (заголовок «Unit1.pas»).

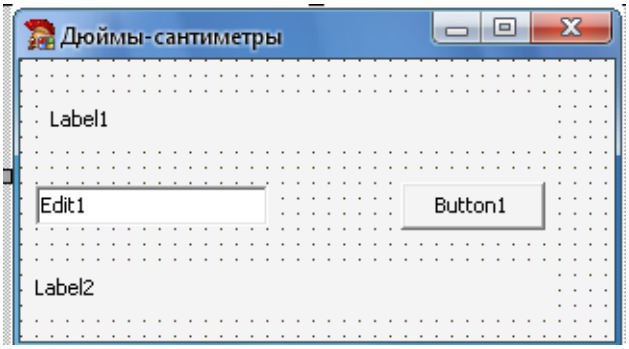
4 - Окно Инспектора Объектов (заголовок «Object Inspector»)

Окно Инспектора Объектов содержит две вкладки – Properties(Свойства) и Events (События).

С помощью вкладки Properties можно просматривать и изменять характеристики (свойства) выделенного объекта. Сразу после запуска Delphi на этой вкладке находятся свойства такого объекта, как сама Форма Form1: заголовок (Caption), имя объекта (Name), его геометрические размеры (Width, Height), цвет (Color) и др.

С помощью вкладки Events можно описать реакцию выделенного объекта на то или иное событие (например, как он будет реагировать на щелчок мыши или нажатие клавиши).

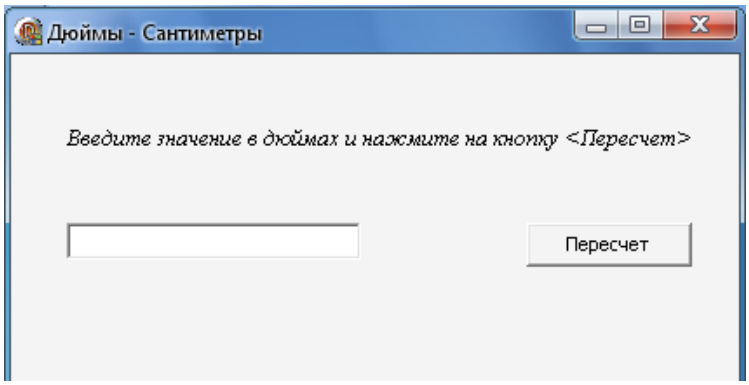
Задание 1. **Создать простейшее приложение в Delphi - программу пересчета.**

1. Создайте папку для файлов проекта «Duimi-sm».	А) File/New/VCL Forms Application Б) File/Save Project As В окне Save Unit1 As создаем папку «Дюймы-сантиметры», открываем ее и задаем имя файла «Main» В) Сохранить Г) Задаем имя файла проекта «Duimi-sm» Д) Сохранить
2. Студенты самостоятельно переименовывают форму (Form1 ➔ «Дюймы-сантиметры»), размещают на форме компоненты (Edit1, Label1, label2, Button1) как показано на рисунке: 	
3. Расположенные объекты выровнять по горизонтали.	При нажатой клавише SHIFT выделяем компоненты и в контекстном меню задаем команду: Position/Align/Horizontal/Centers
4. Удалить текст Edit1 в соответствующих компонентах.	ObjectInspector → Properties → Text, удаляем текст.
5. Переименовать Label1	ObjectInspector → Properties →Caption, пишем Введите количество дюймов и щелкните на <Пересчет>
6. Переименовать Button1	ObjectInspector → Properties →Caption,

	пишем Пересчет
7. Удалить текст Label2 и изменить свойства выводимого текста в соответствующих компонентах	ObjectInspector → Properties → Caption, удаляем текст. Меняем размер шрифта Properties →Font →Size Меняем стиль шрифта Properties →Style →fsBold → True

8. Запрограммировать кнопку «Пересчет»	VAR
	Stroka:String; {для формирования результата}
	D:Real; {Дюймы (исходные данные)}
	S:Real; {Сантиметры (результат расчета)}
	ErrCode:Integer; {код ошибки преобразования
	текстовой строки в число}
	Begin
	VAL(Edit1.Text, D, ErrCode);
	S:=D * 2.54;
	STR(S:5:3, Stroka);
	Stroka:=Concat('В ',Edit1.Text,' дюймах содержится ',
	Stroka, ' сантиметров');
	Label2.Caption :=Stroka;
	End;

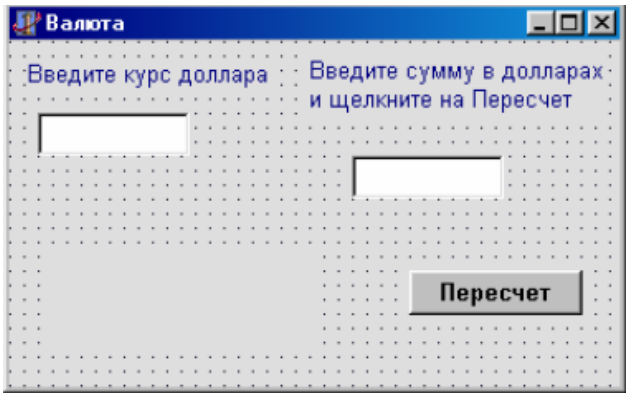
Приложение имеет вид:



САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Изменить значок приложения, создав, например значок с изображением буквы «К» в белом круге.
2. Создать проект для пересчета суммы:
   а) из долларов по данному курсу в рубли.
   б) из евро по данному курсу в рубли
   в) из рублей по данному курсу в евро
   г) из долларов по данному курсу в евро

Окно работающего приложения



Указание:
А) задание *а* выполняют студенты, номер которых в списке группы 1, 5, 9, 13, 17, 21, 25;
Б) задание *б* выполняют студенты, номер которых в списке группы 2, 6, 10, 14, 18, 22;
В) задание *в* выполняют студенты, номер которых в списке группы 3, 7, 11, 15, 19, 23;
Г) задание *г* выполняют студенты, номер которых в списке группы 4, 8, 12, 16, 20, 24;

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение Delphi.
2. Сколько окон содержит запущенная на исполнение Delphi?
3. Назначение Главного меню Delphi.
4. Назначение Панели командных кнопок.
5. Назначение Палитры компонентов.
6. Назначение Окна Формы.
7. Назначение Окна Кода Программы.
8. Где находится вкладка Properties?
9. Где находится вкладка Events?
10. Каково назначение вкладки Events?
11. Каково назначение вкладки Properties?
12. Что такое Caption?
13. Что такое объект Delphi?
14. Можно ли изменить имя объекта Delphi, если «да», то как?
15. Зачем нужен компонент «Label» и где он находится?
16. Зачем нужен компонент «Edit» и где он находится?
17. Зачем нужен компонент «Button» и где он находится?
18. Как увеличить размер шрифта объекта Label?
19. Как у объекта Label сделать полужирный шрифт?
20. Как у объекта Label сделать шрифт красного цвета?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 4

Основы визуального программирования

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке линейных программ: изменение свойств формы, размещение в ней компонентов - Label, Edit, Button; компиляция, сборка и выполнение программ; создание и отладка линейной программы.

Этапы выполнения работы:

1. Под руководством преподавателя студенты производят настройку в среде Delphi.
2. Все студенты на занятии выполняют задание, указанное ниже.
3. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
4. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 Создать приложение, вычисляющее сумму двух чисел

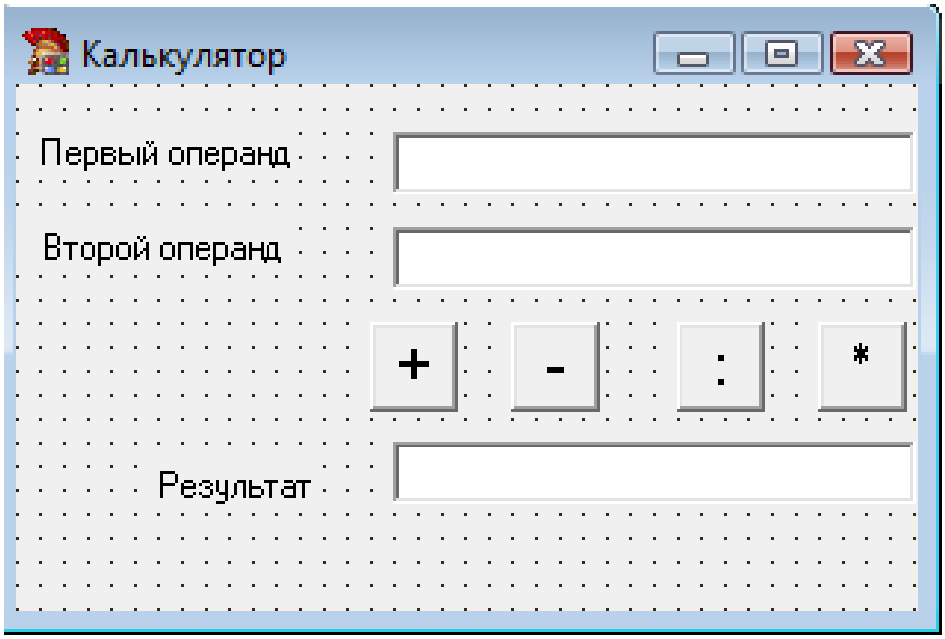
1. Создание нового проекта	File/New/VCL Form Application
2. Сохранение нового проекта	1) File/Save As а) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Сумма чисел»; б) Открыть эту папку; в) Нажать кнопку сохранить; 2) File/Save Project As а) Задать имя файла summa и нажать сохранить
3. Изменение свойств формы а) размер; б) заголовок формы; в) цвет окна; г) изменение цвета, типа, стиля и размера шрифта в заголовке формы.	а) с помощью мыши измените размер формы; б) измените заголовок формы с Form1 на «Сумма двух целых чисел» в окне Object Inspector -> свойство Caption; в) в окне Object Inspector -> Color/clWindow; г) в окне Object Inspector -> Font/Color Font/Style Font/Size Font/Name
4. Добавление компонентов на форму	Двойным щелчком на компоненте (в палитре компонентов) добавить TEdit1, …, TEdit3, TLabel1, TButton1
5. Изменение свойств а) заголовок метки; б) заголовок кнопки; в) размер шрифта; г) выравнивание компонентов на форме; д) удаление текста в Edit; е) добавление компонентов; ж) изменение заголовка добавленных	Щелкните по компоненту, чтобы выделить его а) Label1: Caption/«+»; б) Button1: Caption/«=»; в) Компонент: Font/Size/<значение>; г) При нажатой клавише Shift выделить нужные компоненты, в контекстном меню выбрать команду Position -> Align. В окне Alignment (Выравнивание) в панели Vertical выбрать Centers –> Ok;

компонентов.	д) Object Inspector/Text/удалить текст; е) Добавить 3 метки. Для этого: двойной щелчок по объекту TLabel: ж) Object Inspector/Caption/«Первое слагаемое» «Второе слагаемое» «Сумма» Разместить компоненты над объектами Edit 1-3
6. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
7. Фиксация положения компонентов на форме	Edit/Lock Controls (для отмены повторно задать команду)
8. Сохранение изменений	Save All (на стандартной панели инструментов)
9. Компиляция проекта	Project/Compile summa
10. Просмотр файлов проекта	Откройте папку «Сумма двух целых чисел» 1) summa.dpr – файл проекта содержит код на Object Pascal; с него начинается выполнение программы, обеспечивает инициализацию других модулей; 2) Unit1.pas – файл модуля; 3) Unit1.dfm – файл формы; 4) Summa.dof – файл параметров проекта; 5) Summa.res – файл ресурсов; 6) Unit1.~dfm, Unit1.~pas – файлы резервных копий; 7) Unit1.dcu – откомпилированный модуль; 8) summa.exe – откомпилированный проект
11. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
12. Расположение окна приложения	Object Inspector/ Position /poScreenCenter

13. Создание кода – обработчика события	Для того, чтобы приложение выполнило вычисления при щелчке по кнопке Button1 («=») надо написать кол обработки этого события.
	Object Inspector/Events/OnClick/двойной щелчок на пустом поле списка.
	Курсор будет находиться в процедуре между словами begin и end. В данную заготовку размещаем необходимые операторы, в разделе описаний вводим следующее:
	`var a1,a2,sum:integer;`
	
	Основное тело процедуры будет иметь вид:
	
	`a1:=StrToInt(Edit1.Text); {*}`
	`a2:=StrToInt(Edit2.Text);`
	`sum:=a1+a2;`
	`Edit3.Text:=IntToStr(sum); {**}`
	
	* преобразование строкового типа в целое число;
	** преобразование целого числа в строковый тип;

	Замечание: чтобы получить справку по какому-либо слову кода (например, StrToInt), надо установить курсор на это слово и нажать F1
14. Сохранение изменений	Save All
15. Запуск программы	Run (F9)
16. Закрытие окна проекта	Alt+F4
17. Открытие окна проекта	1 способ – File/OpenProject 2 способ – File/Reopen/summa (открытие последнего редактированного проекта)
18. Изменение свойств визуальных компонентов	Самостоятельно: изменить свойства компонентов Label2, Label3, Label4, задав для них следующие параметры: - стиль шрифта: курсив - цвет шрифта: синий - размер шрифта: 10
19. Изменение значка приложения	1) Для вызова редактора изображений: Tools/ImageEditor; в окне редактора команда: File/Open – открыть файл ресурсов; 2) Выберите папку с проектом, в поле «Тип файла» выберите Resource/DCR files (*.res, *.dcr), в списке файлов выберите файлы ресурсов приложения (summa.res) и нажмите Открыть; 3) + Icon/+ MainIcon/выбрать значок; 4) File/Close (закрывает окно редактирования значка); 5) File/Save, File/Exit
20. Запустите приложение на исполнение	

Задание 2 **Разработать приложение «Калькулятор»**



Алгоритм выполнения задания

1. Создание нового проекта	File/New/VCL Form Application
2. Сохранение нового проекта	3) File/Save As г) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Калькулятор»; д) Открыть эту папку; е) Нажать кнопку сохранить; 4) File/Save Project As б) Задать имя файла calk и нажать сохранить
3. Изменение свойств формы а) размер; б) заголовок формы; в) цвет окна; г) изменение цвета, типа, стиля и размера шрифта в заголовке формы.	а) с помощью мыши измените размер формы; б) измените заголовок формы с Form1 на «Калькулятор» в окне Object Inspector -> свойство Caption; в) в окне Object Inspector -> Color/clWindow; г) в окне Object Inspector -> Font/Color Font/Style Font/Size Font/Name
4. Добавление компонентов на форму	Двойным щелчком на компоненте (в палитре компонентов) добавить TEdit1, …, TEdit3, TLabel1, TLabel2, TLabel3, TButton1, …,TButton4
5. Изменение свойств а) заголовок метки; б) заголовок кнопки; в) размер шрифта; г)выравнивание компонентов на форме; д) удаление текста в Edit;	Щелкните по компоненту, чтобы выделить его а) Label1 (Label2): Caption/ Первый операнд (Второй операнд); б) Button1 (Button2, Button3, Button4): Caption/«+» («-», «:», «*»); в) Компонент: Font/Size/<значение>; г) При нажатой клавише Shift выделить нужные компоненты, в контекстном меню выбрать команду Position -> Align. В окне Alignment (Выравнивание) в панели Vertical выбрать Centers –> Ok; д) Object Inspector/Text/удалить текст.
6. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
7. Фиксация положения компонентов на форме	Edit/Lock Controls (для отмены повторно задать команду)
8. Сохранение изменений	Save All (на стандартной панели инструментов)
9. Проверка синтаксиса	Project/ Syntax Check calk
10. Компиляция проекта	Project/Compile calk
11. Просмотр файлов проекта	Откройте папку «Сумма двух целых чисел» 9) summa.dpr – файл проекта содержит код на Object Pascal; с него начинается выполнение программы, обеспечивает инициализацию других модулей; 10) Unit1.pas – файл модуля; 11) Unit1.dfm – файл формы; 12) Summa.dof – файл параметров проекта; 13) Summa.res – файл ресурсов; 14) Unit1.~dfm, Unit1.~pas – файлы резервных копий; 15) Unit1.dcu – откомпилированный модуль; 16) summa.exe – откомпилированный проект
12. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
13. Расположение окна	Object Inspector/ Position /poScreenCenter

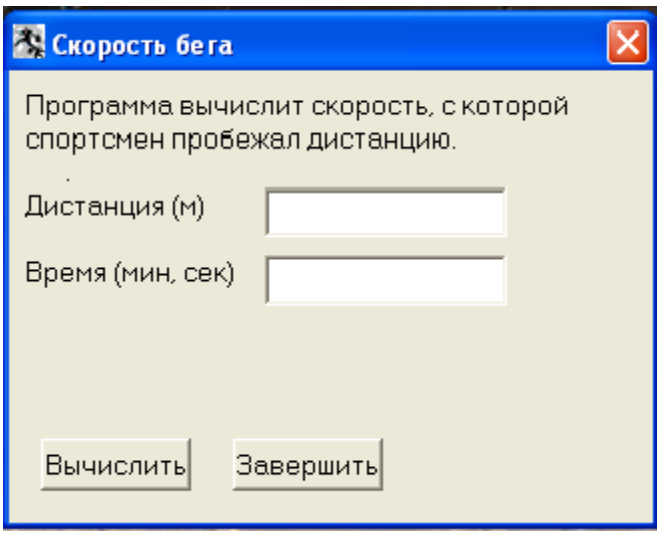
приложения	

14. Создание кода – обработчика события	Для того, чтобы приложение выполнило вычисления при щелчке по кнопке Button1 («+») надо написать код обработки этого события.
	Object Inspector/Events/OnClick/двойной щелчок на пустом поле списка.
	Курсор будет находиться в процедуре между словами begin и end. В данную заготовку размещаем необходимые операторы.
	Основное тело процедуры будет иметь вид:
	procedure TForm1.Button1Click(Sender: TObject);
	begin
	Edit3.Text:=IntToStr(StrToInt(Edit1.text)+StrToInt(Edit2.t xt));
	end;
	Аналогично программируем остальные кнопки.
	Замечание: чтобы получить справку по какому-либо слову кода (например, StrToInt), надо установить курсор на это слово и нажать F1

15. Сохранение изменений	Save All
16. Запуск программы	Run (F9)
17. Закрытие окна проекта	Alt+F4
18. Открытие окна проекта	1 способ – File/OpenProject 2 способ – File/Reopen/calk (открытие последнего редактированного проекта)
19. Изменение свойств визуальных компонентов	Самостоятельно: изменить свойства компонентов Label2, Label3, Label4, задав для них следующие параметры: - стиль шрифта: курсив - цвет шрифта: синий - размер шрифта: 10
20. Изменение значка приложения	6) Для вызова редактора изображений: Tools/ImageEditor; в окне редактора команда: File/Open – открыть файл ресурсов; 7) Выберите папку с проектом, в поле «Тип файла» выберите Resource/DCR files (*.res, *.dcr), в списке файлов выберите файлы ресурсов приложения (calk.res) и нажмите Открыть; 8) + Icon/+ MainIcon/выбрать значок; 9) File/Close (закрывает окно редактирования значка); 10) File/Save, File/Exit
21. Запустите приложение на исполнение	

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Внести в приложение «Калькулятор» изменение: организовать вывод результата с помощью компонента Label.
2. Создать приложение для расчета: площади квадрата, площади прямоугольника, площади круга.
3. Создать приложение, которое позволяет вычислять скорость спортсмена на беговой дорожке.



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5

Основы визуального программирования

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением условного оператора, изменение свойств формы, размещение в ней компонентов - Label, Edit, Button; компиляция, сборка и выполнение программ.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание указанное ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 **Разработать приложение, позволяющее находить частное от деления двух целых чисел.**

1. Самостоятельно создать форму с необходимыми на ней компонентами, изменить свойства объектов, создать код обработчика события, протестировать приложение.

```
VAR
        a,b:integer;
        c:real;
BEGIN
        A:=StrToInt(Edit1.Text);
        B:= StrToInt(Edit2.Text);
        C:=a/b;
        Edit3.Text:=FloatToStr(c);
```

Код обработчика событий.

```
END;
```

2. Дополнить программу так, чтобы перед операцией деления выполнялась проверка делителя на равенство нулю. Если делитель будет равен нулю, то в окне Edit3 должно появиться сообщение «На нуль делить нельзя!».

В процедуру обработчика события TForm1.ButtonClick добавить оператор If.

```
If b<>0 then
        Begin
        C:=a/b;
        Edit3.Text:=FloatToStr(c);
        End
Else Edit3.Text:='На нуль делить нельзя!';
End;
```

3. Сохранить текст модуля под именем main1.pas (File/Save AS). Сохранить проект – delenie1 (File/Save As Project).

4. Ctrl+F12 (View/Units), выбрать программу delenie1 и изменить описание раздела Uses:

```
Uses
        Forms,
        Main1 in 'main1.pas' {Form1};
```

5. Откомпилируйте программу (Project/Compile). Проверьте работу приложения.

6. Выполните отладку приложения в пошаговом режиме.

Команда View/Debug Windows/Watches открывает окно просмотра значений переменных и выражений.

Для включения переменной или выражения в окно просмотра надо выделить в окне редактора кода эту переменную, выражение и нажать CTRL+F5.

В тексте модуля main1 выделите a, b, b<>0, Edit3 и перетащите их в окно Watches List.

Выполните приложение в пошаговом режиме (Клавиша F7).

7. Сделать сообщение «На нуль делить нельзя!» шрифтом красного цвета:

```
BEGIN
        Edit3.Font.Color:=clRed;
        Edit3.Text:='На нуль делить нельзя!';
END;
```

8. Откомпилируйте и запустите на выполнение приложение.

Замечание: В окне Edit3 после вывода сообщения «На нуль делить нельзя!» цвет шрифта все равно остается красным, даже если делитель уже не равен нулю.

9. Восстановить черный цвет шрифта в окне Edit3. Перед применением проверки условия b<>0 пропишите строку:

```
Edit3.Font.Color:= clBlack;
```

10. Откомпилировать и запустить на исполнение приложение.

Замечание: Сообщение в окне Edit3 отобразится не полностью.

11. Изменить программу так, чтобы размер окна Edit3 изменялся в ходе выполнения программы.

В окне Object Inspector для объекта Edit3 установить значение свойства Width=130, а перед оператором if прописать: Edit.Width:=81;

12. Изменить программу так, чтобы сообщение «На нуль делить нельзя!» выводилось в отдельном окне.

Для вывода текста в отдельном окне сообщения используют вызов процедуры:

ShowMessage(const Msg:string);

1) Сохраните проект под именем delinie2, а текст программы по именем main2.pas.

```
BEGIN
    A:=StrToInt(Edit1.Text);
    B:= StrToInt(Edit2.Text);
    Edit3.Text:= ' ';
    If  b=0 then ShowMessage ('На нуль делить нельзя!')
    Else c:=a/b;
    Edit3.Text:=FloatToStr(c);
End;
```

2) View/Units; Изменить раздел описания uses в программе delenie2:

```
Uses
    Forms,
    Main2 in 'main2.pas' {Form1};
```

3) Сохранить проект, откомпилировать программу, запустить приложение на исполнение.

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Создать приложение, позволяющее решать квадратное уравнение, заданное коэффициентами. Создать обработчик события, который будет подготавливать окно к новому расчету (кнопка "Новое"). Осуществить выход из приложения по кнопке "Выход".
2. Составить приложение для вычисления стоимости телефонного разговора в зависимости от дня недели. Цена одной минуты разговора 1 руб. в субботу и воскресенье скидки 50%.
3. Создать приложение для определения оптимального веса. Программа запрашивает Ваш вес и рост, вычисляет оптимальное для Вас значение веса (рост минус сто), сравнивает его с реальным и выводит одно из следующих сообщений:

- Ваш вес оптимален
- Вам надо поправиться на … кг
- Вам необходимо похудеть на … кг

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6

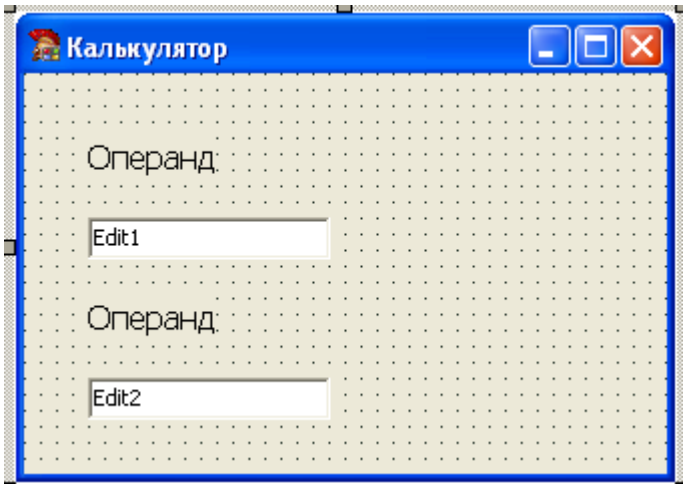
Структура программ Delphi. Управление проектами

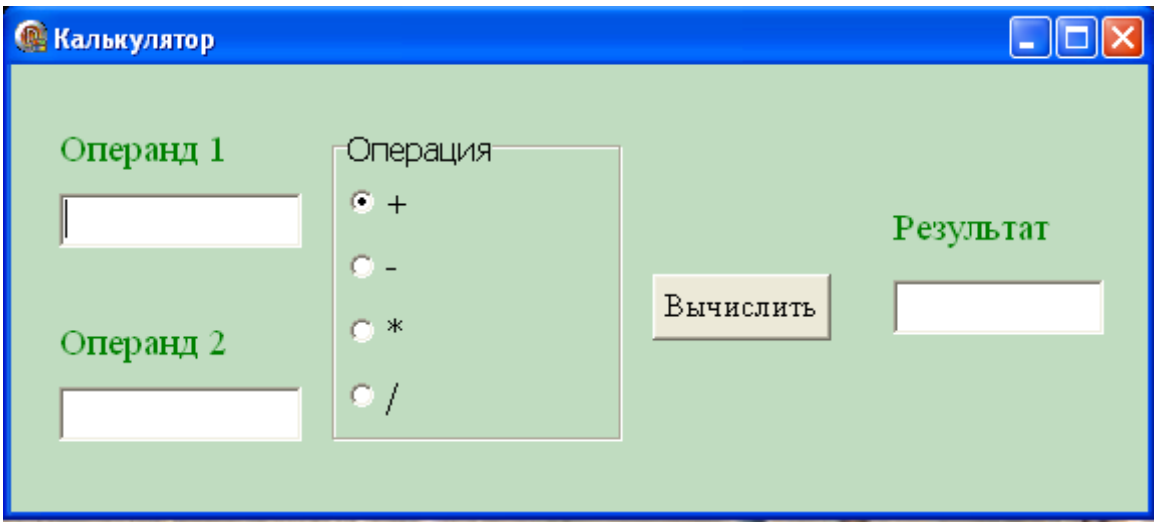
Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонента RadioGroup, создание и отладка программы, содержащей оператор case.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задания, указанные ниже.
2. Выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 **Создать приложение, которое позволяет которое позволяет вводить два целых числа и выполнять над ними арифметические операции. Для выбора операции используйте переключатели, вывод сообщения об ошибке при вводе делителя, равного нулю, отображается в отдельном окне сообщения.**

1. Создайте папку для файлов проекта «Калькулятор».	А) File/New/VCL Forms Application Б) File/Save Project As В окне Save Unit1 As создаем папку «Калькулятор», открываем ее и задаем имя файла «Main» В) Сохранить Г) Задаем имя файла проекта «Calculator» Д) Сохранить
2. Студенты самостоятельно переименовывают форму (Form1 ➔ «Калькулятор»), размещают на форме компоненты (Edit1, Edit2, Label1, label2) как показано на рисунке: 	

3. Расположенные объекты выровнять по горизонтали.	При нажатой клавише SHIFT выделяем компоненты и в контекстном меню задаем команду: Position/Align/Horizontal/Centers
Замечание1: Для выбора одной из четырех опций над операндами используйте переключатели, размещенные на панели RadioGroup.	
4. Добавить компонент RadioGroup1 и изменить его свойства.	1) RadioGroup1 → Caption → Операция 2) ObjectInspector → RadioGroup1 → Properties → Items (список элементов). 3) В окне String List Editor введите список элементов: *, +, /, - и нажмите ОК. 4) ObjectInspector → RadioGroup → ItemIndex → 0 Чтобы сделать первую кнопку выбранной по умолчанию. 5) RadioGroup.Font.Size → 12
5. Добавить недостающие компоненты на форму, чтобы она приняла следующий вид: 	
6. Удалить текст Edit1, Edit2, Edit3 в соответствующих компонентах.	ObjectInspector → Properties → Text, удаляем текст.
7. Самостоятельно выровнять компоненты Label1, Label2, Label3 по горизонтали и зафиксировать компоненты на форме	
❖ Свойство Items определяет количество переключателей в группе и надписи около них.	

8. Запрограммировать кнопку «Вычислить»	(Для получения подсказки по синтаксису оператора Case укажите на этот оператор мышью и нажмите F1).
	`VAR a,b:integer;`
	`c:real;`
	`BEGIN`
	`A:=StrToInt(Edit1.Text);`
	`B:= StrToInt(Edit2.Text);`
	`Edit3.Text:='';  {очистка от`

	результата
	предыдущих вычислений}
	Case RadioGroup1.ItemIndex of
	0: c:=a+b;
	1: c:=a-b;
	2: c:=a*b;
	3: if b=0 then ShowMessage ('На ноль делить нельзя!!!')
	else c:=a/b;
	END;
	If RadioGroup1.ItemIndex <>3 then
	{вывод результата операций}
	Edit3.Text:=FloatToStr(c)
	Else
	Edit3.Text:=FloatToStrF(c, ffgeneral, 10, 4);
	End;

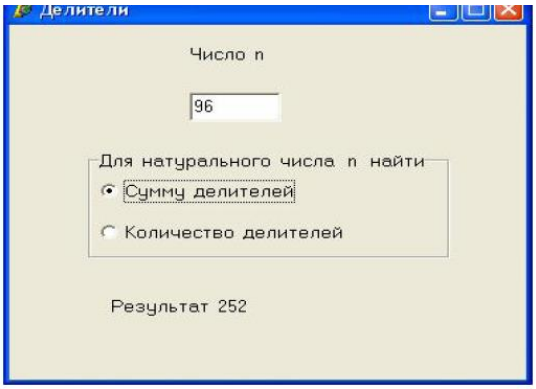
9. Откомпилируйте и запустите приложение на исполнение	а) Тестируем, указывая два операнда. б) Не задаем значение операндов. **Замечание:** В случае б) выводится окно сообщения о некорректности значения операнда
10. Отредактировать текст модуля таким образом, чтобы перед выполнением вычислений выполнялась проверка, заданы ли значения операндов. Если значения не заданы, то следует вывести сообщение об этом в отдельном окне.	Помещаем перед оператором присваивания A:=StrToInt(Edit1.Text); Строку If (Edit1.Text <> '') and (Edit2.Text <> '') Then Begin … А перед последним оператором END прописать следующее: Else ShowMessage ('НЕ ЗАДАНЫ ЗНАЧЕНИЯ');
11. Сохранить, откомпилировать и запустить приложение на исполнение. Проверить работу для случая когда: а) не заданы значения; б) введены, не цифры, а другие символы.	
12. Создать обработчик события, запрещающего ввод любых символов, крое цифр от 0 до 9 и знаков + и -.	ObjectInspector → Edit1 → Events → OnKeyPress → двойной щелчок левой кнопкой мыши на пустом поле списка. После этого окно редактора 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. **Замечание:** Различаются символы в верхнем и нижнем регистрах и символы кириллицы и латиницы. Клавиши SHIFT, CTRL, ALT, функциональные клавиши не вызывают этого события, поэтому нажатие комбинации, например, SHIFT+A соответствует нажатию на символ A. Чтобы распознать комбинации применяют другие обработчики событий: OnKeyDown, OnKeyUp. В тело процедуры TForm1.Edit1KeyPress Вставляем следующий оператор: If not (Key in [‘0’..’9’,’+’,’-’]) then Key := #0; Аналогично создаем процедуру обработчика события нажатия на клавишу в окне Edit2. Чтобы воспринималась клавиша BS необходимо в описанное выше множество добавить элемент #08.
13. Модифицировать обработчик события так, чтобы знаки + и – воспринимались только вначале числа	if (Key='+') and (length(a)<>0) then Key := #0; if (Key='-') and (length(a)<>0) then Key := #0;
14. Сохранить, откомпилировать и запустить приложение на исполнение.	Вводим в качестве операндов не цифры, а другие символы.
Примечание: При обработке таких исключительных ситуаций Delphi использует специальные объекты – исключения, характеризующие возникшую в программе исключительную ситуацию.	

Задание 2 **Описать каждый файл, созданный в папке с приложением: название, назначение. Открыть окно Менеджер проектов. Описать назначение вкладок окна.**

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Создать приложение для расчета характеристик вектора: сумма двух векторов, произведение двух векторов. Выбор осуществить с помощью компонента TRadioGroup.
2. Создать оконное приложение, позволяющее для натурального числа n, введенного в поле Edit, выполнить действие, которое можно выбрать с помощью компонента RadioGroup. Окно приложения имеет вид:



3. Создать оконное приложение, позволяющее менять цвет формы. Выбор цвета осуществить с помощью RadioGroup.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7

Введение в Object Pascal

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонентов BitBtn, RadioButton, Image.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание указанное ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 **Разработать приложение «Денежный калькулятор».**

Программа должна выполнять функции специализированного калькулятора по пересчету денежных сумм из рублей в доллары США и обратно. Предусмотреть возможность ввода нового курса доллара в соответствии с информацией ЦБ РФ. Встроить в проект справочную информацию. Рекомендуемый интерфейс программы изображён на рисунке 1.

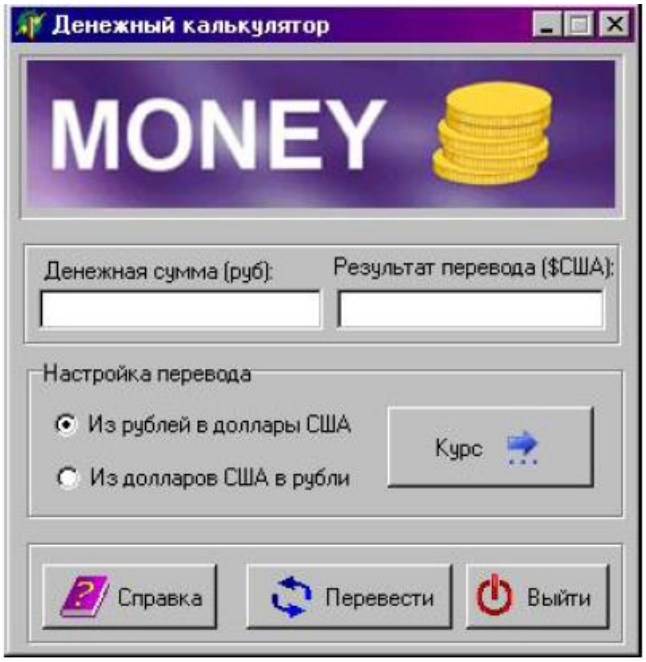


Рисунок 1 – Интерфейс приложения

Разработка интерфейса

1. Измените содержимое Строки заголовка в окне Формы с «Form1» на «Денежный калькулятор». Сохраните проект под именами unit1.pas (для первого модуля) и money.dpr (для проекта). Стадии разработки интерфейса (по шагам) наглядно показаны на рисунке 2.

2. Заблокируйте действие кнопки «Развернуть», находящейся в строке заголовка формы «Денежный калькулятор». Для этого в окне Object Inspector придайте свойству BorderStyle значение bsSingle. Кроме этого, найдите в групповом свойстве +BorderIcons вложенное в него свойство biMaximize и установите его в положение False, т.е. отключите это свойство.

3. Изобразите кювету (углубление) для размещения в ней рисунка. Для этого на Палитре компонентов найдите на вкладке Additional компонент Bevel и щёлкните по нему мышью. Растяните на Форме кювету нужных размеров.

4. Вставьте в кювету рисунок. Для этого на Палитре компонентов найдите компонент Image (на вкладке Additional) и щёлкните по нему мышью. Растяните внутри кюветы прямоугольную область, отводимую под рисунок. В свойстве Picture задайте путь и имя файла с рисунком: \\pdc\public_all\ОАП\PICTURES\money.bmp. В окне Object Inspector придайте свойству AutoSize значение True, тем самым обеспечив автоматическую подгонку размеров компонента и изображения.

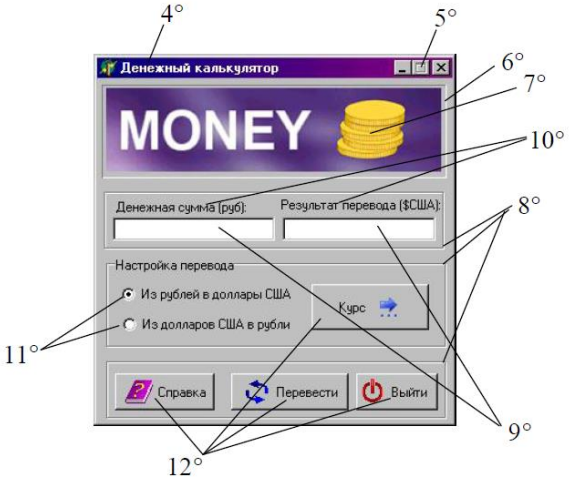


Рисунок 2 – Стадии разработки интерфейса

Примечание: при необходимости сделайте ручную подгонку размеров остальных объектов Формы, ориентируясь при этом на рисунок 1.

5. С помощью компонента GroupBox1 на вкладке Standard изобразите три декоративные рамки (см. рисунок 2, шаг 8°), каждая из которых будет охватывать свою группу объектов. Для верхней и нижней очистите свойство Caption, а для средней задайте этому свойству значение «Настройка перевода».

6. В соответствии с рисунком 2, используя компонент Edit, разместите на Форме «Денежный калькулятор» два текстовых поля: Edit1 и Edit2. Чтобы заблокировать возможность ввода с клавиатуры информации в текстовое поле Edit2 (используемое для вывода результата расчета), придайте его свойству ReadOnly значение True.

7. В соответствии с рисунком 2, используя компонент Label, разместите на Форме «Денежный калькулятор» надписи « Денежная сумма (руб): » и « Результат перевода ($США): » с именами Label1 и Label2, соответственно.

8. В соответствии с рисунком 2, используя компонент RadioButton, разместите на Форме «Денежный калькулятор» два переключателя режимов: RadioButton1 и RadioButton2. Для первого из них задайте текст: «Из рублей в доллары США». Для второго – «Из долларов США в рубли». Первый переключатель переведите в положение «Включен». Для этого задайте свойству Checked значение True.

9. Разместите на Форме кнопки управления. Вместо обычных кнопок Button мы будем использовать кнопки с картинками. Для этого на Палитре компонентов найдите компонент BitBtn (он находится на вкладке Additional). С помощью этого компонента создайте четыре кнопки, как показано на рисунке 2. На кнопке BitBtn1 нанесите надпись «Выйти», на кнопке BitBtn2 – «Перевести», на BitBtn3 – «Справка» и, наконец, на BitBtn4 – «Курс». Изобразите на этих кнопках рисунки. Для этого в свойстве Glyph для каждой из кнопок задайте имя её картинки, находящейся по адресу \\pdc\public_all\ОАП\PICTURES . Для BitBtn1 – exit.bmp, для BitBtn2 – replace.bmp, для BitBtn3 – help.bmp, а для BitBtn4 – rate.bmp. Теперь расположите рисунок кнопки BitBtn4 справа от текста надписи. Для этого в свойстве Layout поменяйте параметр GlyphLeft на GlyphRight. В свойстве Spacing для всех кнопок можно задать расстояние в пикселях между картинкой и текстом надписи (по желанию), а свойству Margin этих кнопок попробуйте присвоить значение – 0 (при таком параметре картинка и надпись будут находиться у левого края кнопки). Интерфейс первой Формы завершен. Обратите внимание на то, что на кнопке <Курс> изображена стрелка с многоточием. Это означает, что при щелчке мышью по этой кнопке должно открыться одноимённое диалоговое окно. Опишем его интерфейс во второй Форме. Пусть он будет выглядеть так, как на рисунке 3.

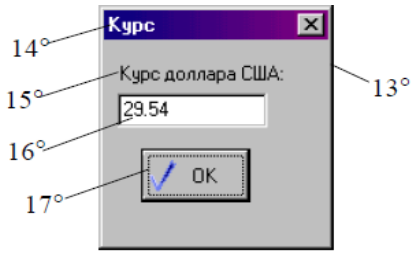


Рисунок 3 – Интерфейс окна Курс

10. Для создания второй Формы (Form2) задайте команду –File/ New/ Form-Delphi for Win32. Сделайте это окно диалоговым. Для этого свойству BorderStyle присвойте значение bsDialog.

11. Создайте с помощью свойства Caption заголовок у Формы 2 – «Курс». Имя формы (Name) оставьте Form2.

12. В соответствии с рисунком 3, используя компонент Label, разместите на Форме «Курс» надпись «Курс доллара США:» с именем Label1.

13. В соответствии с рисунком 3, используя компонент Edit, разместите на Форме «Курс» текстовое поле Edit1. Оно понадобится нам для ввода курса доллара. Введите начальный курс.

14. В соответствии с рисунком 3, используя компонент BitBtn, разместите на Форме «Курс» кнопку BitBtn1. Привяжите к этой кнопке рисунок \\pdc\public_all\ОАП\PICTURES\ok.bmp. Интерфейс второй Формы завершен. Нам потребуется создать ещё и третью Форму для формирования справочного окна «О программе». Образец на рис.4.

Рисунок 4 – Справочное окно

15. Создайте третью Форму (Form3) такой же, как и Form2, т.е. в виде диалогового окна.

16. С помощью компонента Image разместите на Форме картинку titlhelp.bmp.

17. Добавьте на Форму компонент Memo с текстом справки. Дополните объект Memo1 вертикальной полосой прокрутки. Для этого в окне Инспектора объектов свойству ScrollBars установите значение ssVertical.

18. Заполните Memo1 текстовой информацией. Для этого сделайте двойной щелчок по параметру свойства Lines или одиночный щелчок по знаку «…» и в открывшемся окне можно ввести текст вручную с клавиатуры или, удобнее, загрузить

готовый текст из файла. Для этого щелкните правой кнопкой мыши по текстовому полю и в открывшемся контекстном меню выберите команду Load… После этого укажите на файл: help.txt.

19. Наконец, создайте кнопку для закрытия окна – BitBtn1 с рисунком находящимся в папке, указанной преподавателем: close.bmp.

20. Свяжите созданные модули Unit2 и Unit3 с модулем Unit1. Для этого, поочередно выделив Form2 и Form3, выберите команду «File\Use Unit…» в Главном меню Delphi. В открывшемся окне свяжите Unit2 с Unit1 и Unit3 с Unit1. Сохраните результаты проектирования (Unit2.pas и Unit3.pas).

Программирование событий

21. Сделайте активной Форму 1 (Form1) и по [F11] войдите в Unit1.pas. Объявите здесь нужные нам переменные, включите в раздел USES модули Unit2 и Unit3:

```
      VAR
    Form1:    TForm1;    {Переменная    формы,    автоматически
формируемая Delphi}
    K :Real;                  {Курс доллара США}
    Kstr :String;            {Строковое значение курса доллара}
    S :Real;                 {Численное значение денежной суммы}
    Sstr :String;            {Строковое значение денежной суммы}
    Error :Integer;         {Флаг ошибок}

    IMPLEMENTATION
    USES
       Unit2, Unit3;
    {$R *.dfm}
```

22. Опишите событие, связанное с щелчком по кнопке <Выйти>, находящейся на Form1.

```
    PROCEDURE TForm1.BitBtn1Click(Sender: TObject);
    Begin
          Form1.Close;     {Закрытие окна программы}
    End;
```

23. Опишите события, связанные со щелчком мыши по переключателям RadioButton1 или RadioButton2:

```
    PROCEDURE TForm1.RadioButton1Click(Sender: TObject);
    Begin
      Form1.Edit1.Text:='';  {очистить поле ввода исходной
    информации}
      Form1.Edit2.Text:=''; {очистить поле вывода результата
    расчета}
      If Form1.RadioButton1.Checked=True then
      Begin
        Form1.Label1.Caption:='Денежная сумма (руб):';
        Form1.Label2.Caption:='Результат перевода ($США):';
      end else
      begin
          Form1.Label1.Caption:='Денежная сумма ($США):';
          Form1.Label2.Caption:='Результат перевода (руб):';
      end;
    End;
```

```
PROCEDURE TForm1.RadioButton2Click(Sender: TObject);
Begin
     Form1.Edit1.Text:='';{очистить поле ввода исходной
информации}
     Form1.Edit2.Text:='';{очистить поле вывода результата
расчета}
     If Form1.RadioButton2.Checked=False then
     Begin
          Form1.Label1.Caption:='Денежная сумма (руб):';
          Form1.Label2.Caption:='Результат перевода($США):';
     end else
     begin
          Form1.Label1.Caption:='Денежная сумма ($США):';
          Form1.Label2.Caption:='Результат перевода (руб):';
     end;
End;
```

24. Опишите событие, связанное со щелчком мыши по кнопке <Справка>. Эта кнопка, находящаяся на Form1, имеет имя BitBtn3.

```
PROCEDURE TForm1.BitBtn3Click(Sender: TObject);
Begin
     Form3.Show; {Открытие окна справочной информации}
End;
```

25. Опишите, какие действия будет совершать программа в момент ее запуска. Для этого выделите Form1 и в Инспекторе объектов на вкладке Events сделайте двойной щелчок по пункту OnActivate. При этом откроется Окно Кода Программы Unit1.pas. Введите:

```
PROCEDURE TForm1.FormActivate(Sender: TObject);
Begin
     IF Form1.RadioButton1.Checked=True then
     begin
          Form1.Label1.Caption:='Денежная сумма (руб):';
          Form1.Label2.Caption:='Результат перевода($США):';
     end;
     If Form1.RadioButton2.Checked=True Then
     begin
          Form1.Label1.Caption:='Денежная сумма ($США):';
          Form1.Label2.Caption:='Результат перевода (руб):';
     end;
End;
```

26. Опишите событие, связанное со щелчком по самой главной кнопке – <Перевести> (BitBtn2). Именно с помощью этой кнопки выполняется решение поставленной задачи:

```
PROCEDURE TForm1.BitBtn2Click(Sender: TObject);
Begin
    {Если выбран первый переключатель}
     IF form1.RadioButton1.Checked=True then
     begin
          Kstr:=Form2.Edit1.Text;
```

```
                {Перевод в численное значение курса}
               Val(Kstr,K,Error);
               Sstr:=Form1.Edit1.Text;

                {Перевод в численное значение суммы}
               Val(Sstr,S,Error);
               S:=S/K;

               {Перевод в строковое значение результата}
               Str(S:5:2,Sstr);
               Form1.Edit2.Text:=Sstr;
          end;

          {Если выбран второй переключатель}
          If Form1.RadioButton2.Checked=True Then
          begin
               Kstr:=Form2.Edit1.Text;
               Val(Kstr,K,Error);
               Sstr:=Form1.Edit1.Text;
               Val(Sstr,S,Error);
               S:=S*K;
               Str(S:5:2,Sstr);
               Form1.Edit2.Text:=Sstr;
          end;

          {Если в процессе перевода возникали ошибки}
          If Error<>0 then
               begin
                    Form1.Edit2.Text:='Ошибка ввода!';
               end;
     End;
```

27. Опишите событие, связанное со щелчком по кнопке <Закрыть> (BitBtn1) на Форме 3:

```
PROCEDURE TForm3.BitBtn1Click(Sender: TObject);
Begin
     Form3.Close;
End;
```

28. Опишите событие, связанное со щелчком по кнопке <Курс> (BitBtn4) на Форме 1:

```
PROCEDURE TForm1.BitBtn4Click(Sender: TObject);
Begin
     Form2.show;
End;
```

29. Опишите событие, связанное со щелчком по кнопке <OK> (BitBtn1) на Форме 2:

```
PROCEDURE TForm2.BitBtn1Click(Sender: TObject);
Begin
     Form2.Close;
End;
```

Примечание: Для переключения между формами используйте команду Главного меню Delphi: **View – Forms…**

Сохраните проект и запустите на исполнение.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. В приложении Калькулятор добавить кнопку «Очистить».
2. Создать приложение, позволяющее вычислять площадь геометрической фигуры: квадрат, прямоугольник, прямоугольный треугольник. Выбор фигуры осуществить с помощью компонента TRadioGroup.

ВОПРОСЫ ТЕКУЩЕГО КОНТРОЛЯ

1. Как заблокировать кнопку «Развернуть» в строке заголовка приложения?
2. Как изобразить прямоугольную кювету в окне Формы?
3. Как вставить рисунок в окно Формы?
4. Можно ли доработать рисунок, предназначенный для размещения на окне Формы, с помощью какого-нибудь графического редактора?
5. Можно ли украсить окно разрабатываемого приложения какой-нибудь фотографией?
6. Как добиться того, чтобы вставляемый в окно Формы рисунок совпал по своим геометрическим размерам с полем, которое вы для него подготовили?
7. Как изобразить декоративную рамку в окне формы?
8. Как изобразить в окне Формы поле для ввода текстовой информации?
9. Как изобразить надпись в окне Формы?
10. Может ли надпись состоять из нескольких строк?
11. Как изобразить в окне Формы переключатели режимов?
12. Как изобразить в окне Формы кнопку с картинкой и надписью?
13. Возможно ли такое, чтобы приложение создавалось из нескольких Форм?
14. Чем принципиально отличается диалоговое окно от окна приложения?
15. С помощью какого компонента можно разместить на поле Формы многострочный текст?
16. Как вставить в окно с многострочным текстом полосу прокрутки этого текста?
17. Можно ли сделать так, чтобы окно разрабатываемого приложения закрывалось не только с помощью стандартной кнопки «Закрыть», находящейся в строке заголовка этого приложения, или «горячей» клавиши [Alt-F4], но и с помощью «самодельной» кнопки?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8

Общие свойства компонентов

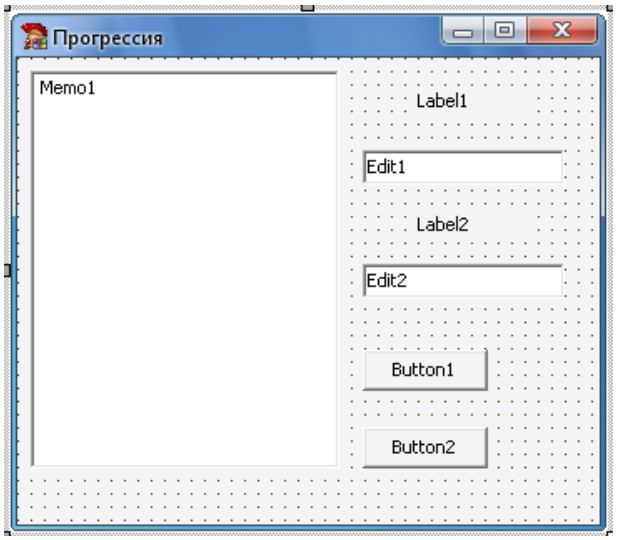
Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонента - Memo; компиляция, сборка и выполнение программ.

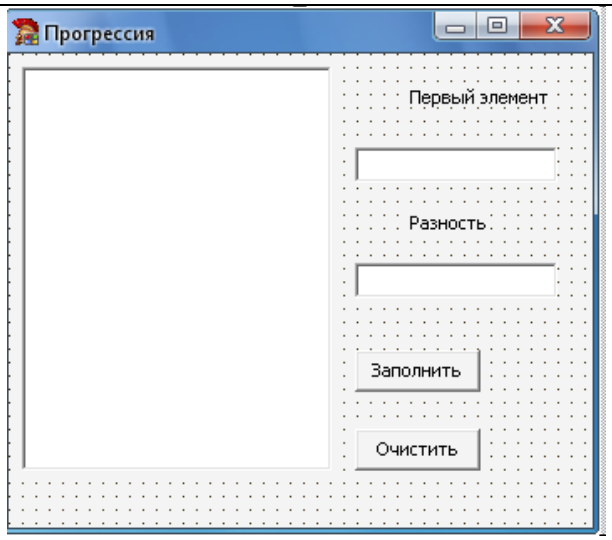
Этапы выполнения работы:

1. Все студенты на занятии выполняют задание указанное ниже.

2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 Создать оконное приложение, вычисляющее первые 20 элементов арифметической прогрессии, для которой заданы первый элемент и разность.

1. Создайте папку для файлов проекта «Прогрессия».	А) File/New/VCL Forms Application Б) File/Save Project As В окне Save Unit1 As создаем папку «Прогрессия», открываем ее и задаем имя файла «Main» В) Сохранить Г) Задаем имя файла проекта «Progressia» Д) Сохранить
2. Студенты самостоятельно переименовывают форму (Form1 ➔ «Прогрессия»), размещают на форме компоненты (Memo1, Edit1, Edit2, Label1, Label2, Button1, Button2) как показано на рисунке:  	
3. Расположенные объекты выровнять по горизонтали.	При нажатой клавише SHIFT выделяем компоненты и в контекстном меню задаем команду: Position/Align/Horizontal/Centers
4. Добавить недостающие компоненты на форму, чтобы она приняла следующий вид:	

<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td>5. Удалить текст Edit1, Edit2, Memo1 в соответствующих компонентах.</td><td>ObjectInspector → Properties → Text, удаляем текст.</td></tr>
<tr><td colspan="2">6. Самостоятельно выровнять компоненты Label1, Label2 по горизонтали и зафиксировать компоненты на форме</td></tr>
<tr><td>7. Запрограммировать кнопку «Заполнить»</td><td>Объявление глобальных переменных:
Var
Form1 : Tform1;
a, d, an, i: integer;

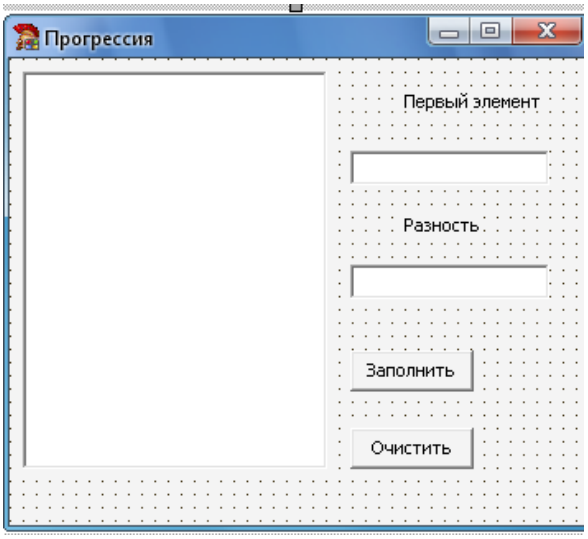
Procedure TForm1.Button1Click(Sender: TObect);
Begin
a:=strtoint(edit1.Text);
d:=strtoint(edit2.Text);
memo1.Lines.Add(edit1.Text);
an:=a;
for i:=2 to 20 do
begin
an:=an+d;
memo1.Lines.Add(inttostr(an))
end;
end;</td></tr>
<tr><td>8. Запрограммировать кнопку «Очистить»</td><td>Produre TForm1.Button2Click(Sender: TObject);
begin
memo1.Lines.Clear;
end.</td></tr>
<tr><td>9. Откомпилируйте и запустите приложение на исполнение</td><td>а) Тестируем, указывая два операнда.
б) Не задаем значение операндов.
<u>Замечание:</u> В случае б) выводится окно сообщения о некорректности значения операнда</td></tr>
<tr><td>10. Отредактировать текст модуля таким</td><td>Помещаем перед оператором присваивания</td></tr>
</table>

образом, чтобы перед выполнением вычислений выполнялась проверка, заданы ли значения операндов. Если значения не заданы, то следует вывести сообщение об этом в отдельном окне.	A:=StrToInt(Edit1.Text); Строку If (Edit1.Text <> ‘’) and (Edit2.Text <> ‘’) Then Begin … А перед последним оператором END прописать следующее: Else ShowMessage (‘НЕ ЗАДАНЫ ЗНАЧЕНИЯ’);
11. Сохранить, откомпилировать и запустить приложение на исполнение. Проверить работу для случая когда: а) не заданы значения; б) введены, не цифры, а другие символы.	
12. Создать обработчик события, запрещающего ввод любых символов, крое цифр от 0 до 9 и знаков + и -.	ObjectInspector → Edit1 → Events → OnKeyPress → двойной щелчок левой кнопкой мыши на пустом поле списка. После этого окно редактора 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. **Замечание:** Различаются символы в верхнем и нижнем регистрах и символы кириллицы и латиницы. Клавиши SHIFT, CTRL, ALT, функциональные клавиши не вызывают этого события, поэтому нажатие комбинации, например, SHIFT+A соответствует нажатию на символ А. Чтобы распознать комбинации применяют другие обработчики событий: OnKeyDown, OnKeyUp. В тело процедуры TForm1.Edit1KeyPress Вставляем следующий оператор: If not (Key in ['0'.. '9', '+', '-',#08]) then Key:=#0; Аналогично создаем процедуру обработчика события нажатия на клавишу в окне Edit2. #08 – соответствует BS.

13. Модифицировать обработчик события так, чтобы знаки + и – воспринимались только вначале числа	if (Key='+') and (length(inttostr(a))<>0) then
	Key:=#0;
	if (Key='-') and ((inttostr(a))<>0) then
	Key:=#0;
	if (Key='+') and (length(inttostr(d))<>0) then
	Key:=#0;

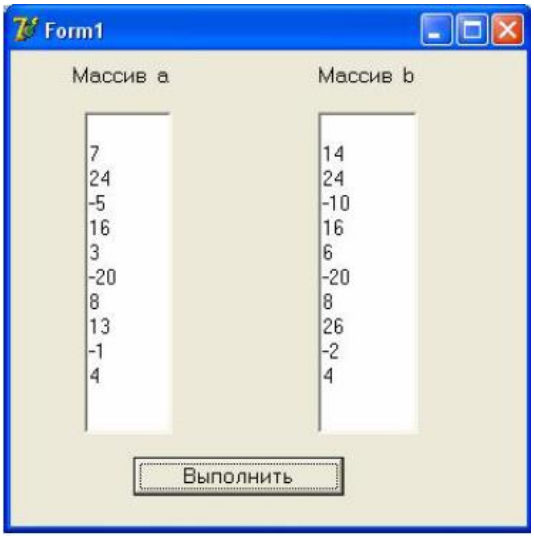
	if (Key='-') and ((inttostr(d))<>0) then Key:=#0;
14. Сохранить, откомпилировать и запустить приложение на исполнение.	Вводим в качестве операндов не цифры, а другие символы.
Примечание: При обработке таких исключительных ситуаций Delphi использует специальные объекты – исключения, характеризующие возникшую в программе исключительную ситуацию.	

Приложение имеет вид:

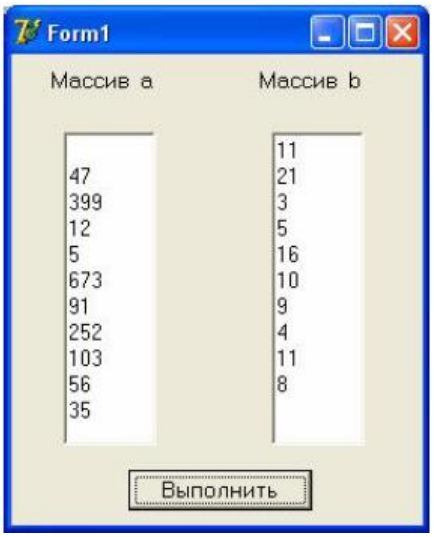


САМОСТОЯТЕЛЬНАЯ РАБОТА

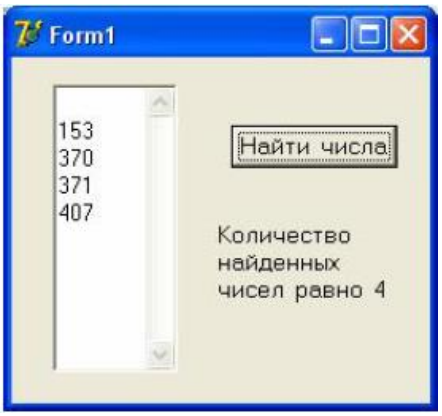
1. Заполнить одномерный целочисленный массив a числами, введенными с клавиатуры в поле Memo1. Получить новый массив b, удвоив нечетные элементы массива a, четные оставить без изменения. Полученный массив b отобразить в поле Memo2. Окно работающего приложения:



2. Заполнить одномерный целочисленный массив a числами, введенными с клавиатуры в поле Memo1. Получить новый массив b, каждый элемент которого равен сумме цифр соответствующего элемента массива a. Полученный массив b отобразить в поле Memo2. Окно работающего приложения:

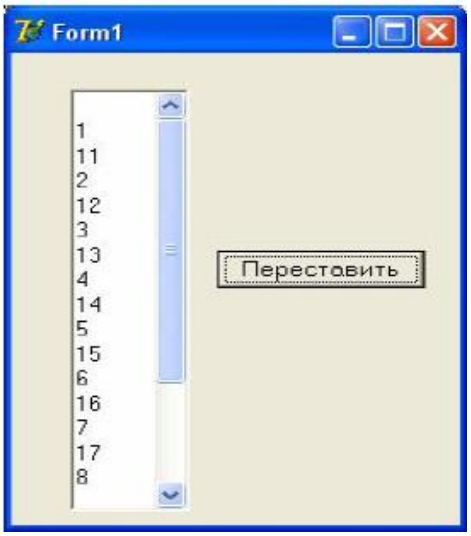


3. Найти все трёхзначные числа, каждое из которых удовлетворяет условию: сумма кубов равняется самому числу. Найденные числа отобразить в поле редактора Memo, подсчитать их количество. Окно работающего приложения:



4. Дан одномерный целочисленный массив из 20 элементов. Ввести его элементы с клавиатуры в поле Memo и удалить последний максимальный элемент массива. Окно работающего приложения:

5. Дан одномерный целочисленный массив a из 20 элементов. Ввести его элементы с клавиатуры в поле Memo, а затем, используя методы класса TStrings, переставить элементы массива a так, чтобы они расположились в следующем порядке a1, a11, a2, a12, . . . ,a10, a20. Окно работающего приложения:



ВОПРОСЫ ТЕКУЩЕГО КОНТРОЛЯ

1. Назначение компонента Memo.
2. Назвать пять свойств данного компонента.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №9

Общие свойства компонентов

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонента - ListBox; компиляция, сборка и выполнение программ.

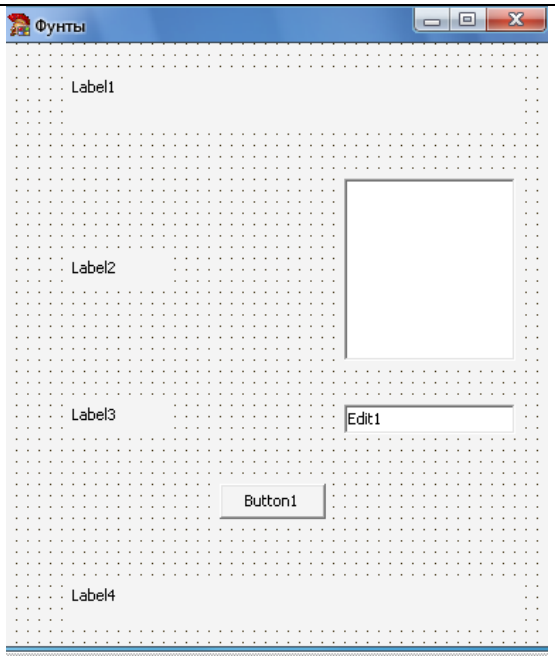
Этапы выполнения работы:

1. Все студенты на занятии выполняют задания указанные ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

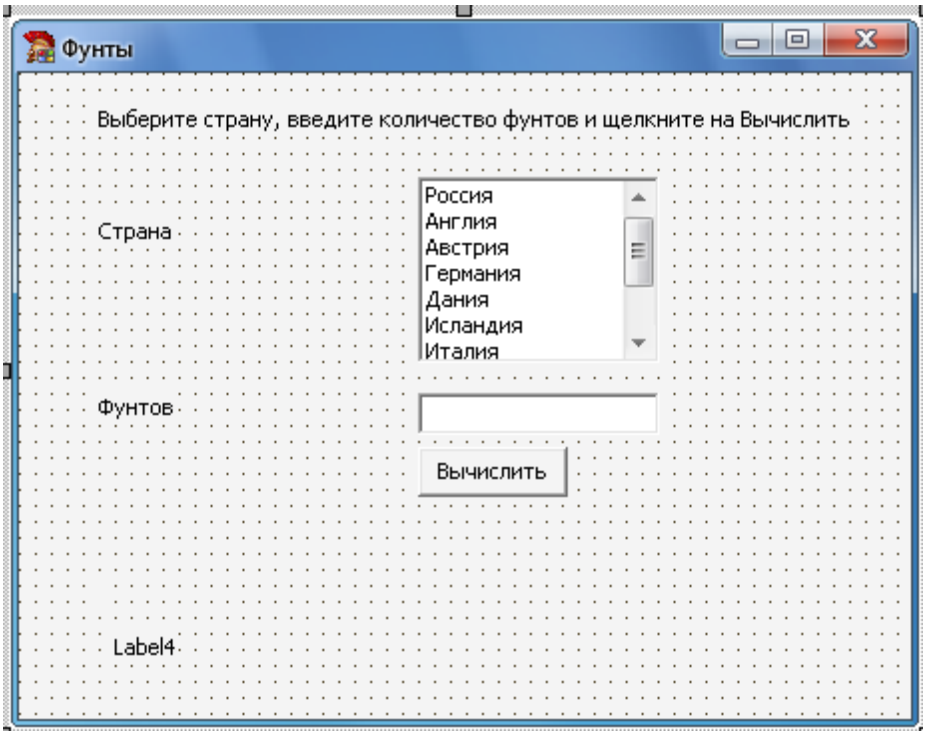
Задание 1 **Написать программу, которая пересчитывает вес из фунтов в килограммы с учетом того, что в разных странах фунт «весит» по-разному.**

Россия	**0,4059**
Англия	**0,453592**
Австралия	**0,56001**
Германия	**0,5**
Дания	**0,5**
Исландия	**0,5**
Италия	**0,31762**
Нидерланды	**0,5**

1. Создайте папку для файлов проекта «Фунт».	А) File/New/VCL Forms Application Б) File/Save Project As В окне Save Unit1 As создаем папку «Фунты», открываем ее и задаем имя файла «Main» В) Сохранить Г) Задаем имя файла проекта «Funti» Д) Сохранить
2. Студенты самостоятельно переименовывают форму (Form1 ➔ «Фунты»), размещают на форме компоненты (Edit1, Label1, Label2, Label3, Label4, Button1, ListBox1) как показано на рисунке:	



3. Добавить недостающие компоненты на форму, чтобы она приняла следующий вид:



4. Удалить текст Edit1 в соответствующих компонентах.	ObjectInspector → Properties → Text, удаляем текст.
5. Заполнить ListBox	ObjectInspector → Items → щелкнуть на кнопке редактора на которой изображены три точки. В появившемся диалоговом окне StringListEditor следует набрать список, поместив каждый элемент списка на отдельной строке. Ввод очередного элемента списка

	должен заканчиваться нажатием клавиши Enter.

6. Запрограммировать кнопку «Вычислить»	Procedure TForm1.Button1Click(Sender: TObect);
	Var funt, kg, k: real;
	Begin
	Case ListBox1.ItemIndex of
	0: k:=0.4059;
	1: k:=0.453592;
	2: k:=0.056001;
	3..5,7: k:=0.5;
	6: k:=0.31762
	end;
	funt:=StrToFloat(Edit1.Text);
	kg:=k*funt;
	Label4.Caption:= Edit1.Text + ‘ фунт(а/ов) – это ’ + FloatToStrF(kg, ffFixed,6,3) + ‘ кг’;
	end;

7. Откомпилируйте и запустите приложение на исполнение	а) Тестируем, указывая операнд. б) Не задаем значение операнда. **Замечание:** В случае б) выводится окно сообщения о некорректности значения операнда

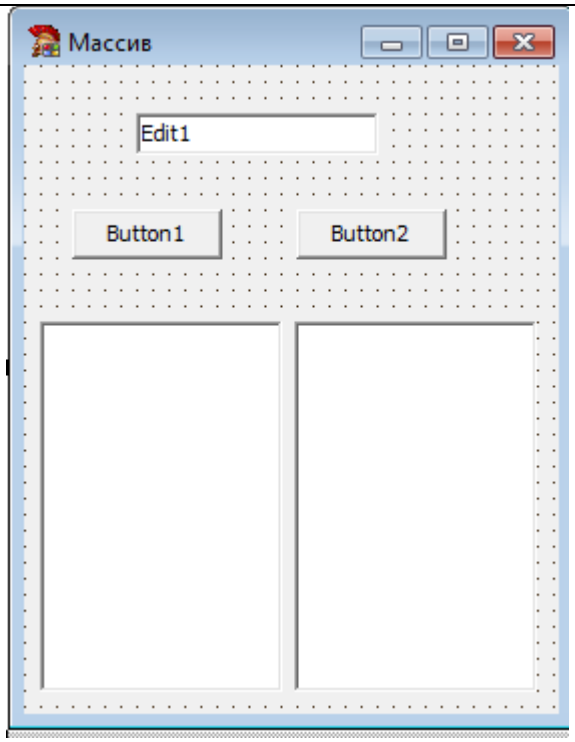
8. Отредактировать текст модуля таким образом, чтобы перед выполнением вычислений выполнялась проверка, заданы ли значения операндов.
Если значения не заданы, то следует вывести сообщение об этом в отдельном окне.
	Помещаем перед оператором присваивания
	funt:=StrToFloat(Edit1.Text);
	Строку
	If (Edit1.Text <> ‘’)
	Then
	Begin …
	А перед последним оператором END прописать следующее:
	Else ShowMessage (‘НЕ ЗАДАНЫ ЗНАЧЕНИЯ’);

9. Сохранить, откомпилировать и запустить приложение на исполнение. Проверить работу для случая когда: а) не заданы значения; б) введены, не цифры, а другие символы.	
10. Создать обработчик события, запрещающего ввод любых символов, крое цифр от 0 до 9 и знаков + и -.	ObjectInspector → Edit1 → Events → OnKeyPress → двойной щелчок левой кнопкой мыши на пустом поле списка. После этого окно редактора 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 тело процедуры TForm1.Edit1KeyPress Вставляем следующий оператор:

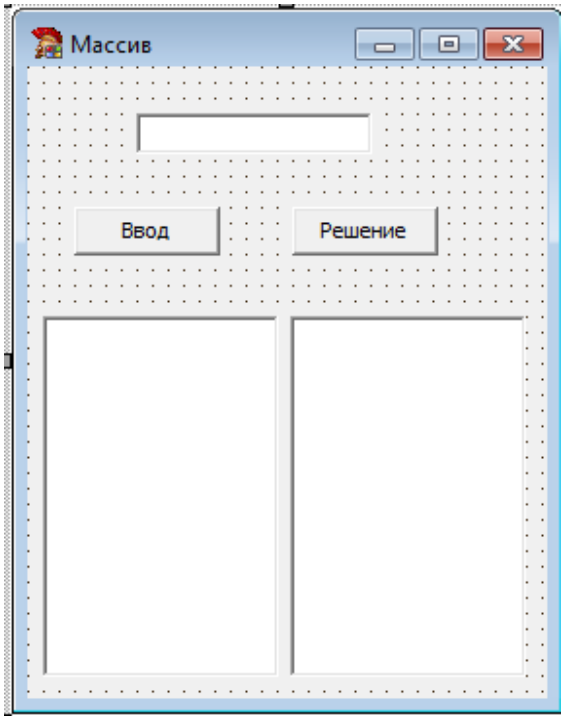
	If not (Key in [‘0’..’9’,’+’,’-’б, #08]) then Key:=#0;
11. Модифицировать обработчик события так, чтобы знаки + и – воспринимались только вначале числа	if (Key='+') and (length(Edit1.Text)<>0) then Key:=chr(0); if (Key='-') and (length(Edit1.Text)<>0) then Key:=chr(0);
12. Сохранить, откомпилировать и запустить приложение на исполнение.	Вводим в качестве операндов не цифры, а другие символы.

Задание 2. **В целочисленном массиве из 10 элементов (все элементы различны) найти максимальный и минимальный элементы и поменять их местами. (Для отображения массивов в форме использовать компонент ListBox).**

1. Создайте папку для файлов проекта «Массив».	А) File/New/VCL Forms Application Б) File/Save Project As В окне Save Unit1 As создаем папку «Массив», открываем ее и задаем имя файла «Main» В) Сохранить Г) Задаем имя файла проекта «Massiv» Д) Сохранить
2. Студенты самостоятельно переименовывают форму (Form1 ➔ «Массив»), размещают на форме компоненты (Edit1, Button1, Button2, ListBox1, ListBox2) как показано на рисунке:	



3. Форма должна принять следующий вид:



4. Для ввода целых чисел используется поле редактирования Edit1. Ввод каждого числа завершается щелчком мыши по кнопке Ввод. Введенный массив отображается в поле ListBox1. После щелчка по кнопке Решение преобразованный массив отображается в поле ListBox2.

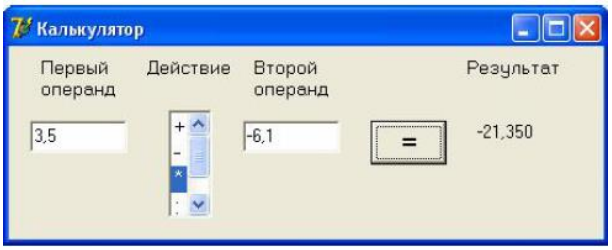
5. Вставим описание массива и используемых переменных.	В разделе описания глобальных переменных пишем: var

<table>
<tr><td></td><td>a: array[1..10] of integer;
i, min, max, imin, imax: integer;</td></tr>
<tr><td>6. Зададим начальное значение индекса массива.</td><td>Щелчок по форме Form1 → в окне Object Inspector откроем вкладку Events → двойной щелчок по OnCreate. В открывшемся редакторе кода пишем:
Procedure TForm1.FormCreate(Sender: TObject);
Begin i:= 0;
End;</td></tr>
<tr><td>7. Запрограммировать кнопку «Ввод»</td><td>Выделим кнопку Ввод → в окне Object Inspector откроем вкладку Events → двойной щелчок по OnClick. В открывшемся редакторе кода пишем:
Procedure TForm1.Button1Click(Sender: TObect);
Begin
ListBox1.Items.Add(Edit1.Text);
i:=i+1;a[i]:= StrToInt(Edit1.Text);
Edit1. SetFocus
End;</td></tr>
<tr><td colspan="2"><u>Примечание:</u> Функция Add формирует список ListBox1, заполняя его числами, вводимыми в поле Edit1. Одновременно этими же числами заполняется массив a. Процедура SetFocus устанавливает фокус ввода в поле Edit1.</td></tr>
<tr><td>8. Запрограммировать кнопку «Решение»</td><td>Выделим кнопку Решение → в окне Object Inspector откроем вкладку Events → двойной щелчок по OnClick.
В открывшемся редакторе кода пишем:
Procedure TForm1.Button2Click(Sender: TObect);
Var k: integer;
Begin
max:=a[1]; imax:=1; min:=a[1]; imin:= 1;
for k:=2 to 10 do
begin
if max< a[k] then begin max:= a[k];
imax:=k end;
if min> a[k] then begin min:= a[k];
imin:=k end;
end;
a[imax]:=min;
a[imin]:=max;
for k:=1 to 10 do
ListBox2.Items.Add(IntToStr(a[k]));
end;</td></tr>
<tr><td colspan="2">9. Сохранить, откомпилировать и запустить приложение на исполнение.
Проверить работу для случая когда:</td></tr>
</table>

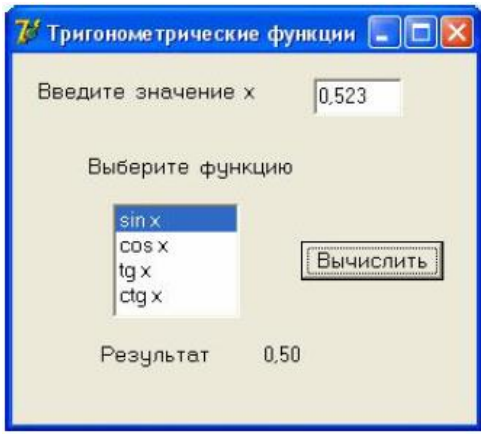
а) не заданы значения; б) введены, не цифры, а другие символы. в) создать обработчик события, запрещающего ввод любых символов, крое цифр от 0 до 9 и знаков + и -.	
10. Сохранить, откомпилировать и запустить приложение на исполнение.	Вводим в качестве операндов не цифры, а другие символы.

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Написать программу, выполняющую работу арифметического калькулятора с четырьмя арифметическими действиями над действительными числами. Окно работающего приложения:

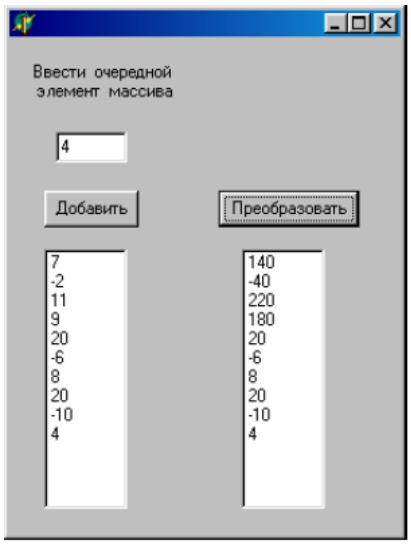


2. Написать программу для вычисления тригонометрической функции (sin, cos, tg, ctg) для данного действительного числа x. Окно работающего приложения:



3. Создать список, состоящий из 10 элементов числами Фибоначчи. Для отображения массива в форме использовать ListBox. Окно работающего приложения:

4. Даны 10 элементов. Все элементы этого списка, предшествующие первому по порядку элементу со значением max(a1, ...,a10), умножить на этот максимальный элемент. Для отображения элементов в форме использовать ListBox1 и ListBox2. Окно работающего приложения:



ВОПРОСЫ ТЕКУЩЕГО КОНТРОЛЯ

1. Назначение компонента ListBox.
2. Назвать пять свойств данного компонента.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №10

Контрольная работа (рубежный контроль)

Типовое задание: разработать интерфейс приложения, создать обработчик кода.
Часть 1

Создать приложение Геометрия. Форма содержит поля для ввода данных. Приложение должно вычислять площади геометрических фигур (любые три). Выбор фигуры осуществляется на форме 1, ввод данных и вывод результата на форме 2 (3 или 4).
Результат должен выводиться красным цветом.

Часть 2

Модифицировать код программы таким образом, чтобы в полях ввода данных можно было внести только численное значение. Если в какое-нибудь поле не были внесены данные, приложение должно выдавать соответствующее сообщение в отдельном окне.

Часть 3

Приложение должно содержать кнопку «Справка», открывающее одноименное окно, содержащее информацию по работе данного приложения, и кнопку «Выход».

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №11

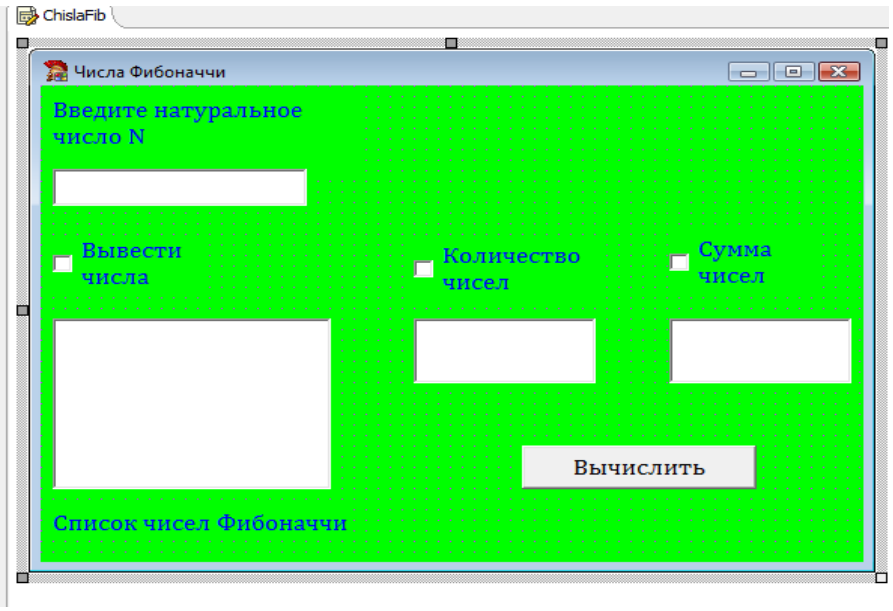
Формы

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонентов CheckBox, ListBox; создание и отладка программы, содержащей операторы циклов; компиляция, сборка и выполнение программ.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание указанное ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 **Разработать приложение, позволяющее по заданному натуральному числу N находить числа Фибоначчи, не превышающие данного N, а также определяло бы количество найденных чисел и их сумму (по требованию). Список чисел вывести в компоненте ListBox.**



1. Самостоятельно создать форму с необходимыми на ней компонентами, изменить свойства объектов в соответствии с рисунком. Организовать запрет ввода в поле для натурального числа N любого текста, кроме целых чисел.

2. Если пользователь введет единицу, то в окне должно появиться сообщение «Чисел Фибоначчи до 1 не существует».

3. Обращение к компоненту CheckBox организовать можно так:

```
if CheckBox1.State=cbChecked then
  for i := 1 to k do
  ListBox1.Items.Add(IntToStr(mas[i]));
```

или так:

```
if CheckBox2.Checked then
Edit2.text:= IntToStr(k);
if CheckBox3.Checked then
Edit3.text:= IntToStr(Sum);
```

4. Сохранить текст модуля под именем chislaF.pas (File/Save AS). Сохранить проект – chislaFib (File/Save As Project).

5. Откомпилируйте программу (Project/Compile). Проверьте работу приложения.

6. Выполните отладку приложения в пошаговом режиме.

Команда View/Debug Windows/Watches открывает окно просмотра значений переменных и выражений.

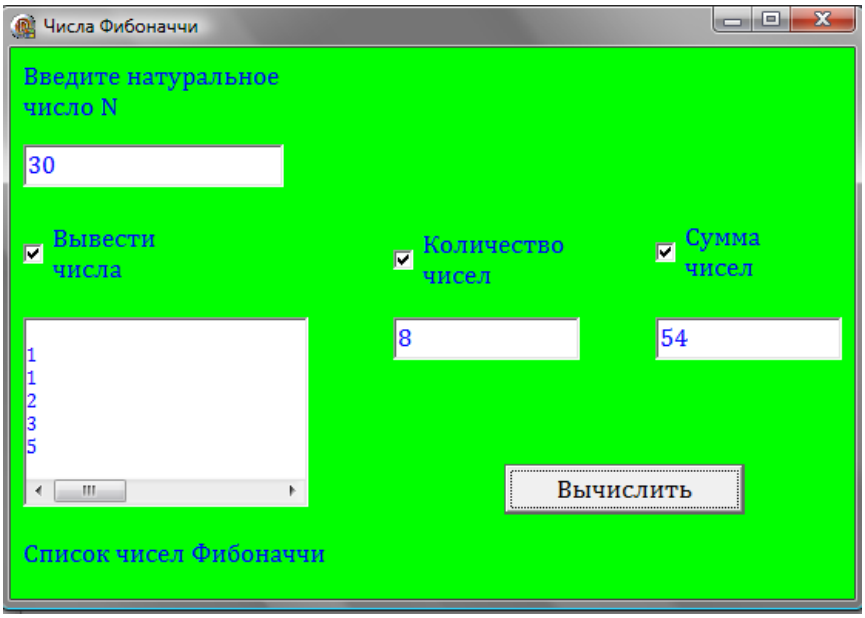
Для включения переменной или выражения в окно просмотра надо выделить в окне редактора кода эту переменную, выражение и нажать CTRL+F5.

В тексте модуля chislaF выделите a, b, b<>0, Edit3 и перетащите их в окно Watches List.

Выполните приложение в пошаговом режиме (Клавиша F7).

Листинг

Разработать приложение, позволяющее по заданному натуральному числу N находить числа Фибоначчи, не превышающие данного N, а также определяло бы количество найденных чисел и их сумму (по требованию). Список чисел вывести в компоненте ListBox.




```
procedure TForm1.Button1Click(Sender: TObject);
var a,i,k,Sum:integer;
    mas:array [1..255] of integer;
begin
 if edit1.text <>'' then
 begin
    a:=StrToInt(edit1.text);
    Sum:=2; k:=2; i:=3;
    mas[1]:=1; mas[2]:=1;
    if a=1 then ShowMessage ('Чисел Фибоначчи до 1 не существует') else
    begin
    repeat
      mas[i]:= mas[i-2]+mas[i-1];
      if mas[i]<a then
        begin
           k:=k+1;
           Sum:=Sum+mas[i];
          inc(i);
      end
```

```
    else break;
   until mas[i-1]>=a;
    if CheckBox1.State=cbChecked then
      for i := 1 to k do
      ListBox1.Items.Add(IntToStr(mas[i]));
      if CheckBox2.Checked then
      Edit2.text:= IntToStr(k);
      if CheckBox3.Checked then
      Edit3.text:= IntToStr(Sum);
    end;
 end
 else ShowMessage ('Вы не ввели число');
end;
```

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Создать приложение для расчета характеристик вектора: сумма двух векторов, произведение двух векторов. Выбор характеристики осуществить с помощью: а) компонента CheckBox; б) компонента ListBox.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Каково назначение визуального компонента CheckBox? Какими способами можно запрограммировать состояние компонента CheckBox?
2. Чем отличаются компоненты CheckBox.и ListBox? Как задать список элементов в объекте ListBox?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №12

Библиотека визуальных компонентов

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений обрабатывающих строки.

Этапы выполнения работы:

4. Все студенты на занятии выполняют задание указанное ниже.
5. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
6. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 Создайте приложение, предлагающее пользователю ввести строку текста, а затем подсчитывающее количество символов в тексте. Количество символов должно сочетаться со словом «символ», «символа», «символов». Например, «21 символ»,

«15 символов», «33 символа». Образец оформления приложения представлен на рисунке 1.

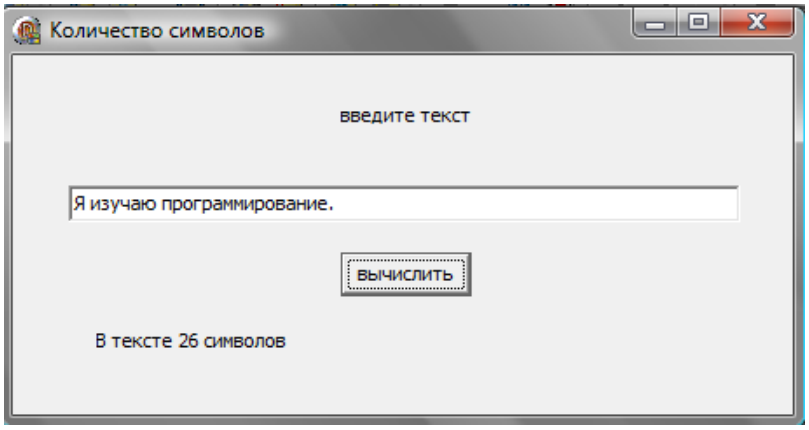


Рисунок 1 - Форма к заданию 1

1. Самостоятельно создать форму с необходимыми на ней компонентами, изменить свойства объектов в соответствии с рисунком.
2. Используя оператор Case, создаем код обработчика события кнопки Button1.

```
Procedure TForm1.Button1Click(Sender: TObject);
var
  n:integer;                {Последняя цифра}
  s:string; {Изменяемое слово “символ”}

Begin
    n:=Length(edit1.text);
    Label2.Caption:='В тексте ';
    If n>20 then n:=n mod 10;            {выделение последней цифры}
        Case n of
                  1 : S:=' символ';
              2..4 : S:=' символа';
           0,5..20 : S:=' символов';
        End;
         {Вывод результата}
     Label2.Caption:=Label2.Caption+IntToStr(Length(Edit1.text))+S;
End;
```

3. Сохранить проект, откомпилировать и запустить программу на выполнение. Протестировать программу при различном количестве символов.

Задание 2 Создайте приложение, предлагающее пользователю ввести строку текста, а затем заменяющее символы в тексте и подчитывающее количество заменённых символов. Вариант замены символов должен определяться по положению соответствующего флажка CheckBox.

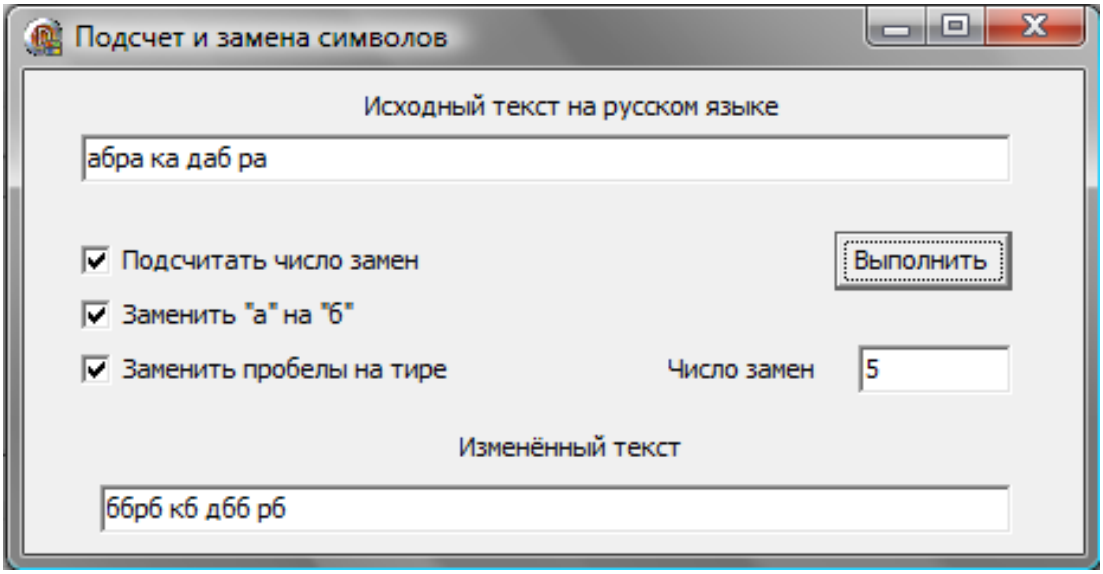


Рисунок 2 - Форма к заданию 2

Алгоритм создания приложения

1. Создание нового проекта	File/New/VCL Form Application
2. Сохранение нового проекта	1) File/Save As а) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Замена и подсчет символов»; б) Открыть эту папку; в) Нажать кнопку сохранить; 2) File/Save Project As а) Задать имя файла simvol и нажать сохранить
3. Самостоятельно расположить на форме нужные компоненты в соответствии с рисунком. Изменить свойства формы и компонентов.	
4. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
5. Фиксация положения компонентов на форме	Edit/Lock Controls (для отмены повторно задать команду)
6. Сохранение изменений	Save All (на стандартной панели инструментов)
7. Компиляция проекта	Project/Compile simvol
8. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
9. Расположение окна приложения	Object Inspector/ Position /poScreenCenter
10. Создание кода – обработчика события	Для того, чтобы приложение выполнило вычисления при щелчке по кнопке Button1 («=») надо написать код обработки этого события. Размещаем необходимые операторы, в разделе описаний вводим следующее: var S:String; {строка текста}

	N:Byte; {количество замен символов}
	
	Основное тело процедуры будет иметь вид:
	
	n:=0;
	Edit2.Text:=’’;
	Edit3.Text:=’’;
	S:=Edit1.Text;
	
	If CheckBox2.Checked then
	While Pos(‘a’, S) > 0 do
	Begin
	N:=n+1;
	S[Pos(‘a’,S)]:= ‘б’;
	End;
	
	If CheckBox3.Checked then
	While Pos(‘ ’, S) > 0 do
	Begin
	N:=n+1;
	S[Pos(‘ ’,S)]:= ‘-’;
	End;
	
	If CheckBox1.Checked then Edit2.Text:= IntToStr(N);
	
	Edit3.Text:=S;
	End;

11. Сохранение изменений	Save All
12. Запустите приложение на исполнение	

Задание 3 Создать приложение, которое позволяет вводить список строк в поле редактора Memo1, а затем при нажатии на кнопку переписывает все строки, содержащие более 5 символов, в поле редактора Memo2.

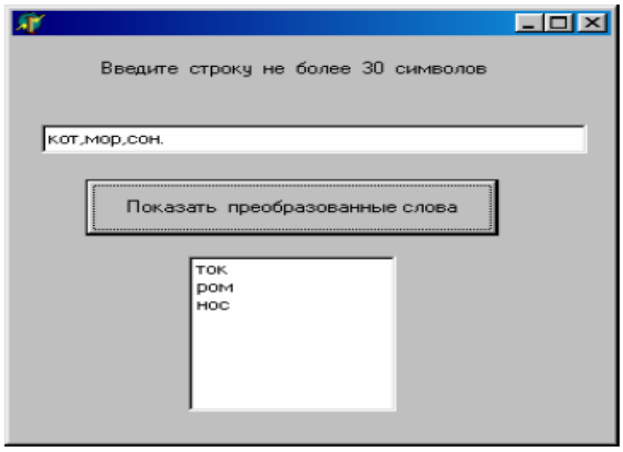
Процедура обработки события щелчка по кнопке будет иметь вид:

```
Var i:integer;
Begin
        For i:=0 to memo1.lines. count-1 do
                If length (memo1.lines. Strings[i])>5
                then
                Memo2.Lines.Add(memo1.lines. Strings[i]))
End;
```

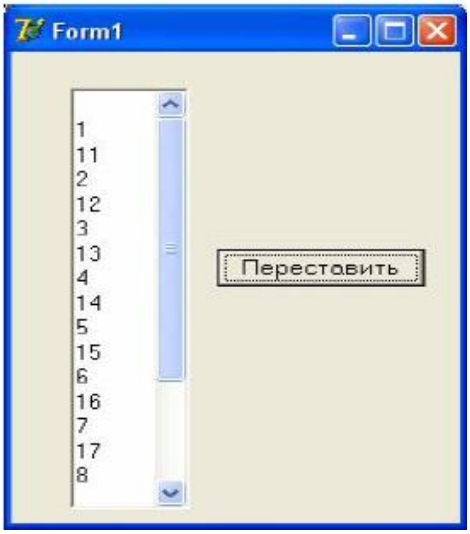
ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Создать приложение, которое определяет, является ли введенная строка перевертышем.

2. Создать приложение, которое позволяет вводить строку и выводит ее в обратном виде, т.е. при вводе «Привет» выведется «тевирП».
3. Создать приложение, которое позволяет вводить список строк в поле редактора Memo, а затем, используя процедуру, переставить строки в обратном порядке.
4. Создать приложение, которое позволяет вводить список строк в поле редактора Memo1, а затем при нажатии на кнопку удаляет все строки, содержащие более 5 символов.
5. Дана строка, содержащая не более 30 символов, являющихся строчными русскими буквами и запятыми. Последний символ – точка. Назовем словом – последовательность букв между ближайшими двумя знаками препинания. Вывести в поле компонента ListBox в столбец все слова, меняя местами в каждом слове первую и последнюю букву.



6. Дан одномерный целочисленный массив a из 20 элементов. Ввести его элементы с клавиатуры в поле Memo, а затем, используя методы класса TStrings, переставить элементы массива a так, чтобы они расположились в следующем порядке a1, a11, a2, a12, . . . ,a10, a20. Окно работающего приложения:



ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Дать характеристику существующим типам строк: String, ShortString, AnsiString, WideString.

2. Дать характеристику процедурам и функциям обработки строк: Concat, Length, Copy, Delete, Insert, Pos, StrToInt, FloatToStr.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №13

Библиотека визуальных компонентов

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений, обрабатывающих линейные массивы.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задания указанные ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 Создайте приложение, предлагающее пользователю задать размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива, а затем по выбору пользователя определяет минимальный и максимальный элементы, сумму всех элементов и количество положительных элементов. Образец оформления приложения представлен на рисунке 1.

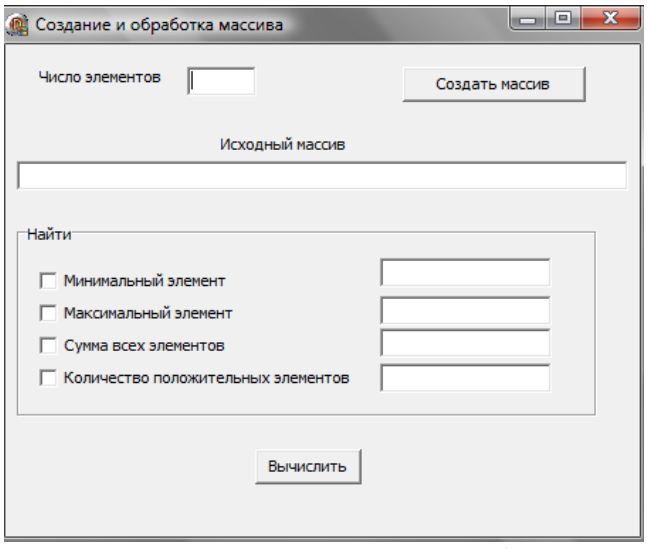


Рисунок 3 - Форма к заданию 1

1. Самостоятельно создайте верхнюю часть формы с необходимыми на ней компонентами, измените свойства объектов в соответствии с рисунком. Вторая часть формы «Найти» организована с помощью панели GroupBox[1]. В панели GroupBox разместите компоненты CheckBox1, CheckBox2, CheckBox3, CheckBox4.

[1] Панель GroupBox – это контейнер с рамкой и надписью, объединяющий группу связанных органов управления, таких как переключатели RadioButton, флажки CheckBox и др. Эта панель используется для выделения на форме группы функционально объединенных компонентов.

Напротив этих компонентов разместите компоненты Edit3-Edit6. Выровняйте компоненты на форме и зафиксируйте их положение (Edit → Lock Controls).

Замечание: если компоненты CheckBox1, CheckBox2, CheckBox3, CheckBox4 окажутся под панелью GroupBox1, то надо выделить эту панель и в контекстном меню задать команду Control → Send To Back (Порядок → На задний план).

2. Сохраните файл проекта и программного модуля.
3. В разделе описаний переменных запишем:

```
Var
  Form: TForm1;
  N, I : integer;
  M : array of integer;              {описание динамического массива[2] целых чисел}
```

где N – это размер массива;
I – порядковый номер элемента массива;
M – динамический массив целых чисел.

4. Создайте обработчик события, запрещающего ввод любых символов, кроме цифр от 0 до 9. Учесть возможность стирания содержимого Edit1 с помощью кнопки BS (08 – код по таблице ASCII).
5. В процедуре обработчика события щелчка мышью на кнопке Button1 описываем создание массива целых чисел.

```
Procedure TForm1.Button1Click(Sender: TObject);
Begin
    Randomize;
    n:=StrToInt(edit1.text);    {число элементов массива}
    SetLength(M,N);             {задать массиву M длину N}
    Edit2.Text:='';
    For i:=0 to n-1 do
       Begin
           M[i]:=round(sin(random(100))*100);
           Edit2.Text:=Edit2.Text+' ' +IntToStr(M[i]);
        End;
End;
```

6. Сохраните проект, откомпилируйте и запустите программу на выполнение.
7. Обработку массива описываем в процедуре обработчика события щелчка мышью на кнопке Button2.

В окне Редактора кода в заготовку процедуры обработчика события procedure TForm1.Button2Click (Sender: TObject); в раздел описания локальных переменных добавим следующее описание:

```
Var max, min, sum, kol:integer;
```

где
Max – максимальный элемент массива;
Min –минимальный элемент массива;
Sum – сумма всех элементов массива;
Kol – количество положительных элементов массива.

---

[2] Динамические массивы отличаются от статических тем, что для них заранее не известна длина - число элементов массива. Прежде чем использовать массив, надо задать его размер с помощью процедуры SetLength.

Текст процедуры обработки массива может быть записан следующим образом:

```
Procedure TForm1.Button2Click(Sender: TObject);
Var max, min, sum, kol:integer;
Begin
        If checkBox1.Checked then min:=m[0];
        Edit3.Text:='';
        If checkBox2.Checked then max:=m[0];
        Edit4.Text:='';
        Sum:=0;
        Edit5.Text:='';
        Kol:=0;
        Edit6.Text:='';
                {обработка массива}
        For i:=0 to n-1 do
            Begin
                If checkBox1.Checked then
                   If min>m[i] then min:=m[i];
                If checkBox2.Checked then
                   If max<m[i] then max:=m[i];
                If checkBox3.Checked then
                   Sum:=sum+m[i];
                If checkBox4.Checked then
                   If m[i]>0 then kol:=kol+1;
             End;
                        {вывод результатов обработки}
                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(kol);

        End;
```

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

Задание 2 Создайте приложение, предлагающее пользователю задать размер линейного массива, заполняет этот массив случайными целыми числами, выводит список элементов массива, а затем по выбору пользователя выполняет сортировку массива (по возрастанию, по убыванию) и выводит новый массив. Образец оформления приложения представлен на рисунке 2.

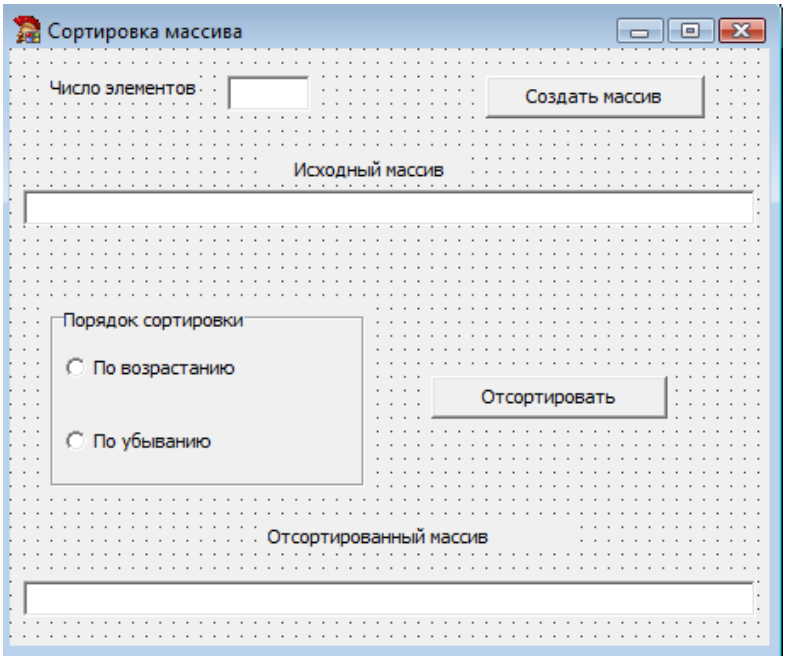


Рисунок 4 - Форма к заданию 2

Алгоритм создания приложения

1. Создание нового проекта	File/New/VCL Form Application
2. Сохранение нового проекта	3) File/Save As г) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Сортировка массива»; д) Открыть эту папку; е) Нажать кнопку сохранить; 4) File/Save Project As б) Задать имя файла massiv и нажать сохранить
3. Самостоятельно расположить на форме нужные компоненты в соответствии с рисунком. Изменить свойства формы и компонентов.	
4. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
5. Сохранение изменений	Save All (на стандартной панели инструментов)
6. Компиляция проекта	Project/Compile massiv
7. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
8. Расположение окна приложения	Object Inspector/ Position /poScreenCenter
9. Создание массива	См. п. 5, задание 1
10. Сохранение изменений	Save All
11. Запустите приложение на исполнение	F9

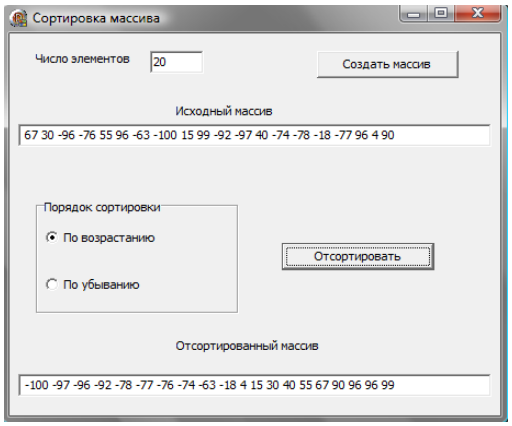
12. Обработка события нажатия кнопки Button2 «Отсортировать»	For i:=0 to n-2 do
	For j:=i+1 to n-1 do
	begin
	If (RadioGroup1.ItemIndex=0) and (m[i]<m[j]) or (RadioGroup1.ItemIndex=1) and (m[i]>m[j]) then
	Begin

	Tmp:=M[i]; M[i]:=M[j]; M[j]:=tmp; End; End;
13. Вывод отсортированного массива	For i:=0 to n-1 do Edit3.Text:=edit3.text+' ' +IntToStr(m[i]);
14. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.	

Листинг к заданию 2

```
unit mas;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Label2: TLabel;
    Edit2: TEdit;
    Button2: TButton;
    RadioGroup1: TRadioGroup;
    Label3: TLabel;
    Edit3: TEdit;
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
   n,i:integer; M:array of integer;
implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin

Begin
      Randomize;
```

```
      n:=StrToInt(edit1.text);    {число элементов массива}
      SetLength(M,N);               {задать массиву M длину N}
      Edit2.Text:='';
      For i:=0 to n-1 do
          Begin
               M[i]:=round(sin(random(100))*100);
               Edit2.Text:=Edit2.Text+' '+IntToStr(M[i]);
           End;
End;
end;


procedure TForm1.Button2Click(Sender: TObject);
Var j,tmp:integer;
Begin
  edit3.Text:='';
  For i:=0 to n-2 do
   For j:=i+1 to n-1 do
      begin
         If   (RadioGroup1.ItemIndex=0)   and   (m[i]>m[j])   or
(RadioGroup1.ItemIndex=1) and (m[i]<m[j]) then
           Begin
            Tmp:=M[i];
             M[i]:=M[j];
             M[j]:=tmp;
            End;
       End;

               {вывод результатов обработки}
     For i:=0 to n-1 do
     Edit3.Text:=edit3.text+' '+IntToStr(m[i]);
end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
    if not (key in ['0'..'9',#08]) then  key:=#0;
end;
end.
```



ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Заполнить одномерный целочисленный массив A числами, введенными с клавиатуры в поле Memo1. Получить новый массив B, возведя в квадрат четные элементы массива A, а нечетные оставить без изменения. Полученный массив B отобразить в поле Memo2.
2. В поле Memo1 ввести числа, произвести их анализ. Распределить отрицательные и положительные числа в поля Memo2 и Memo3.
3. Изменить предыдущее задание так, чтобы числа можно было вводить в поле Edit1.
4. Дан массив A. Вывести сумму положительных элементов и произведение отрицательных. выбор осуществить с помощью RadioGroup.
5. Изменить приложение в № 4 так, чтобы можно было применить компонент CheckBox.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Опишите отличия статических и динамических массивов. Каков порядок описания и применения динамических массивов?
2. Каково назначение визуального компонента GroupBox? Как задать надпись в этом компоненте?
3. Чем отличаются компоненты RadioGroup и GroupBox? Как задать список элементов в объекте RadioGroup?
4. Для чего и каким образом выполняется фиксация положения компонентов на форме?

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №14

Библиотека визуальных компонентов

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонента StringGrid

Этапы выполнения работы:

1. Все студенты на занятии выполняют задания указанные ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Примечание 1.

В следующей таблице описаны некоторые свойства компонента StringGrid.

свойство	**обозначение**
Количество колонок таблицы	ColCount
Количество строк таблицы	RowCount
Соответствующий таблице двумерный сроковый массив(индексы нумерованы от 0)	Cells
Количество зафиксированных колонок таблицы	FixedColls
Количество зафиксированных строк таблицы	FixedRows
Признак допустимости редактирования содержимого ячеек таблицы	Options.goEditing
Ширина колонок таблицы	DefaultColwidth
Высота строк таблицы	DefaultRowHeight
Толщина линий таблицы	GridLineWidth

Задание 1. Создайте приложение с компонентом StringGrid, обеспечивающее пользователю возможность ввода значений элементов двумерного массива в ячейки и выполняющее вычисление суммы элементов массива.

1. Создайте форму	File/New/VCL Form Application
2. Самостоятельно расположить на форме нужные компоненты. Изменить свойства формы и компонентов.	
3. Сохраните файл проекта и программного модуля.	5) File/Save As ж) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Обработка двумерного массива»; з) Открыть эту папку; и) Нажать кнопку сохранить; 6) File/Save Project As Задать имя файла Main и нажать сохранить

4. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
5. Фиксация положения компонентов на форме	Edit/Lock Controls (для отмены повторно задать команду)
6. Сохранение изменений	Save All (на стандартной панели инструментов)
7. Компиляция проекта	Project/Compile simvol
8. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
9. Расположение окна приложения	Object Inspector/ Position /poScreenCenter
10. На странице свойств Инспектора объектов разверните список свойства StringGridl.Options и задайте для его подсвойств: goFixedVertLine, goFixedHorzLine, goVertLine, goHorzLine, goRangeSelect, goDrawFocusSeLected и goEditing значение True.	

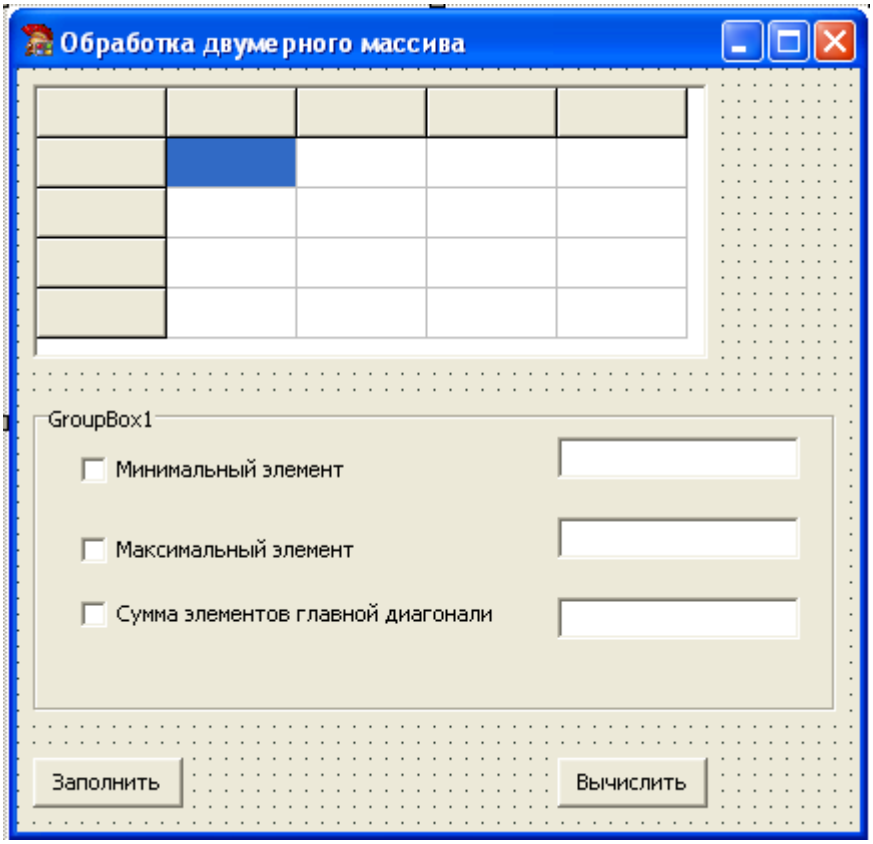
11. Выберите в окне Инспектора объектов объект StringGridl и на странице События произведите двойной щелчок на пустом поле списка в событии OnEnter, наступающем в момент получения элементом фокуса. После этого в окне Редактора кода будет сгенерирована заготовка процедуры обработчика события procedure TForml.FormCreate(Sender: TObject);	Так как эта процедура сводится к выводу в окне приложения таблицы StringGrid1 подписи номеров строк и столбцов, то текст этой процедуры можно записать следующим образом:
	
	procedure TForml.FormCreate(Sender: TObject) var
	{номера столбцов и строк}
	{вывести номера строк в 0 столбце и столбцов в 0 строке}
	{столбец 0}
	I.J: Integer;
	begin
	with StringGridl do
	begin
	i:=0; {строка 0}
	for J:=1 to RowCount-1 do
	Cells[I,J]:=IntToStr(J);
	J:=0;
	for I:=1 to ColCount-1 do
	Cells[I.J]:=IntToStr(I);
	end;
	end;

12. Выберите в окне Инспектора объектов объект Buttonl и на странице События произведите двойной щелчок на пустом поле списка в событии OnClick, наступающем в момент получения элементом фокуса.	После этого в окне Редактора кода будет сгенерирована заготовка процедуры обработчика события procedure TForml.ButtonlClick(Sender: TObject);, в которую следует добавить следующие операторы:
	
	procedure TForml.ButtonlClick(Sender: TObject);
	{подсчет суммы элементов}
	var
	I.J.Sum : integer;
	begin with StringGridl do
	begin
	Sum:=0;

	{обнулить сумму перед суммированием элементов}
	Editl.Text:='';
	for J:=l to RowCount - 1 do for I := 1 to ColCount - 1 do if Cells[J,I]<>'' then
	{если ячейка заполнена, то суммировать ее значение}
	Sum: =Sum+StrToInt (Cells [J, I]);
	Editl.Text:=IntToStr(Sum);
	{вывод результата суммирования}
	end;
	end;

13. Сохраните файлы модуля и проекта, откомпилируйте и запустите приложение.	

Задание 2 Создайте приложение, которое выводит двумерный массив случайных целых чисел в объекте StringGrid и определяет минимальный и максимальный элементы, а также сумму элементов массива, расположенных на главной диагонали.



1. Сохраните файл проекта и программного модуля	1) File/Save As к) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Обработка двумерного массива»; л) Открыть эту папку;

	м) Нажать кнопку сохранить; 2) File/Save Project As Задать имя файла mas2 и нажать сохранить
Примечание2. Компонент StringGrid представляет собой таблицу, содержащую строки. Таблица может иметь полосы прокрутки, причем заданное число первых строк и столбцов может быть фиксированным и не подвергаться прокрутке. Таким образом, можно задать заголовки столбцов и строк, постоянно присутствующие в окне компонента. Каждой ячейке таблицы может быть поставлен в соответствие некоторый объект	
2. самостоятельно расположить на форме нужные компоненты в соответствии с рисунком. Изменить свойства формы и компонентов.	
3. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
4. Сохранение изменений.	Save All (на стандартной панели инструментов)
5. Компиляция проекта.	Project/Compile mas2
6. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
7. Расположение окна приложения	Object Inspector/ Position /poScreenCenter
8. Прописать в глобальных переменных	Var Form1: TForm1; i,j:integer;

10. Создание кода – 1-го обработчика события	Выделить Button1 => Object Inspector => Events => OnClick => После этого отредактировать заготовку процедуры обработчика этого события следующим образом:
	
	procedure TForml.ButtonlClick(Sender: TObject);
	{заполнение массива}
	begin Randomize;
	with StringGridl do
	{вывести номера строк в 0-м столбце и столбцов в 0-й строке}
	begin
	i:=0;
	{столбец 0}
	for J:= 1 to RowCount - 1 do
	{вывести номера строк}
	Cells[I,J] := IntToStr(J);
	J:=0;
	{строка 0}
	for I:= 1 to ColCount - 1 do
	{вывести номера столбцов}
	Cells[I,J]:= IntToStr(I);
	end;
	with StringGridl do
	{вывести в таблице элементы двумерного массива}
	for I := 1 to ColCount - 1 do
	for J:= 1 to RowCount - 1 do
	begin

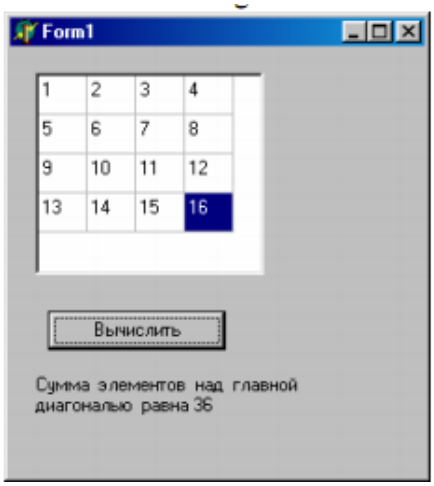
	Cells[I,J] := IntToStr(Round(Sin(Random(100))*100)); end; end;

11. Создание кода – 2-го обработчика события	Выделить Button2 => Object Inspector => Events => OnClick => После этого отредактировать заготовку процедуры обработчика этого события следующим образом:
	procedure TForml.Button2Cl1ck(Sender: TObject);
	{обработка массива}
	var
	Min, Max, Sum:Integer;
	{локальные переменные - результаты обработки массива}
	begin
	if Checkboxl.Checked then
	{определение Min-элемента}
	with StringGridl do
	begin
	Min:=StrToInt(Cells[l,l]);
	{пусть - это Min-элемент}
	for I := 1 to Col Count - 1 do
	for J:= 1 to RowCount - 1 do
	if StrToInt(Cells[I,J])<Min then
	Min:=StrToInt(Cells[,.J]);
	Editl.Text:=IntToStr(Min);
	end
	else Editl.Text:='';
	if Checkbox2.Checked then
	{определение Max-элемента}
	with StringGridl do
	begin
	Max:=StrToInt(Cells[l,l]);
	{пусть - это Max-элемент}
	for I := 1 to ColCount - 1 do
	for J:= 1 to RowCount - 1 do
	if StrToInt(Cells[I,J])>Max then
	Max:=StrToInt(Cells[I,J]);
	Edit2.Text:=IntToStr(Max);
	end
	else Edit2.Text:='';
	if Checkbox3.Checked then
	{вычисление Sum}
	with StringGridl do
	begin
	Sum:=0;
	{обнулить значение суммы перед подсчетом}
	for I := 1 to ColCount - 1 do
	Sum:=Sum+StrToInt(Cells[I,I]);
	Edit3.Text:=IntToStr(Sum);

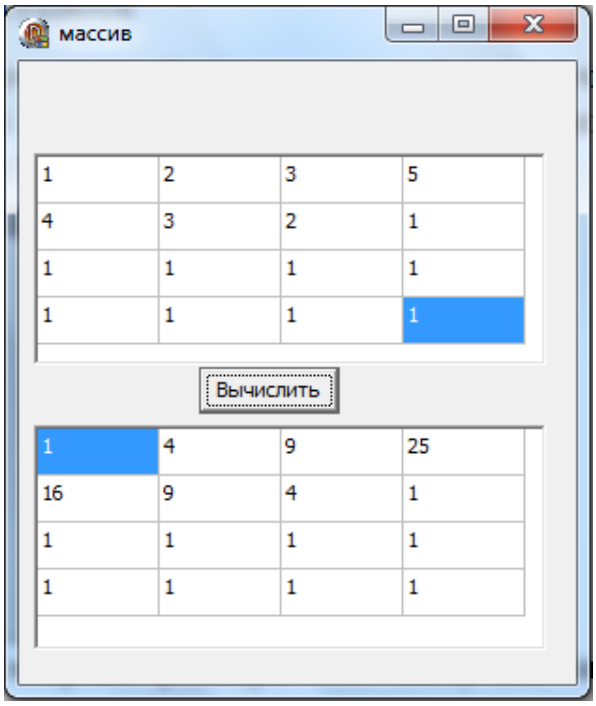
	end else Edit3.Text:=''; end;
12. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.	
13. Щелкнув мышью на кнопке Заполнить, проверьте заполнение надписей номеров строк и столбцов, а также заполнение объекта StringGrid значениями элементов двумерного массива.	

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Дан двумерный целочисленный массив размера 4x4. Подсчитать сумму элементов, стоящих над главной диагональю. Для отображения массива использовать компонент StringGrid



2. Дана целочисленная матрица размера 4x4. Получить квадрат этой матрицы.



3. Создать приложение, осуществляющее вывод в окно двумерного массива – табеля успеваемости по нескольким школьным предметам с оценками за 1-4 четверти и за год, причем годовая оценка должна вычисляться как среднее арифметическое всех оценок по данному предмету. Используйте закраску ячеек таблицы разными цветами, которые могут определяться как на стадии разработки формы приложения, так и во время работы приложения, в зависимости от значения элемента массива.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №15

Библиотека визуальных компонентов

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонентов MainMenu и PopupMenu.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задания указанные ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Главное меню является компонентом класса TMainMenu. Основным свойством этого класса является свойство

Property Items : TmenuItem; default;

Содержит элементы нулевого уровня главного меню приложения. Чаще всего главное меню создается на этапе конструирования формы.

Кроме главного меню, связанного с формой приложения, имеется компонент Popup Menu, предназначенный для создания контекстного меню. Контекстное меню может быть создано для любого оконного элемента управления. Для вызова контекстного меню необходимо поместить курсор мыши на оконный элемент и нажать правую кнопку мыши. Для установления связи между оконным элементом и компонентом Popup Menu используется свойство Popup Menu, в котором следует указать имя соответствующего компонента меню. Контекстное меню, как и главное, создается пи помощи конструктора меню.

Элементы, как главного так и контекстного меню являются объектами класса TMenuItem, который является непосредственным потомком класса TComponent. Элемент меню может представлять собой подменю или разделительную линию.

Рассмотрим основные свойства класса TMenuItem:

Property Caption : string; //содержит текст меню

Property Checked : Boolean; //если свойство имеет значение True, то элемент меню помечается галочкой

Property Enabled: Boolean; //если True, то элемент меню реагирует на события от мыши и клавиатуры, иначе элемент недосупен и выделяется тусклым цветом

Property Items[index:integer] : TMenuItem; default; //задает младшие элементы по отношению к текущему элементу, нумерация с 0

Property ShortCut : TShortCat; //определяет комбинацию «горячих» клавиш

Property OnClick : TNotifyEvent: //это событие возникает при выборе элемента мышью или при нажатии клавиши Enter, когда элеемнт меня активен

Property Alignment : TPopup Alingment; //определяет расположение контекстного меню относительно курсора мыши

Property AvtoPopup : Boolean; //если свойство имеет значение true, контекстное меню появляется при нажатии клавиши мыши, если значение False, меню не появляется(в этом случае следует использовать метод Popup). По умолчанию имеет свойство True. Метод Popup определяется следующим образом:

Procedure Popup(x, y : integer); virtual; // Выводит на экран меню, при этом координаты его левого верхнего угла равны x и y. В классе TPopupMenu определено событие OnPopup:

Property OnPopup: TNotifyEvent; //возникает при вызове контекстного меню при нажатии правой клавиши мыши, если свойтво AvtoPopup имеет значение True, либо при вызове метода Popup.

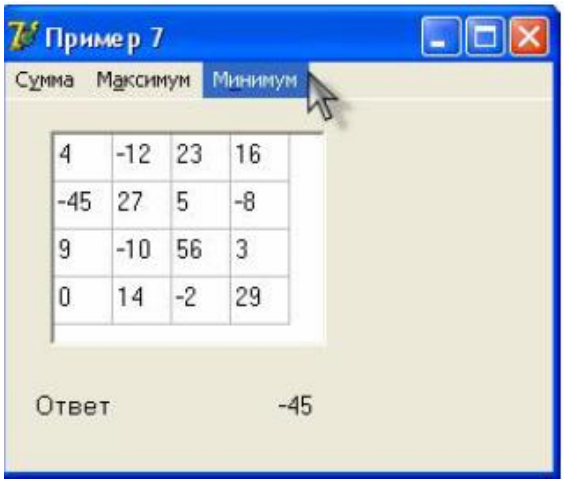
Задание 1. Дана квадратная целочисленная матрица 4x4 . создать главное меню, позволяющее выбрать одну из трех команд:

Максимум (нахождение максимума элементов матрицы)

Минимум (нахождение минимума элементов матрицы)

Сумма (нахождение суммы элементов матрицы).

Для ввода и отображения на форме элементов матрицы использовать компонент StringGrid.



1. Сохраните файл проекта и программного модуля	1) File/Save As н) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Главное меню»; о) Открыть эту папку; п) Нажать кнопку сохранить; 2) File/Save Project As Задать имя файла matrica и нажать сохранить
2. Самостоятельно расположить на форме нужные компоненты в соответствии с рисунком. Изменить свойства формы и компонентов.	
3. Примечание 2 Для решения задачи поместим на форму компонент StringGrid1 и установим ему значения свойств, указанные в примере 7. Поместим на форму компонент Label1, свойство Caption которого равно «Ответ», и компонент Label2, в котором будем размещать вычисленное значение. Со страницы Standard поместим на форму компонент MainMemu1 и щелкаем по кнопке с тремя точками. Появится око конструктора меню. Введем название пунктов меню. Для определения реакции на выбор пунктов следует по очереди выбирать все пункты меню и щелкать по ним мышкой. По умолчанию свойству Name пунктов меню присваиваются соответственно имена N1, N2, N3.	
4. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
5. Фиксация положения компонентов на форме.	Edit/Lock Controls (для отмены повторно задать команду)
6. Сохранение изменений.	Save All (на стандартной панели инструментов)
7. Компиляция проекта.	Project/Compile matrica
8. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
9. Расположение окна приложения	Object Inspector/ Position /poScreenCenter
14. Полный текст программы	Type Tform1= class(TForm) StringGrid1 : TString; Label1:Tlabel;

```
        Label2:Tlabel;
        MAinMenu1: TMainMenu;
        N1: TMenuItem;
        N2: TMenuItem;
        N3: TMenuItem;
        Procedure N1Click(Sender: Tobject);
        Procedure N2Click(Sender: Tobject);
        Procedure N3Click(Sender: Tobject);
        Private {Private declarations }
        Public {Public declarations}
End;
Var Form1: TForm1;
Implementation
{$R*.dfm}
Procedure TForm1.N1Click(Sender: TObject);
Var i , j , s: integer;
        Begin
        S:=0;
        For i:=o to stringgrid1.colcount-1 do
        For j:=0 to stringgrid1.rowcount-1 do
        S:=s+strtoint(stringgrid1.cells[i,j]);
        Label2.caption:= inttostr(s);
        End;
Procedure TForm1.N2Click(Sender: Object);
Var i , j ,max: integer;
Begin
        Max:= strtoint (stringgrid1.cells[0,0]);
        For i:=o to stringgrid1.colcount-1 do
        For j:=0 to stringgrid1.rowcount-1 do
        If strtoint(stringgrid1.cells[i,j])>max
        Then
        max:=strtoint(stringgrid1.cells[i,j]);
        Label2.Caption: inttostr(max)
 End;
Procedure TForm1.N3Click(Sender: Object);
Var i , j ,min: integer;
Begin
        Min:= strtoint (stringgrid1.cells[0,0]);
        For i:=o to stringgrid1.colcount-1 do
        For j:=0 to stringgrid1.rowcount-1 do
        If strtoint(stringgrid1.cells[i,j])<min
        Then min:=strtoint(stringgrid1.cells[i,j]);
        Label2.Caption: inttostr(min)
End;
End.
```

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

Задание 2. Создать контекстное меню для изменения цвета и размера шрифта в поле строки ввода Edit.

1. Создайте форму	File/New/VCL Form Application
2. Самостоятельно расположить на форме нужные компоненты. Изменить свойства формы и компонентов.	
3. Сохраните файл проекта и программного модуля.	3) File/Save As р) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Контекстное меню»; с) Открыть эту папку; т) Нажать кнопку сохранить; 4) File/Save Project As Задать имя файла menu и нажать сохранить
4. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
5. Фиксация положения компонентов на форме	Edit/Lock Controls (для отмены повторно задать команду)
6. Сохранение изменений	Save All (на стандартной панели инструментов)
7. Компиляция проекта	Project/Compile menu
8. Установим нужные свойства Edit1	Width=210 Height=70 AvtoSize=False
9. Создаем контекстное меню	Поместить в поле Edit компонент PopupMenu→Выделить его→нажать на кнопку с тремя точками около свойства Items→в открывшемся окне вводим название подменю: «Цвет» и «Размер» →выделим цвет и жмем правой кнопкой мыши→ выбираем Create SubMenu→вводим название пунктов подменю Цвет(Красный и синий) →Аналогично с пунктом Размер(16,24) →для определения реакции на выбор пунктов меню следует по очереди выбирать все пункты меню и щелкать по ним мышкой
10. Процедура обработки «Цвет→красный»	Procedure TForm1.N2Click(Sender: TObject); Begin edit1.Font.Color:=clred;end;
11. Процедура обработки «Цвет→синий»	Procedure TForm1.N3Click(Sender: TObject); Begin edit1.Font.Color:=clblue;end;
12. Процедура обработки «размер→16»	Procedure TForm1.N5Click(Sender: TObject); Begin edit1.Font.Size:=16;end;

13. Процедура обработки «размер→24»	Procedure TForm1.N6Click(Sender: TObject); Begin edit1.Font.Size:=24;end;
Примечание 2 В заключение установим свойство PopupMenu компонента Edit1 равным PopupMenu1	
14. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
15. Расположение окна приложения	Object Inspector/ Position /poScreenCenter
16. Сохраните файлы модуля и проекта, откомпилируйте и запустите приложение.	

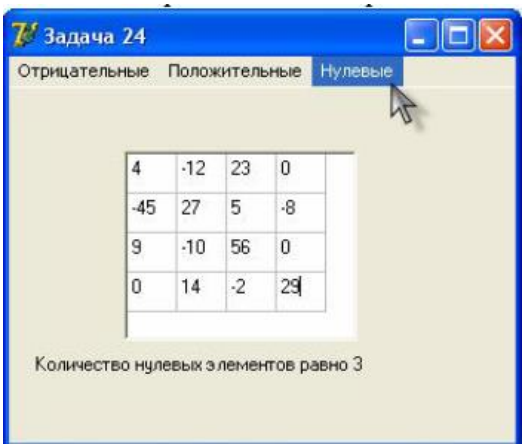
САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Дана квадратная целочисленная матрица размера 4x4. Создать приложение с главным меню для выбора одной из трех команд, позволяющих найти количество:

   Отрицательных элементов матрицы

   Положительных элементов матрицы

   Нулевых элементов матрицы

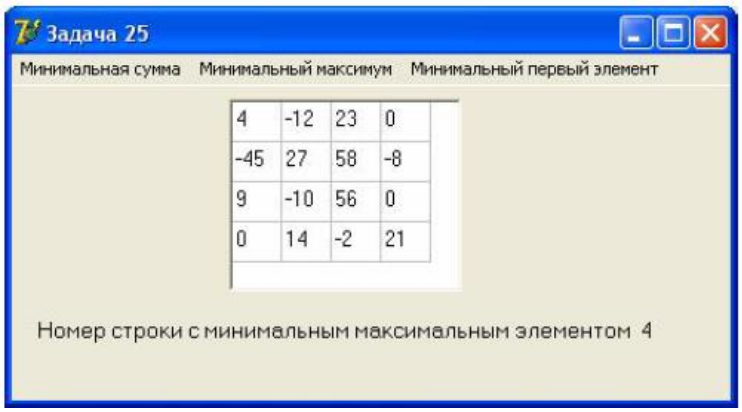


2. Дана квадратная целочисленная матрица размера 4x4. Создать приложение с главным меню для выбора одной из трех команд, позволяющих найти количество:

   С минимальной суммой элементов строки

   С минимальным максимальным элементом строки

   С минимальным первым элементом строки



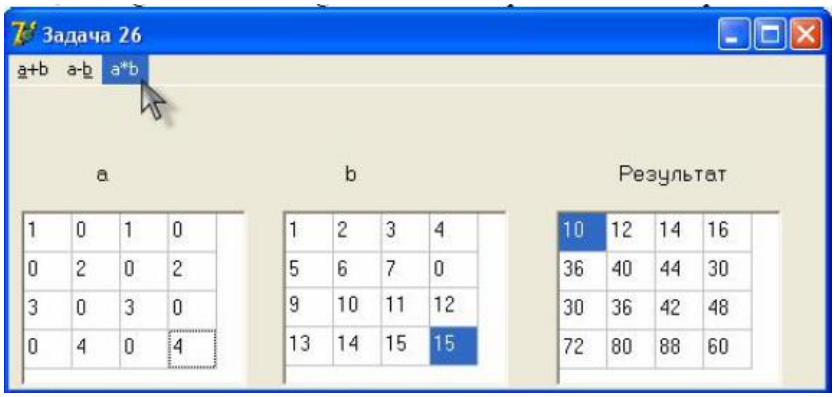
3. Даны две квадратные целочисленные матрицы a и b размера 4x4. Создать приложение с главным меню для выбора одной из трех команд, позволяющих найти

   a+b

   a-b

   a*b

   Для ввода и отображения на форме элементов матриц a и b, а также матрицы, получающейся в результате, использовать компоненты StringGrid1, StringGrid2, StringGrid3



4. Создать приложение, позволяющее менять цвет формы с помощью контекстного меню.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №16

Библиотека визуальных компонентов

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений с применением компонентов TShape, TTimer.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задания указанные ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Примечание

Таблица1 - Описание свойств компонента Shape

Свойство	Обозначение
Высота	Height
Ширина	Width
Вид	Shape
Заливка	Brush
Обводка	Pen

Задание1 Создайте приложение с компонентом Shape, организуйте и запрограммируйте двух студенческий диалог между ними.

1. Создайте форму	File/New/VCL Form Application
2. Добавьте на форму компоненты: TEdit1, TButton1, TButton2, TShape1 – лицо студента (stRoundSquare) TShape2,3 – брови студента (stRectangle) TShape4,5 – глаза студента (stCircle) TShape6,7 – зрачки студента (stCircle) TShape8 – нос студента (stEllipse) TShape9,10 – губы студента TShape11 – лицо студентки (stCircle) TShape12,13 – брови студентки (stRectangle) TShape14,15 – глаза студентки (stCircle) TShape16,17 – зрачки студентки (stCircle) TShape18 – нос студентки (stEllipse) TShape19,20 – губы студентки	
3. Сохраните файл проекта и программного модуля.	7) File/Save As у) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «Диалог»; ф) Открыть эту папку; х) Нажать кнопку сохранить; 8) File/Save Project As Задать имя файла Main и нажать сохранить
4. Активизация окна редактора	F12 (были сгенерированы описания формы и размеры компонентов на ней)
5. Фиксация положения компонентов на форме	Edit/Lock Controls (для отмены повторно задать команду)
6. Сохранение изменений	Save All (на стандартной панели инструментов)
7. Компиляция проекта	Project/Compile dialog
8. Запуск программы и завершение работы приложения	Run (F9), Alt+F4
9. Расположение окна приложения	Object Inspector/ Position /poScreenCenter

10. Установить невидимость компонентов Shape10, Label1.	Procedure TForml.FormCreate(Sender: TObject)
	begin
	Shape10.Visible:=false;
	label1.Visible:=false;
	end;

11. Программируем речь студента. Выберите в окне Инспектора объектов объект Buttonl и на странице События произведите двойной щелчок на пустом поле списка в событии OnClick, наступающем в момент получения	После этого в окне Редактора кода будет сгенерирована заготовка процедуры обработчика события procedure TForml.ButtonlClick(Sender: TObject);, в которую следует добавить следующие операторы: procedure TForm1.Button1Click(Sender:

элементом фокуса.	TObject);
	begin
	label1.Caption:='Что такое Делфи?';
	Timer2.Enabled:=true;
	end;

12. Программируем речь студентки. Выберите в окне Инспектора объектов объект Button2 и на странице События произведите двойной щелчок на пустом поле списка в событии OnClick, наступающем в момент получения элементом фокуса.	procedure TForm1.Button2Click(Sender: TObject);
	begin
	Timer1.Enabled:=true;
	Edit1.Text:='Привет';
	Font.Size:=8;
	end;

13. Программируем действия студента, происходящие во время выполнения Timer1.	procedure TForm1.Timer1Timer(Sender: TObject);
	begin
	Shape18.Visible:=false;
	label1.Visible:=true;
	Shape10.Visible:=true;
	label1.Caption:='Здравствуйте';
	Edit1.Text:='';
	Timer1.Enabled:=false;
	end;

14. Программируем действия студентки, происходящие во время выполнения Timer2.	procedure TForm1.Timer2Timer(Sender: TObject);
	begin
	Shape10.Visible:=false;
	label1.Visible:=false;
	Shape18.Visible:=true;
	Edit1.Text:='Неужели не знаешь?!';
	Timer2.Enabled:=false;
	end;

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

САМОСТОЯТЕЛЬНАЯ РАБОТА

Создать приложение с применением компонентов TTimer и TShape.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислить свойства компонента TTimer.
2. Перечислить свойства компонента TShape.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №17

Файлы

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений по обработке файлов.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задания указанные ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Диалоговые окна

В состав Windows входит ряд типовых диалоговых окон, предназначенных для открытия и сохранения файлов, выбора шрифта, цвета и некоторые другие. В Delphi реализованы классы, объекты которых дают программисту способы создания и использования таких окон. Рассмотрим свойства некоторых компонентов, с помощью которых в Delphi реализуются диалоговые окна.

Компонент OpenDialog предназначено для просмотра файловой системы компьютера и выбора имени требуемого файла, он не предназначен для автоматического открытия файлов,а позволяет лишь получить имя выбранного пользователем файла.

Рассмотрим основные свойства класса TOpenDialog.

property DefaultExt: string; Содержит расширение, добавляемое к имени файла, если у него не указано расширение.

property FileName : string; Содержит имя выбранного файла.

property Files : TStrings; Содержит список имён выделенных файлов.

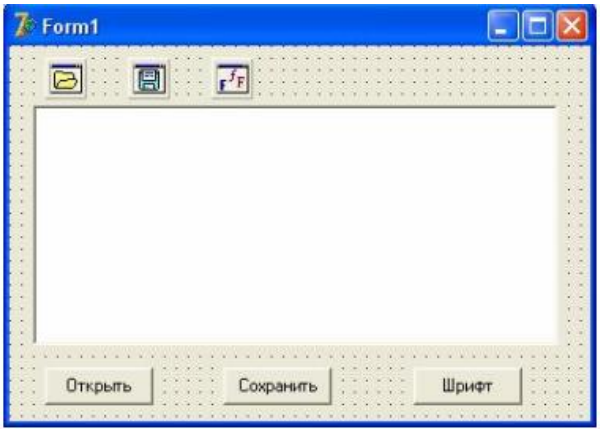
property Filter : string; Содержит описание одного или нескольких файловых фильтров. Па- пример, фильтр * . pas поможет пользователю отображать в диалоговом окне только файлы, имеющие расширение . pas. property InitialDir : string; Определяет папку, содержимое которой появляется при открытии диалогового окна, function Execute : boolean;

Размещает диалоговое окно на экране в модальном режиме. Модальный режим означает, что выполнение приложения приостанавливается до тех пор, пока пользователь не закроет модальное окно. Функция возвращает значение true, если окно закрыто кнопкой Открыть, и false, если закрыто кнопкой Отмена.

Диалоговое окно SaveDialog очень похоже на окно OpenDialog, но в отличие от него используется при сохранении файла.

Диалоговое окно FontDialog позволяет пользователю выбирать шрифт и устанавливать его характеристики. Основным свойством компонента FontDialog является свойство Font, задающее характеристики шрифта.

Задание 1. Создадим простой текстовый редактор, который позволил бы с помощью диалоговых окон открывать и сохранять текстовые файлы, а также изменять характеристики шрифта.



1. Сохраните файл проекта и программного модуля	9) File/Save As ц) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «file1»; ч) Открыть эту папку; ш)Нажать кнопку сохранить; 10) File/Save Project As Задать имя файла fil1 и нажать сохранить

Примечание 2

Разместим на форме следующие компоненты: OpenDialog1, SaveDialog1, FontDialog1, Memol, Button1, Button2, Button3.

Выберем свойство Filter компонента OpenDialog1 и щёлкнем по появившейся кнопке с тремя точками. Появится диалоговое окно Filter

Filter Name	**Filter**
Текстовые файлы (*.txt, *.doc)	*.txt; *.doc
Все файлы (*.*)	* *

После заполнения нажать кнопку Ок.

Для компонента SaveDialog значение свойства DefaultExt установим равным txt, т.е., если при сохранении файла расширение не будет указано, то по умолчанию добавится расширение txt.

Кнопкам Button1, Button2, Button3 установим свойство Caption равным 'Открыть', 'Сохранить', 'Шрифт'.

2. Полный текст программы	unit Unit 1; interface uses Windows, Messages, SvsUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls; type TForml = class(TForm) OpenDialogl: TOpenDialog; SaveDialogl: TSaveDialog; Button1: TButton; Button2: TButton; Button3: TButton;

```
procedure    ButtonlClick(Sender:    TObject);
procedure    Button2Click(Sender:    TObject);
procedure    Button3Click(Sender: TObject);

private
    { Private declarations }
public
    { Public declarations } end;

var
    Forml: TForml;
 implementation {SR *.dfm}

    procedure TForml.ButtonlClick(Sender:
TObject);
    begin
          if not opendialogl.execute then exit;
          memo1.Lines.LoadFromFile(OpenDialogl.
filename)
    end;

    procedure TForml .Button2Click(Sender:
TObject);
    begin
          if not savedialogl.execute then exit;
          memo1.Lines.SaveToFile(savedialogl
.filename)
    end;

    procedure TForml.Button3Click(Sender:
TObject);
    begin
          if not fontdialogl.execute then exit;
          memo1.Font:=fontdialogl.font
     end;
end.
```

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

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Стандартные подпрограммы для доступа к файлам

Перед использованием файловой переменной она должна быть связана с внешним файлом (файлом на диске) с помощью вызова процедуры

AssignFile(< файловая переменная>, < имя файла>);

здесь <файловая переменная> - имя переменной файлового типа, объявленной в программе;

<имя файла>- символьная строка, содержащая имя файла.

Если файл находится в одной папке с обрабатывающей его программой, то достаточно указать только имя файла, в противном случае надо указать полный путь к файлу, например:

'c:\files\zl.txt'

Reset(< файловая переменная>); Процедура reset открывает существующий внешний файл, имя которого было связано с файловой переменной. Если внешний файл с указанным именем отсутствует, то возникает ошибка периода выполнения программы. Если файл уже открыт, то он сначала закрывается, а затем открывается вновь. Файловый указатель устанавливается на элемент файла с порядковым номером 0.

Текстовый файл, открытый процедурой reset, доступен только для чтения. Для типизированных и нетипизированных файлов, открытых процедурой reset, допускается выполнять операции чтения и записи в файл.

Rewrite(<файловая переменная >); Процедура rewrite создаёт новый файл, имя которого связано с файловой переменной. Если файл с указанным именем уже существует, то он удаляется и на его месте создаётся новый пустой файл. Текущая позиция в файле устанавливается на начало файла, т.е. указатель будет указывать на элемент с порядковым номером 0.

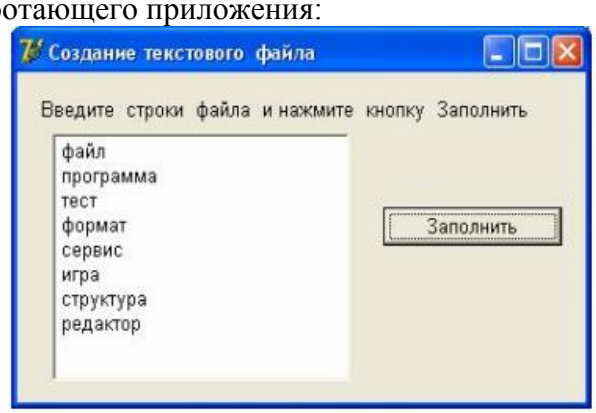
Append(<файловая переменная >); Процедура append открывает уже существующий внешний файл, связанный с файловой переменной, для добавления новой информации. Если файла с указанным именем не существует, то возникает ошибка. Если файл уже открыт, то он сначала закрывается, а затем открывается заново. Указатель будет указывать на конец файла. В результате обращения к append текстовый файл становится доступным только для записи.

CloseFile(<файловая переменная>); Процедура closefile закрывает открытый файл. При этом обеспечивается сохранение в файле всех новых записей и регистрация файла в папке. Процедура closefile не разрывает связь файла с файловой переменной, поэтому файл можно открыть снова без повторного использования процедуры assignfile.

Уничтожить файл f можно с помощью процедуры Erase(var f). Переименовывает файл процедура Rename(var f; NewName : string). Здесь NewName - строка, содержащая новое имя файла. Перед выполнением этих процедур необходимо закрыть файл.

Текстовые файлы

Задание 2. Создать текстовый файл и заполнить его информацией, введённой в редакторе Memo. Окно работающего приложения:



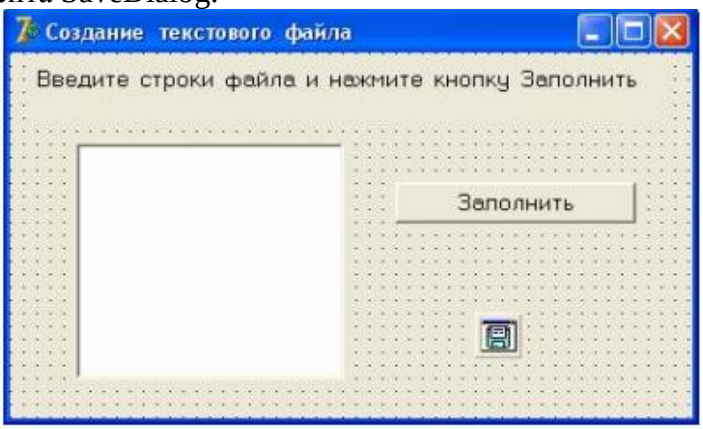
1. Сохраните файл проекта и программного модуля	11) File/Save As щ)При сохранении модуля в диалоговом окне Save Unit1 As создать папку «file2»; ы) Открыть эту папку; э) Нажать кнопку сохранить; 12) File/Save Project As Задать имя файла fil2 и нажать сохранить

2. Создание текстового файла с использованием метода SaveToFile	procedure TForml.ButtonlClick (Sender: TObject);
	begin
	memo1.Lines.SaveToFile('c:\Files\zl.txt');
	end;

3. Создание текстового файла без использования метода SaveToFile	procedure TForml.ButtonlClick (Sender: TObject);
	var
	f: textfile;
	n, i : integer;
	begin
	assignfile(f, 'c:\Files\zl.txt');
	rewrite(f);
	n:=memol.Lines.Count;
	for i:=0 to n-1 do writeln(f, memo1. lines. strings[i]);
	closefile(f);
	end;

Примечание 3

Приведённый здесь текст программы содержит имя создаваемого файла (c:\Files\zl.txt). Поэтому, чтобы создать другой файл, нужно будет внести изменения в текст программы. Добавив на форму компонент SaveDialog, мы получим удобное средство для выбора имени создаваемого файла на этапе работы приложения. Окно приложения после добавления компонента SaveDialog:



4. Преобразованная процедура Button1Click	procedure TForml.Button1Click(Sender: TObject);
	var f: textfile;
	n, i :integer;
	begin
	if not savedialogl.Execute then exit;
	assignfile(f, savedialogl.filename);
	rewrite(f);

	n:=memol.Lines.Count;
	for i:=0 to n-1
	do writeln(f, memo1.lines.stringsf[i]);
	closefile(f);
	end;

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

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Типизированные файлы

Доступ к текстовым файлам возможен только последовательно, то есть, когда элемент считывается или записывается, указатель файла перемещается к следующему по порядку элементу файла. К типизированным файлам можно организовать прямой доступ с помощью стандартной процедуры

procedure seek(var f; n:longint); которая перемещает файловый указатель в типизированном файле, связанном с файловой переменной f, на элемент с номером n. Нумерация элементов в файле начинается с нуля.

Для определения текущей позиции в файле используется функция:

function filepos(var f):longint;

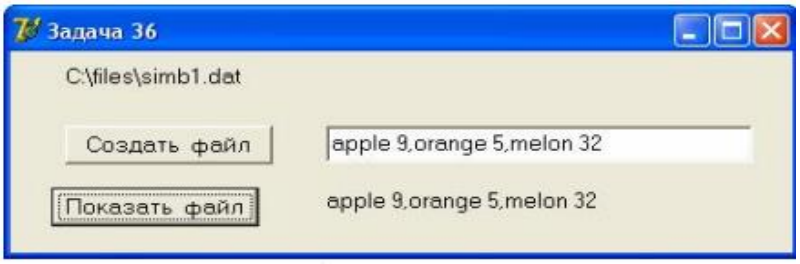
Для определения размера файла используется функция:

function filesize(var f):integer;

Например, для установки файлового указателя на последний элемент файла f достаточно записать:seek(f, filesize(f)-l); на первый элемент файла seek(f, 0); вернуться на один элемент назад seek(f, filepos(f)-l).

Применение процедур assignfile и closefile для типизированных файлов не отличается от текстовых файлов. Процедура reset, в отличие от текстовых файлов, допускает для типизированных файлов не только чтение, но и запись в файл. Процедура rewrite, в отличие от текстовых файлов, допускает не только запись, но и чтение из файла. Процедура append и функция eoln для типизированных файлов не работают.

Задание 3. Создать типизированный файл, состоящий из символов, введённых в окно ввода Edit. Вывести содержимое созданного файла в поле метки Label.



1. Сохраните файл проекта и программного модуля	13) File/Save As ю)При сохранении модуля в диалоговом окне Save Unit1 As создать папку «file3»; я) Открыть эту папку; аа) Нажать кнопку сохранить;

<table>
<tr><td></td><td>14) File/Save Project As
Задать имя файла fil3 и нажать сохранить</td></tr>
<tr><td>2. Создание типизированного файла символов:</td><td>procedure Forml.ButtonlClick(Sender: TObject);
var f: file of char;
i : integer;
begin
if not savedialogl.execute then exit;
label1.Caption:=savedialogl.FileName;
assignfile(f, savedialogl.FileName);
rewrite(f);
for i:=l to length(editl.Text) do
write(f, edit1.text[i]);
closefile(f);
end;</td></tr>
<tr><td>3. Вывод содержимого типизированного файла в поле метки Label2</td><td>procedure TForml.Button2Click(Sender: TObject);
var f: file of char;
d : char;
begin
if not opendialogl.execute then exit;
assignfile(f, opendialogl.FileName);
reset(f);
label2.Caption:=’’;
while not eof(f) do
begin
read(f,d);
label2.caption:=label2.caption+d
end;
closefile(f)
end;</td></tr>
<tr><td colspan="2">4. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.</td></tr>
</table>

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Нетипизированные файлы

Нетипизироваиный файл представляет собой последовательность байтов, содержащих данные произвольного типа и структуры. Основное назначение нетипизированных файлов - обеспечение совместимости с любыми типами файлов и организация высокоскоростного обмена данными между внешними запоминающими устройствами и оперативной памятью

Описание нетипизированного файла f имеет вид:

var f: file;

В процедурах reset и rewrite для нетипизированных файлов указывается дополнительный параметр RecSize, чтобы задать размер записи, используемой при передаче файла:

procedure reset (var f: file {; RecSize : word});
procedure rewrite (var f: file {; RecSize : word});

Если параметр RecSize не указан, то принимаемая по умолчанию длина записи равна 128 байтам. Длина записи измеряется в байтах и может быть задана произвольным целым числом - от 1 байта до 2 Гбайт. Если задать длину записи, кратную 512 байт, то это позволит выполнять операции чтения-записи для нетипизировапного файла с максимальной скоростью.

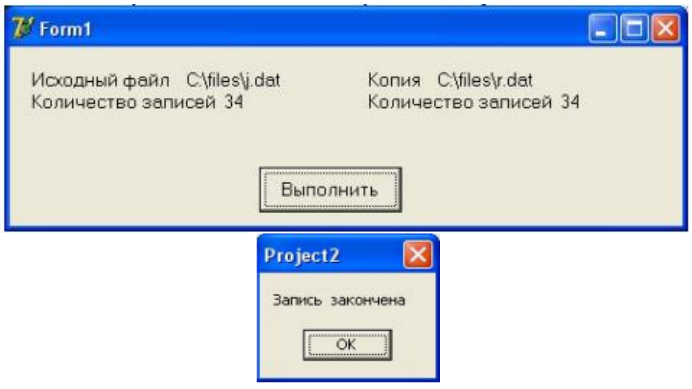
Процедуры blockread и blockwrite, позволяют пересылать данные с высокой скоростью: procedure blockread(var f: file; var buf; count: integer; {var at: integer}); procedure blockwrite(var f: file; var buf; count: integer; {var at: integer});

Здесь f - имя файловой переменной, связанной с нетипизированным файлом, buf - переменная, в которую будут помещаться данные при чтении из файла или из которой будут извлекаться данные при записи в файл. Count - параметр целого типа, указывающий, какое количество записей необходимо прочитать или записать за одно обращение к файлу. Переменная buf должна иметь длину не меньшую, чем count*RecSize байт. Необязательный параметр at содержит количество реально прочитанных или записанных записей.

Задание 4. Составить программу для копирования файла. Если задать длину записи (RecSize), кратную 512 байт, то скорость копирования будет большая, но так как длина файла в общем случае не кратна заданной длине записи, то в файле будут присутствовать неполные записи. В частности, если длина файла окажется меньше RecSize, то созданный в результате копирования файл будет пустой. Если задать длину записи, равную одному байту, то это позволит точно отразить размер любого файла, так как в этом случае в файле не могут присутствовать неполные записи, то есть записи с длиной меньшей, чем RecSize.

Размер одной записи в процедурах reset и rewrite установим равным 1 байту. В процедурах blockread и blockwrite параметр count установим равным 10 (запись и считывание будут осуществляться по 10 записей). Значение параметра count должно быть как можно больше при копировании больших файлов. В качестве буфера будем использовать переменную b, являющуюся массивом типа byte, содержащим 10 элементов.

Для вывода сообщения об окончании переписывания файла будем использовать процедуру showmessage(str), параметром которой является строка выводимого сообщения. Окно работающего приложения:



1. Сохраните файл проекта и программного модуля	15) File/Save As бб) При сохранении модуля в диалоговом окне Save Unit1 As создать папку «file4»; вв) Открыть эту папку;

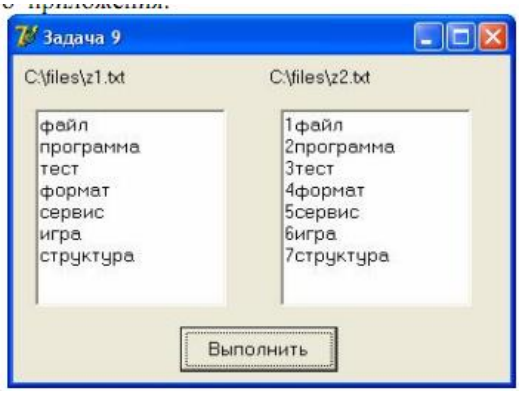
	гг)Нажать кнопку сохранить; 16) File/Save Project As Задать имя файла fil4 и нажать сохранить

2. Полный текст программы	procedure TForml.Button1Click(Sender: TObject);
	var fl,f2:file;
	at1, at2:integer;
	b:arrav[l.. 10] of byte;
	begin
	if not opendialogl.Execute then exit;
	assignfile(f1, opendialog1.filename);
	if not savedialogl.Execute then exit;
	assignfile(f2, savedialog1.filename);
	reset(fl, 1);
	rewrite(f2, 1);
	repeat
	blockread(fl, b, 10, atl);
	blockwrite(f2, b, atl, at2);
	until atl<10;
	label1.Caption:=’Исходный файл'+
	opendialogl.filename + #13 +
	'Количество записей ' +
	inttostr(filesize(f1));
	label2.Caption:=’Копия ' +
	savedialogl.filename + #13 +
	'Количество записей ' +
	inttostr(filesize(f2));
	closefile(fl);
	closefile(f2);
	showmessage(Запись окончена');
	end;

Примечание 4 Например, пусть копируется файл C:\files\j.dat, содержащий следующую информацию: file,edit,search,view,project,run. Количество записей этого файла равно 34. Поэтому на первых трёх шагах цикла repeat значение переменной at1 равно 10, а на четвёртом шаге atl=4<10 и, следовательно, цикл завершит свою работу.	
3. Сохраните файлы проекта и программного модуля, откомпилируйте и запустите программу на выполнение.	

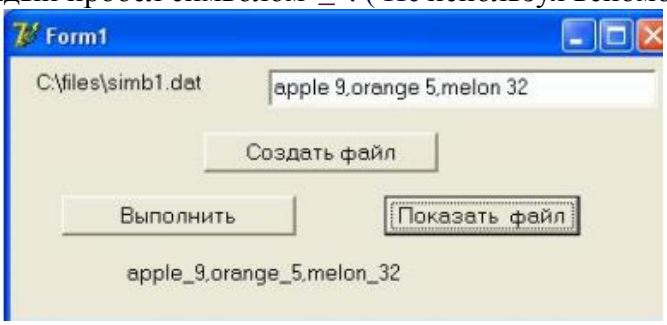
САМОСТОЯТЕЛЬНАЯ РАБОТА

1. В начало каждой строки файла вставить её номер и записать преобразованные строки в новый файл. Дня выбора имён исходного и создаваемого файла использовать диалоговые окна. Окно работающего приложения:

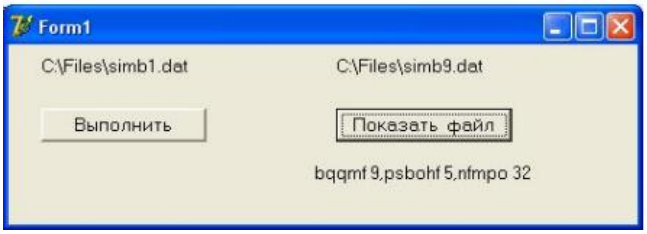


2. Дан текстовый файл, содержащий строки. Переписать содержимое файла в другой файл, сдвигая циклически каждую строку вправо на 3 символа. Например, результатом сдвига строки abcdewrt будет строка wrtabcde. Для выбора имён исходного и создаваемого файла использовать диалоговые окна. Окно работающего приложения:

3. Создать типизированный файл, состоящий из символов, введённых в окно ввода Edit. Заменить в файле каждый пробел символом '_ '. (Не используя вспомогательный файл.)



4. Переписать содержимое данного символьного файла в новый символьный файл, заменяя каждую встречающуюся строчную английскую букву (кроме буквы z) на следующую по алфавиту, а все остальные элементы, оставляя без изменения.



5. Создать типизированный файл, состоящий из целых чисел, введенных в столбик в окно редактора Memo. Вывести содержимое созданного файла в поле метки Label1.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №18

Списки и коллекции

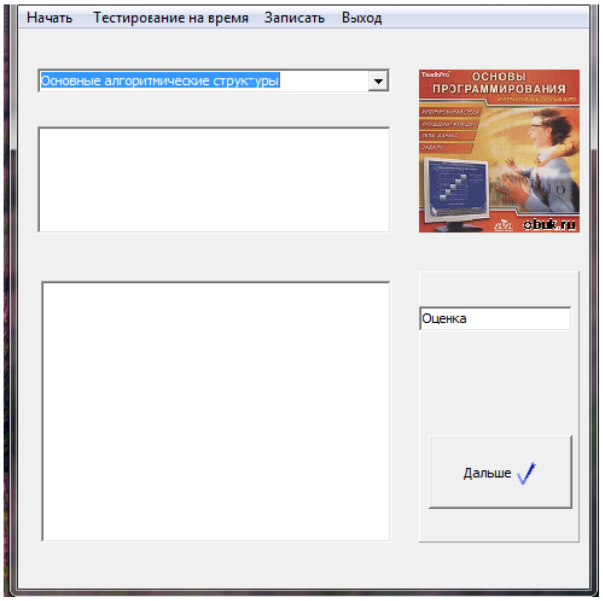
Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений, создание и отладка программы, содержащей списки и коллекции

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание указанное ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Задание 1 Разработать приложение для тестирования знаний по языку программирования Pascal. Тестирование проводится по пяти разделам курса. Последовательность выполнения тестирования следующая:

- Выбирается раздел предмета тестирования. На экран последовательно выводятся вопрос и четыре альтернативных ответа в случайном порядке. Студент выбирает один из ответов. Программа подсчитывает количество правильных ответов и выдает результирующую оценку по каждому разделу курса.
- Тестирование должно проводиться в двух режимах: без учета времени тестирования и с учетом времени тестирования.
- В программе необходимо предусмотреть запись результатов тестирования на диск. Для выбора раздела тестирования использовать компонент TComboBox.
- Вопросы и ответы тестов хранятся в текстовых файлах Ftextl, Ftext2, Ftext3, Ftext4, Ftext5 в соответствии с разделом тестирования. Вопрос отделен от ответов строкой из пяти звездочек, поскольку вопрос может располагаться в нескольких строках.



Алгоритм выполнения задания

№ п/п	Алгоритм	Конкретное действие, соответствующее предложенному алгоритму
1.	Создать заготовку для нового приложения и разработать конструкцию формы	Расположение компонентов как на рисунке. Форма Forml: BorderStyle — bsSingle; BorderIcon — biMinimize; Position — poScreenCenter. Комбинированный список ComboBoxl: Text — Разделы тестирования; Items — Основные алгоритмические структуры; Пользовательские типы; Процедуры, модули; Динамические структуры; Объекты. Многострочный редактор Memol: Lines — очистить. Список ListBox1. Метки Labell, Label2: Caption — очистить. Панель Panell: Caption — очистить. Редактор Editl: Text - Оценка. Кнопка Bitbtnl: Caption - Дальше; Glyph - файл arrow3r.bmp из каталога Images\Buttons. Контейнер Imagel: Picture - файл handshak.bmp из каталога Splash\256color. Компонент TMainMenu: N1 - Начать; N2 - Тестирование на время; N3 - Записать; N4 - Выход. Компонент Timer: Enabled - False; Interval - 180000. Компонент TSaveDialog.
2.	Составить перечень событий, на которые должно откликаться приложение	ListBoxlClick - выбор элемента списка (правильного ответа); ComboBoxlClick - выбор раздела тестирования; BitBtnlClick - переход к следующему вопросу раздела тестирования; FormCreate - для ввода фамилии тестируемого; N1Click - восстановление исходного режима работы; N2Click - для включения таймера; N3Click - запись результатов тестирования на диск; N4Click - конец работы приложения; TimerlTimer - определение времени окончания тестирования.
3.	Создать обработчики событий	Обработчики событий приводятся ниже по тексту.
4.	Отладить программу	Клавиша F9 для запуска программы.

Описание переменных:

```
Const count:array [1..5] of Byte=(60,90,100,120,40);
{Количество вопросов в разделах курса}
Type ar = Array[1..4] of Byte;          {Описание для массива 4-х случайных чисел}
Var
f : Text;                       {Файловая переменная для связи с файлом тестов}
otv : Array [1..4] of String;           {Массив для ответов}
a : ar;                         {Массив для четырех случайных чисел}
```

```
s,filename,fio : String;   {Переменные для строки из файла, имени файла, фамилии тестируемого}
Sav : Array [1..5] Of Integer;   {Массив оценок}
ivopr, kvopr, sum : Byte;   {Номер вопроса, номер раздела, сумма баллов}
```

Вспомогательная процедура для генерирования массива из четырех целых случайных чисел в интервале от 1 до 4:

```
Procedure Forty(Var a: ar);
    Var j : integer;
    m : Boolean;
    a1 : Set Of 1..4;
    Begin
          a1 := [];
               For j := 1 To 4 Do
                    Begin
                      m := False;
                      While Not m Do
                              Begin
                                 a[j] := (Random(4)+1);
                                If Not (a[j] in a1) Then
                                   Begin
                                      m := True;
                                      a1 := a1 + [a[j]]
                                   End;
                              End;
                  End;
      End;
End;
```

Выбор ответа:

```
procedure TForm1.ListBox1Click(Sender: TObject);
    Var k : Integer;
     Begin
         k:= Listbox1.ItemIndex;                    {Зафиксировали номер ответа}
         Listbox1.ItemIndex := -1;
                                                    {Вывод правильного ответа в положении метки}
         Labell.Caption := 'Правильный ответ ' + otv[1];          {Если выбран правильный ответ}
         If a[1]=k Then sum:=sum + 1;
     end;
end;
```

Выбор раздела:

```
procedure TForm1.ComboBox1Click(Sender: TObject);
    Var j :Integer;
    begin
      Label1.Caption:='Вопрос 1';
      Edit1.Text := 'Оценка';
      Label2.Caption := 'Выбери ответ';
```

```
        Randomize;
        sum := 0;
        kvopr:=ComboBox1.Itemindex +1; {Номер раздела тестирования}
        Filename:='Ftext'+inttostr(kvopr) +'.txt';
        Assignfile(f, Filename);
        Reset(f);
        Ivopr := 1;
                {Блок считывания вопросов из файла}
         s := '';
        While s<>'*****' do
         Begin
              Readln(f, s);
              If s <> '*****' Then Memo1.Lines.Add(s);
         end;
                {Считывание ответов из файла}
         For j:=1 to 4 do
         Begin
              Readln(f, s);
              otv[j]:= s;
          End;
          forty(a);                    {Сгенерировали случайный массив}
                              {Отразили ответы в ListBox1 в случайном порядке}
           For j:=1 To 4 Do Listbox1.Items[a[j]]:=otv[j];
End;
```

С помощью кнопки переходим к следующему вопросу:

```
procedure TForm1.BitBtn1Click(Sender: TObject);
   Var j, ss : Integer;
Begin
              {Очистили содержимое окон}
      For j := 0 To 4 Do Listbox1.Items[j]:= '';
      Memo1.Clear;
      Label3.Caption := 'Выбери ответ';
       ivopr:=ivopr + 1;
             { Если вопросы еще не окончились}
      If ivopr <= count[kvopr] Then
       Begin
            Label1.Caption := 'Вопрос ' + IntToStr(ivopr); {Считываем вопрос}
            s := '';
           While s<>'*****' Do
           Begin
                Readln(f, s);
                If s <>'*****' Then Memo1.Text := Memo1.Text + s;
           End;
                  {Считали ответы}
           For j:=1 To 4 Do
           Begin
                Readln(f, s);
               otv[j]:= s;
            End;
```

```
            forty(a);
            For j:=1 To 4 Do Listbox1.Items[a[j]] := otv[j];
        End
    Else                    {Если вопросы по теме закончились}
        Begin
                            {Обрабатываем результаты}
            If (Sum / Count[kvopr] >= 0.9) Then ss := 5;
            If (Sum / Count[kvopr] < 0.9) And (Sum / Count[kvopr] >= 0.75) Then ss := 4;
            If (Sum / Count[kvopr] < 0.75) And (Sum / Count[kvopr] >= 0.6) Then ss := 3;
            If (Sum / Count[kvopr] < 0.6) Then ss:= 2;
            Label3.Caption:='Вопросы по данной теме закончились! Сумма ' +IntToStr(sum);
            Edit1.Text := 'Оценка = ' + IntToStr (ss);
            Sav[Kvopr] := ss;
         End;
End;
```

Процедура для ввода фамилии тестируемого:

```
procedure TForm1.FormCreate(Sender: TObject);
Begin
       fio := InputBox('Фамилия', 'Введите фамилию','');
End;
```

Обработчик выбора пункта меню "Тестирование на время", который является подопцией пункта "Режим" главного меню приложения:

```
procedure TForm1.N2Click(Sender: TObject);
Begin
     Timer1.Enabled := True;          {Включили таймер}
End;
```

Обработчик для пункта меню "Выход" главного меню приложения:

```
procedure TForm1.N4Click(Sender: TObject);
Begin
    Form1.Close;
    Application.Terminate;
End;
```

Обработчик для подопции "Начать" пункта меню "Режим" главного меню приложения:

```
procedure TForm1.N2Click(Sender: TObject);
Begin
     fio := InputBox('Фамилия','Введите фамилию','');
End;
```

Обработчик события OnTimer (окончания времени, установленного в свойстве Interval компонента Timerl):

```
procedure TForm1.Timer1Timer(Sender: TObject);
Var j,ss : Integer;
Begin
                        {Обработка результатов тестирования}
    If (Sum / Count[kvopr] >= 0.9) Then ss := 5;
```

```
    If (Sum / Count[kvopr] < 0.9) And (Sum / Count[kvopr] >= 0.75) Then ss := 4;
    If (Sum / Count[kvopr] < 0.75) And (Sum / Count[kvopr] >= 0.6) Then ss := 3;
    If (Sum / Count[kvopr] < 0.6) Then ss := 2;
                {Вывод результатов тестирования}
    Label2.caption:='Время! Сумма = '+ IntToStr(sum);
    Editl.text := 'Оценка = ' + IntToStr(ss);          {Сохранили результат}
    Sav[Kvopr] := ss;                    {Очистили счетчик вопросов и окна приложения}
    ivopr := 0;
    Memol.Clear;
    For j := 0 To 4 Do Listbox1.Items[j]:= '';
     {Выключили таймер}
    Timer1.Enabled := False;
end;
```

Обработчик подопции "Сохранить результаты тестирования" опции "Сохранить" главного меню приложения:

```
procedure TForm1.N3Click(Sender: TObject);
Var ff : TextFile;
       i : Integer;
Begin
    With SaveDialogl Do              {Используем компонент SaveDialogl
     If Execute Then                 и его метод Execute}
        Begin
            AssignFile(ff, FileName);
            Rewrite (ff);
            Writeln(ff, fio);
             For i := 1 To 5 Do
                Writeln(ff, sav[i]);
            CloseFile (ff);
         End;
End;
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Разработать приложение для пересчета величины давления из одной системы измерения в другую. Выбор системы измерения осуществить с помощью компонента TComboBox.

1 Дин/см^{л}2 = 0.1 Па; 1 Кгс/м^{л}2 = 9.81 Па; 1 мм. Рт. Ст. = 133.0 Па 1 физ. Атм. = 1. 013 * 10^{л}5.

Предусмотреть два варианта расчета: для единичного значения и для интервала значений (выбор варианта можно организовать с помощью TRadioGroup).

2. Разработать приложение для ввода и вывода результатов сессии учебной группы. Вывести список двоечников; список студентов, сдавших сессию без троек; список студентов, сдавших сессию на одни пятерки. Ввод исходной информации представить в табличной форме (использовать компонент TStringGrid). Выбор типа вывода осуществить с помощью компонента TListBox. Предусмотреть вывод на печатающее устройство.

3. Разработать справочную систему по компонентам Delphi. Выбор страницы библиотеки компонентов осуществить с помощью TTabControl.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Характеристика компонента TRadioGroup.
2. Характеристика компонента TListBox.
3. Характеристика компонента TStringGrid.
4. Характеристика компонента TComboBox.
5. Характеристика компонента TTimer.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №19

Графика в Delphi

Цель занятия: формирование навыка работы со средой программирования Delphi при разработке приложений, содержащих элементы графики.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задания, указанные ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Рисование на канве по пикселам

Задание 1 Создайте приложение, которое строит пикселами график функции y= sin(x)*cos(x/6)

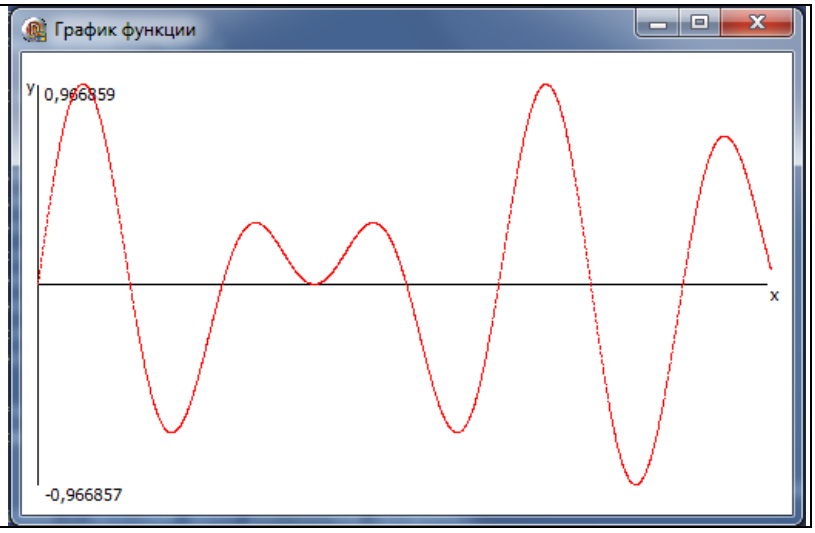
1. Создайте форму, свойству **Caption** которой присвойте значение «График функции». В окне Инспектора объектов задайте для свойства **Form1.ClientHeight** (высота окна формы) значение 300, для свойства **Form1.ClientWidht** (ширина окна формы) задайте значение 500, для свойства **Form1.Color** задайте значение **clWhite** (белый).	
2. Сохранение изменений	Save All (на стандартной панели инструментов) Сохранить модуль под именем gr1.pas
Примечание: Так как при программировании процедуры построения графика функции нам придется использовать для рисования **Canvas** (холст, канва), окрашивая пикселы канвы, изучите справочную информацию о свойствах **Canvas** и **Pixels.** Поскольку перед построением графика придется чертить прямые линии — оси координат и выводить текст на изображении, то следует также изучить справку Delphi о методах **MoveTo, LineTo** и **TextOut.**	
3. Создание процедуры обработчика события **OnPaint** (перерисовать изображение)	выберите в окне Инспектора объектов объект **Form1** и на странице **Events** произведите двойной щелчок на пустом поле списка в событии **OnPaint.** После этого в окне Редактора кода будет сгенерирована заготовка процедуры обработчика события **procedure TForml.FormPaint(Sender: TObject);** Введите в текст процедуры вызов процедуры построения графика

	функции **DrawGraph:**
	`procedure TForml.FormPaint(Sender: Object);`
	`begin`
	`  DrawGraph;`
	`{вызов процедуры построения графика функции}`
	`End;`

4. Код процедуры построения графика функции:	`procedure DrawGraph; {Построение графика функции}`
	`var`
	`xl, x2: real; {границы изменения аргумента функции}`
	`yl, y2: real; {границы изменения значения функции}`
	`x: real; {аргумент функции}`
	`y: real; {значение функции в точке x}`
	`dx: real;{приращение аргумента}`
	`left, b: integer; {левый нижний угол области вывода графика}`
	`w, h: integer; {ширина и высота области вывода графика}`
	`mx, my: real; {масштаб по осям X и Y}`
	`x0, y0: integer; {точка - начало координат}`
	
	`Function f(x: real):real; {функция, график которой надо построить}`
	`begin`
	`f:= Sin(x)*Cos(x/6);`
	`end;`
	
	`Begin`
	`{область вывода графика на форме}`
	`left:=10; {X - координата левого верхнего угла}`
	`b:=Forml.ClientHeight-20; {Y - координата левого верхнего угла}`
	`h:=Forml.ClientHeight-40; {высота}`
	`w:=Forml.Width-40; {ширина}`
	`xl:=0; {нижняя граница диапазона аргумента}`
	`x2:=25; {верхняя граница диапазона аргумента}`
	`dx:=0.01; {шаг аргумента}`
	
	`{определение максимального и минимального значений функции на отрезке [xl,x2] для масштабирования}`
	`yl:=f(xl); {минимальное значение функции}`
	`y2:=f(xl); {максимальное значение функции}`
	`x:=xl;`
	
	`{вычислять значения функции для значений x<=x2 и определить максимальное и минимальное`

	значения для масштабирования}
	repeat
	y:=f(x);
	{если текущее значение y меньше минимального, то yl:=y}
	if y<yl then yl:=y; {если текущее значение y меньше минимального, то y1:=y}
	if y>y2 then y2:=y; {если текущее значение y больше минимального, то y2:=y}
	x:=x+dx; {перейти к следующему значению x}
	until x>=x2; {как только x>=x2, прекратить}
	my:=h/abs(y2-yl); {масштаб по оси Y}
	mx:=w/abs(x2-xl); {масштаб по оси X}
	
	{задать начало координат}
	X0:=left;
	У0:=b-Abs(Round(yl*my));
	with Forml.Canvas do {рисование графика}
	
	begin
	{начертить оси координат}
	MoveTo(left, b); LineTo(left,b-h); {ось x}
	MoveTo(x0,y0); LineTo(x0+w,y0); {ось y}
	TextOut(left+5, b-h, FloatToStrF(y2,ffGeneral, 6, 3));
	{вывести максимальное значение}
	TextOut(x0+w,y0, 'x'); {подписать ось x}
	TextOut(1eft+5, b, FloatToStrF(yl,ffGeneral ,6,3));
	{вывести минимальное значение}
	TextOut(1eft-7, b-h-5, 'y'); {подписать ось y}
	{построение графика}
	x:=x1;
	repeat {повторять рисовать точки}
	y:=f(x);
	Forml.Canvas.Pixels[x0+Round(x*mx),y0-Round (y * my)]:=c1Red;
	{clRed - красный цвет рисования графика}
	x:=x+dx;
	until (x>=x2); {прекратить, как только x>=x2}
	end;
	end;

5. Проверка синтаксиса	Project/ Syntax Check gr1
6. Сохранение изменений	Save All
7. Компиляция проекта	Project/Compile gr1
8. Запуск программы	Run (F9)

9. Результат	

Рисование пером

Задание №2 Создайте приложение, осуществляющее рисование пером графика функции y=sin(x)*cos(x/6) на интервале $X_{min}=0$, $X_{max}=4\pi$. Вывод графика осуществляется после щелчка по кнопке «Построить график». Задать толщину пера 2 пиксела, цвет линии – синий (clBlue).

1. Самостоятельно создать форму «Рисование графика функции пером». Поместить в верхнюю часть формы компонент Image1, ниже компонент Button1. Сохраните файл под названием gr2.	
2. Сохранение изменений	Save All (на стандартной панели инструментов)

3. Создание кода обработчика[3] события.	Object Inspector/ Events/ OnClick/ двойной щелчок на пустом поле списка
	Основное тело процедуры будет иметь вид:
	procedure TForm1.Button1Click(Sender: TObject);
	var x,y:real; {значение аргумента и функции}
	px,py: longint; {координаты пикселов, соответствующих x,y}
	begin
	with Image1.Canvas do
	begin
	{задать толщину пера для рисования осей координат}
	Pen.Width:=1; {задать толщину пера для рисования осей координат}
	Pen.Color:=clBlack; {задать черный цвет линии}
	MoveTo(0,Image1.Height div 2);
	LineTo(Image1.Width, Image1.Height div 2); {Ось x}
	MoveTo(5,0);
	LineTo(5,Image1.Height); {Ось y}

[3] Повторить свойства Pen, Font, методы LineTo, MoveTo, TextOut.

	{рисовать график от начала координат на всю ширину Image1}
	Pen.Width:=2; {задать толщину пера для рисования графика}
	Pen.Color:=clBlue; {задать синий цвет линии}
	{установить перо в начало координат}
	MoveTo(5, Image1.Height div 2);
	For px:=5 to image1.width do
	Begin
	{вычисление значения аргумента x для масштабирования графика в соответствии с шириной Image1}
	x:=px*4*PI/Image1.Width;
	{вычисление значения функции}
	Y:=sin(x)*cos(x/6);
	{вычисление координаты пиксела, соответствующей значению функции Y}
	py:=Trunc(Image1.Height-(y+1)* Image1.Height/2)
	{рисование линии}
	LineTo(px,py);
	end;
	FontSize:=18; {задать размер шрифта в пунктах}
	TextOut(180,20,'y=sin(x)*cos(x/6)');
	End;
	End;

4. Проверка синтаксиса	Project/ Syntax Check gr2
5. Сохранение изменений	Save All
6. Компиляция проекта	Project/Compile gr2
7. Запуск программы	Run (F9)

Рисование кистью

У канвы имеется свойство Brush — кисть. Оно определяет фон и заполнение замкнутых фигур на канве. Свойство Brush имеет, в свою очередь, ряд подсвойств: Color, Bitmap, Handle и Style.

Color — цвет кисти. По умолчанию - clWhite (белый).

Handle — дескриптор кисти окна, определяющий доступ к дескриптору объекта GDI Windows.

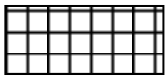
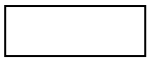
Style — определяет шаблон заполнения (штриховку).

Bitmap - указатель на внешнюю матрицу растрового изображения, используемую в качестве шаблона заполнения. Свойство кисти - Bitmap - определяет нестандартное заполнение заданным шаблоном, который задается матрицей размером 8 на 8 разрядов. Если для кисти задан шаблон Bitmap, то заполнение производится именно этим шаблоном, независимо от значения свой Style. Шаблон Bitmap может создаваться в процессе выполнения приложения или, например, загружаться из файла, как в приведенном ниже примере:

```
var
  Bitmap: TBitmap;
```

```
begin
  Bitmap := TBitmap.Create:                    {создание бъекта}
  try
  Bitmap.LoadFromFile('MyBitmap.bmp');  {загрузка в объект Bitmap
                                          матрицы из файла}
  Imagel.Canvas.Brush.Bitmap := Bitmap;
  …
        {использование шаблона для заполнения фигур на канве Imagel}
finally
Imagel.Canvas.Brush.Bitmap:=nil;              {обнуление указателя}
Bitmap.Free;                                 {освобождение памяти}
End;
End;
```

В этом примере создается объект **Bitmap** типа **TBitmap** и в него загружается матрица из файла с именем «MyBitmap.bmp». Затем свойству **Image1.Canvas.Brush.Bitmap** присваивается указатель на этот объект. После этого загруженный шаблон можно использовать для заполнения фигур на канве **Image1.** В конце фрагмента кода свойству **Bitmap** присваивается значение **nil,** после чего заполнение опять начинает определяться свойством **Style.** Затем объект **Bitmap** уничтожается, чтобы освободить занимаемую им память. Свойство **Style** может принимать следующие значения:

Значение	Шаблон	Значение	Шаблон
bsSolid		bsCross	
bsClear		bsDiagCross	
bsBDiagonal		bsHorizontal	
bsFDiagonal		bsVertical	

Имеется еще один метод канвы, связанный с кистью. Это метод **FrameRect.** Он рисует на канве текущей кистью прямоугольную рамку, не закрашивая ее.

Синтаксис метода **FrameRect:**

procedure FrameRect(const Rect: TRect);

Параметр **Rect** определяет позицию и размеры прямоугольной рамки. Толщина рамки составляет 1 пиксел. Область внутри рамки не заполняется. Метод **FrameRect** отличается от рассмотренного ранее метода **Rectangle** тем, что рамка рисуется цветом кисти (в методе **Rectangle** — цветом пера), и область не закрашивается (в методе **Rectangle** она закрашивается).

Например, оператор

```
with Imagel.Canvas do
begin
  Brush.Color := clBlack;
  FrameRect(Rect(10, 10, 100,100));
```

```
end;
```

рисует на канве компонента **Image1** черную рамку.

При работе с кистью имеется возможность использовать метод **TextRect** для более красивого вывода текста.

Например, чтобы нарисовать в заданном месте канвы компонента **Image1** красный прямоугольник и внутри него по центру вывести текст, введенный пользователем в окно редактирования **Edit1,** можно создать следующий код:

```
var st:string;
  X1, Y1, X2, Y2: integer;
Begin
 st:=Editl.Text;
 Xl:=100; Yl:=100; X2:=200; Y2:=150;    {координаты прямоугольника}
 with Imagel.Canvas do
   begin
     Brush.Color:=clRed;          {задать красный цвет кисти}
                                  {вывести текст, выровненный  по центру
                                   прямоугольника}
     TextRect (Rect(Xl, Yl, X2,Y2), Xl+(X2-Xl-TextWidth(st)) div 2, Yl+(Y2-Yl-
     TextHeight(st)) div 2,st);
   end;
end;
```

ПРИМЕЧАНИЕ

Если текст оказывается длиннее ширины прямоугольника, то он усекается. В данном примере будет видна только середина текста, так как текст выровнен по центру.

САМОСТОЯТЕЛЬНАЯ РАБОТА

1. Изменить приложение 1 таким образом, чтобы оно строило график функции y = sqrt(x)*sin(x). Вывести результат на печать.
2. Создать приложение, рисующее пером домик. Вывести результат на печать.
3. Протестировать примеры рисования кистью.
4. Создайте приложение, которое при щелчке на кнопке **Нарисовать** рисует в окне приложения цветок ромашки. Вывести результат на печать.
5. Создайте приложение, которое позволяет рисовать многоугольник. Количество сторон задается в окне редактора. Вывести результат на печать.
6. В таблице TDrawGrid отразить картинки, хранящиеся в файлах bmp. Имена файлов задать в виде типизированной константы-массива.
7. Создать авторский рисунок и распечатать его на принтере.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое мультимедиа?
2. Назначение свойства Canvas.
3. Какова разница между рисованием на канве по пикселам и пером?
4. Назначение методов LineTo, MoveTo, TextOut.
5. Как задать координаты и цвет пиксела канвы?
6. Как задается толщина, цвет пера и стиль линии, которую оно рисует?
7. Каково назначение свойства канвы **Brush** (кисть)? Какие подсвойства имеются у кисти?
8. Опишите назначение методов **FillRect, FloodFUl, FrameRect.**

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №20

Мультимедиа в Delphi

Цель занятия: формирование навыка работы со средой программирования Delphi при создании простой мультипликации графическими средствами, при воспроизведении звуков и видеоклипов.

Этапы выполнения работы:

1. Все студенты на занятии выполняют задание указанное ниже.
2. Отвечают на вопросы для самоконтроля и выполняют задания для самостоятельной работы.
3. На следующем занятии каждый студент отвечает на вопросы преподавателя и демонстрирует решение задач.

Примечание

Под *мультипликацией* можно понимать многократное перерисовывание изображения. Мультфильм представляет собой совокупность множества кадров, каждый из которых немного отличается от предыдущего, поэтому при быстром поочередном просмотре этих кадров возникает иллюзия движения.

Задание 1 Создайте форму, для свойства Caption которой задайте значение «Движущаяся окружность». Поместите на форму компонент Timer из палитры System.

<table>
<tr><td>1. Создание нового проекта</td><td>File/New/VCL Form Application</td></tr>
<tr><td colspan="2">2. Самостоятельно создать форму. Поместить компонент Timer. Для задания темпа смены кадров присвойте свойству Timerl.Interval значение 50 мс (0,05 с).</td></tr>
<tr><td>3. Для запуска анимации создайте обработчик события OnActivate, наступающего, когда форма получает или теряет фокус, например, при щелчке по ней мышью.</td><td>ObjectInspector/ Form1/ Events/ двойной щелчок на пустом поле списка в событии OnActivate
Добавьте в раздел описания переменных переменные целого типа x и y — координаты центра окружности и dx — приращение координаты x при движении окружности.
var
Form1: TForml:
x.y: byte; {координаты центра окружности}
dx: byte; {приращение координаты x при движении окружности}
Отредактируйте текст процедуры следующим образом:
procedure TForml.FormActivate(Sender: TObject);
begin
x:=0; {начальная позиция рисунка}
y:=30;
dx:=3; {шаг изменения координаты x}
Timerl.Interval:=50; {период возникновения события OnTimer - 0.5 сек}
Forml.Canvas.Brush.Color:=Forml.Color; {цвет окраски канвы}</td></tr>
</table>

	end;

4. Создайте процедуру обработчика события OnTimer	ObjectInspector/ **Timer1/ Events/** двойной щелчок на пустом поле списка в событии **Timer1Timer**
	
	В тело процедуры добавьте вызов процедуры рисования Ris
	Procedure TForm1.Timer1Timer(Sender: TObject);
	Begin
	Ris;
	End;
	Так как процедура должна быть описана до ее вызова, то выше этой процедуры в тексте программного модуля опишите процедуру рисования Ris:
	procedure Ris; {рисование}
	begin
	Forml.Canvas.Pen.Color:=Forml.Color; {задать перу цвет формы}
	Forml.Canvas.Ellipse(x, y, x+20, y+20); {стереть окружность – нарисовать ее цветом формы}
	x:=x+dx; {перейти в следующую позицию}
	Forml.Canvas.Pen.Color:=clBlack; {задать перу черный цвет}
	Forml.Canvas.Ellipse(x, y, x+20, y+20); {нарисовать окружность на новом месте}
	end;

5. Полный текст программного модуля будет выглядеть следующим образом:	unit main;
	interface
	uses
	Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls, StdCtrls;
	
	type
	TForml = class(TForm)
	Timerl: TTimer;
	procedure TimerlTimer(Sender: TObject);
	procedure FormActivate(Sender: TObject);
	private
	{ Private declarations }
	public
	{ Public declarations }
	End;
	
	var
	Forml: TForml;
	x.y: byte:
	{координаты центра окружности}
	dx: byte;
	{приращение координаты x при движении

	окружности}
	implementation
	{SR *.DFM}
	procedure Ris;
	
	begin
	Forml.Canvas.Pen.Color:=Fortnl.Color;
	Forml.Canvas.EllipseCx.y,x+20,y+20);
	x:=x+dx;
	Forml.Canvas.Pen.Color:=clBlack;
	Forml.Canvas.Ellipse(x,y,x+20,y+20);
	end;
	
	procedure TForml.TimerlTimer(Sender: TObject);
	begin
	Ris;
	end;
	
	procedure TForml.FormActivateCSender: TObject);
	begin
	x:=0; y:=30; dx:=3; Timerl.Interval:=50;
	Forml.Canvas.Brush.Color:=Forml.Color;
	end;
	end.

6. Сохранение изменений	Save All
7. Запустите приложение на исполнение	Run

Замечание: чтобы не было мерцания используется буферизация изображения – рисование очередного кадра производится не на холсте, а на канве невидимого компонента типа Bitmap, а затем готовый рисунок переносится на видимый холст методами Draw или CopyRect.

Задание 2 Создайте приложение-мультипликацию, в котором на фоне снежных гор летит самолет. В качестве значка приложения используйте рисунок самолета.

1. Создание нового проекта	File/New/VCL Form Application
2. Расположение окна приложения	Object Inspector/ Position /poScreenCenter
3. Оформление формы	Form1/ Caption/ «Полет самолета» Поместите на форму компоненты Timer и Image. Image1.Align/ alClient (компонент занимает всю клиентскую область контейнера и во время выполнения приложения его размеры изменяются при изменении размеров контейнера) Timer1Interval/ 100 мс (0,1 с) – темп смены кадров
4. Сохранение	Файл проекта – airplane Файл модуля – main
5. Описание	Back, bitmap, buf: TBitmap; {фон, картинка, буфер}

переменных добавить в раздел переменных модуля	BackRect, bitmapRect, BufRect: TRect; {область фона, картинки, буфера} x,y : integer; {координаты левого верхнего угла картинки} W, H: integer; {размеры картинки – ширина, высота}

6. Создание кода – обработчика события для запуска анимации	procedure TForml.FormActivate(Sender: TObject);
	begin
	Back := TBitmap.Create; {фон}
	bitmap := TBitmap.Create; {картинка}
	Buf := TBitmap.Create; {буфер}
	{загрузить и вывести фон из файла Backgraund.bmp}
	Back.LoadFromFile('Backgraund.bmp');
	Form1.Imagel.Canvas.Draw(0,0,Back);
	{загрузить изображение самолета, которое будет двигаться}
	bitmap.LoadFromFi1e('aplane.bmp');
	bitmap.Transparent := true;
	bitmap.TransParentColor := bitmap.Canvas.Pixels[1.1];
	{создать буфер для созранения копии области фона, на которую накладывается картинка}
	W:= bitmap.Width;
	H:= bitmap.Height;
	Buf.Width:= W;
	Buf.Height:=H;
	Buf.Palette:=Back.Palette; {обеспечить соответствие палитр}
	Buf.Canvas.CopyMode:=cmSrcCopy;
	BufRct:=Bounds(0,0,W,H);
	x:=-W;
	y:=20;
	BackRct:=Bounds(x.y,W,H); {определить сохраняемую область фона}
	Buf.Canvas.CopyRect(BufRct, Back.Canvas,BackRct); {сохранить ее} end;

7. Для получения эффекта мультипликации создайте процедуру обработчика события OnTimer следующим образом:	Выберите в окне Инспектора объектов объект Timerl и на странице Events произведите двойной щелчок на пустом поле списка в событии TimerlTimer.
	Отредактируйте тело процедуры обработчика события:
	
	procedure TForml.TimerlTimer(Sender: TObject);
	begin
	Forml.Imagel.Canvas.Draw(x, y, Buf); {восстановлением фона (из буфера) - удалить рисунок}
	x:=x+2; {смещение самолета по оси x}
	if x>Forml.Imagel.Width then X:=-W; {если самолет "вылетел" за пределы рисунка фона}
	BackRct:=Bounds(x, y, W, H); {определить сохраняемую область
	фона}

	Buf.Canvas.CopyRect(BufRct,Back.Canvas, BackRct); {сохранить ее копию} Forml.Imagel.Canvas.Draw(x,y,bitmap); {вывести рисунок самолета в новой позиции} end;
8. Для освобождения памяти от переменных, в которых хранились значения фона, рисунка самолета и буфера с фрагментом фона, создайте процедуру обработчика события закрытия окна приложения	Для этого выберите в окне Инспектора объектов объект Forml и на странице Events произведите двойной щелчок на пустом поле списка в событии OnClose. Вставьте в тело процедуры обработчика события вызов метода Free для освобождения памяти, динамически выделенной под объекты Back, bitmap и Buf. procedure TForml.FormClose(Sender: TObject; var Action: TCloseAction); begin Back.Free; {освободить память} bitmap.Free; Buf.Free; end;
9. Сохранение изменений	Save All
10. Запустите приложение на исполнение	F9

ВОСПРОИЗВЕДЕНИЕ ЗВУКА

Наиболее простой процедурой, управляющей звуком, является процедура **Beep**. Она не имеет параметров и воспроизводит стандартный звуковой сигнал, установленный в Windows, если компьютер имеет звуковую карту и задан стандартный сигнал (он устанавливается в Панели управления Windows после щелчка мышью по значку Звук). Если звуковой карты нет, или стандартный сигнал не установлен, звук воспроизводится через динамик компьютера в виде короткого щелчка.

Более серьезной функцией является **MessageBeep**. Она определена как

```
Function MessageBeep(uType:word):boolean;
```

Параметр uType указывает воспроизводимый звук как идентификатор раздела [sounds] реестра, в котором записаны звуки, сопровождающие те или иные события в Windows. С помощью приложения Звук в Панели управления Windows пользователь может удалить или установить соответствующие звуки. Для параметра uType определены следующие константы:

Значение	*Звук*
MB_ICONASTERISK	Звездочка
MB_ICONEXCLAMATION	Восклицание
MB_ICONHAND	Критическая ошибка
MB_ICONQUESTION	Вопрос
MB_OK	Стандартный звук

После инициализации воспроизведения звука функция MessageBeep возвращает управление в точку вызова и воспроизведение звука производится асинхронно. Если функция MessageBeep не находит указанный тип звука, она пытается воспроизвести стандартный звук. Если и он не установлен или если компьютер не снабжен звуковой картой, то звук воспроизводится через динамик компьютера.

При успешном завершении функция возвращает ненулевое значение (true). Если функция вернула нулевое значение, то получить информацию об ошибке можно с помощью вызова **GetLastError**.

Задание 3 Создайте приложение, которое позволит прослушать стандартные звуки Windows. Для выбора воспроизводимого звука используйте переключатели, размещенные в панели Radio-Group.

1. Создание нового проекта	File/New/VCL Form Application

2. Создание кода – обработчика события	procedure TForml.ButtonlClick(Sender: TObject);
	begin
	Case RadioGroupl.ItemIndex of {выбор варианта звука}
	0 : Beep; {стандартный звуковой сигнал}
	1 : MessageBeep(MB_ICONASTERISK); {звездочка}
	2 : MessageBeep(MB_ICONEXCLAMATION); {восклицание}
	3 : MessageBeep(MB_ICONHAND); {критическая ошибка}
	4 : MessageBeep(MB_ICONQUESTION); {вопрос}
	else MessageBeep(MB_OK); {стандартный звук Windows}
	end;
	end;

3. Сохранение изменений	Save All
4. Запустите приложение на исполнение	Run (F9), Alt+F4

Функция PlaySound

Функция **PlaySound** воспроизводит указанный волновой файл, или звук системного события, или звук из ресурса. Функция PlaySound определена следующим образом:

function PlaySound(pszSound:PChar; hmod:HINST; fdwSound:Cardinal) :boolean;

Параметр pszSound представляет собой строку с нулевым символом в конце и определяет воспроизводимый звук. В зависимости от значения флага fdwSound (SND_FILENAME, SND_ALIAS или SND_RESOURCE), параметр pszSound может определять имя волнового файла, псевдоним системного события или идентификатор ресурса. Если ни один из этих флагов не указан, то функция ищет в реестре Windows или в файле Win.ini указанное имя звука. Если звук найден, он воспроизводится, иначе – параметр pszSound интерпретируется как имя файла. Если параметр pszSound равен 0, то воспроизведение любого волнового файла прерывается. Для прерывания воспроизведения звука, не связанного с волновым файлом, следует указать SND_PURGE в параметре fdwSound.

Параметр hmod используется только при параметре fdwSound, равном – SND_RESOURSE. В этом случае hmod является дескриптором исполняемого файла, содержащего ресурс, который должен загружаться. Иначе hmod=0.

Задание 4 Создайте приложение, которое позволит прослушивать музыкальные файлы, открывая их с помощью OpenDialog

1. Создание приложения	Создайте форму и задайте для ее свойства Caption значение «Прослушивание звуковых файлов».
2. Изменение свойств компонентов формы	Поместите в любое место формы компонент OpenDialog из палитры Dialogs и задайте для свойства OpenDialog1.Title значение «Открыть звуковой файл», которое будет отображаться в заголовке диалогового окна открытия файла. Задайте для свойства OpenDialog1.Filter в поле FilterName значение «звуковые файлы *.wav», а в поле Filter – значение «*.wav». Поместите на форму компонент Button и задайте для его свойства Caption значение «Прослушать».
3. Создание кода обработчика события - щелчка мышью на кнопке Buttonl	TForml. Button1Click(Sender: TObject); Так как в процедуре обработки нажатия кнопки Button1 используется вызов функции PlaySound, то следует включить в раздел описания Uses вызов модуля mmSystem. Использование OpenDialog1 для открытия звукового файла реализуется при помощи оператора if OpenDialogl.Execute then PlaySound(PChar(OpenDialogl.FileNam e), 0, SND_ASYNC);

	где параметр SND_ASYNC указывает на асинхронное воспроизведение звука.
4. Сохраните файлы проекта и модуля в папке «Прослушивание звуковых файлов»	Save all
5. Откомпилируйте и запустите приложение.	Щелчком мышью на кнопке Послушать в окне приложения откройте диалоговое окно Открыть звуковой файл, в котором выберите звуковой файл, например в папке C:\Windows\Media

Примечание: функция PlaySound позволяет воспроизводить и системные звуки, просто называя их псевдонимы. Например, оператор PlaySound('SystemStart',0, SND_ASYNC); воспроизведет тот же звук, что и приведенный ранее оператор, указывавший имя файла и путь к нему. Оператор PlaySound('C:\Windows\ Media\ Звук Microsoft. wav', 0, SND_ASYNC or SND_LOOP); многократно асинхронно воспроизводит стандартный звук Microsoft, начиная воспроизведение снова и снова, как только оно заканчивается.

Примечание: если вы ввели в свое приложение подобный оператор, то следует предусмотреть какую-нибудь кнопку, по которой воспроизведение прерывается заданием нового звука или выполнением оператора PlaySound(0,0, SND_PURGE);.

ВИДЕОКЛИПЫ - КОМПОНЕНТ ANIMATE

Для воспроизведения стандартных видеоклипов Windows (типа копирования файлов, поиска файлов и т. п.) и не сопровождаемых звуком видеофайлов формата *.avi (Audio Video Interleaved) используется компонент типа **TAnimate.** Эти файлы представляют собой последовательность кадров битовых матриц. Они могут содержать и звуковую дорожку, но компонент **Animate** способен воспроизводить только немые клипы AVI. Он работает только с файлами AVI, не использующими сжатие, или с клипами AVI, сжатыми с использованием технологии RLE (run-length encoding).

Задание 5 Создайте приложение, которое позволит проигрывать стандартные клипы Windows и видеоклипы формата *.avi, открывая их с помощью OpenDialog.

1. Создание нового проекта	File/New/VCL Form Application
2. Изменение свойств компонентов формы	Form1/ Caption/ «Воспроизведение видеоклипов без звуков» Вверху формы разместить компонент Label1, свойство Caption очистить OpenDialog1.Title/ «Открыть файл видеоклипа» OpenDialog1.Filter/ «видеоклип (*.avi)|*.avi)» Поместите 3 кнопки Button/ задайте им имена - BtWind, BtStop, BtFile Задайте для свойств Caption этих кнопок значения «Клипы Windows», «Стоп», «Клип из файла»

	Animate1/ Visible/ False
3. Расположение окна приложения	Object Inspector/ Position /poScreenCenter

4. Создание обработчика щелчка на кнопке BtWind	Var
	Forml: TForml;
	I : byte;
	Отредактируйте текст процедуры TForml.BtWindClick следующим образом:
	procedure TForml.BtWindClick(Sender: TObject);
	begin
	Animatel.Visible:=True; {сделать видимой область анимации}
	I:=1; {указывает на первый клип Windows}
	Animatel.CommonAVI:=aviFindFolder; {воспроизвести первый стандартный клип - поиск в папке}
	Label1.Caption: = 'Клип Windows: поиск в папке';
	Animatel.Active:=True; {активизировать анимацию}
	End;

5. Создайте процедуру обработчика щелчка мышью на кнопке BtStop	procedure TForml.BtStopClick(Sender: TObject);
	begin
	Animatel.Stop; {вызов процедуры смены стандартного видеоклипа Windows}
	end;

6. Процедуру смены стандартного видеоклипа Windows реализуйте в виде процедуры обработчика события окончания воспроизведения видеоклипа **OnStop**	procedure TForml.AnimatelStop(Sender: Tobject);
	begin
	Inc(I); {увеличить счетчик клипов}
	with Animatel do
	begin
	case I of
	2: begin
	CommonAVI:=aviFindFi1e;
	Labell.Caption:='Клип Windows: поиск файла';
	end;
	3: begin
	CommonAVI:=avi FindComputer;
	Label 1.Caption: = 'Клип Windows: поиск компьютера';
	end;
	4: begin
	CommonAVI:=avi CopyFi1es:
	Label 1.Caption: = 'Клип Windows: копирование файлов';
	end;
	5: begin
	CommonAVI:=aviCopyFile;
	Labell.Caption: = 'Клип Windows: копирование файла';
	End;

	6: begin
	CommonAVI:=aviRecycleFile;
	Label 1.Caption: = 'Клип Windows: удаление файла в корзину';
	end;
	7: begin
	CommonAVI:=aviEmptyRecycle;
	Labell.Caption: = 'Клип Windows: очистка корзины';
	end;
	8: begin
	CommonAVI:=aviDeleteFile;
	Labell.Caption:='Клип Windows: удаление файла;
	end;
	end;
	if I<9 then Active:=True
	else
	begin
	Visible:=False;
	Label1.Caption:=''; {очистить поле показа клипов}
	end;
	end;
	end;

7. Для воспроизведения видеоклипов из файлов формата *.avi создайте процедуру обработчика щелчка мышью на кнопке **BtFile**	procedure TForml.BtFi1eClick(Sender: TObject);
	var
	FName : string; {переменная для хранения части имени файла}
	begin
	if OpenDialogl.Execute then
	with Animatel do
	begin
	I:=9;
	Fi1eName:=OpenDialog1.FileName;
	FName:=FileName;
	repeat {удалить часть имени до '\'}
	Delete(FName, l, Pos('\', FName));
	until Pos('\', FName)=0;
	Visible:=True;
	Label1.Caption:='Клип из файла '+FName;
	Active:=True;
	end;
	end;

8. Сохранение изменений	Save All
9. Запустите приложение на исполнение	F9

Универсальный проигрыватель аудио- и видеоинформации MediaPlayer

Для проигрывания аудио и видеоинформации в Delphi (начиная с Delphi 2) имеется универсальный проигрыватель аудио- и видеофайлов — MediaPlayer. Этот компонент по умолчанию входит в состав палитры System и инкапсулирует интерфейс управления носителями (Media Control Interface, MCI) Windows 95 и Windows NT. MediaPlayer имеет ряд кнопок, управляющих такими устройствами мультимедиа, как CD-ROM и др., управляемых мышью.

Кнопка	***Значение***	***Действие***
Play	btPlay	Воспроизведение
Pause	btPause	Пауза воспроизведения или записи (если проигрыватель в момент щелчка уже находится в состоянии паузы, то воспроизведение или запись возобновляется)
Stop	btStop	Останов воспроизведения или записи
Next	btNext	Переход на следующий трек или на конец записи
Prev	btPrev	Переход на предыдущий трек или на начало записи
Step	btStep	Перемещение вперед на заданное число кадров
Back	btBack	Перемещение назад на заданное число кадров
Record	btRecord	Начало записи
Eject	btEject	Освобождение объекта, загруженного в устройство

Проигрыватель может управляться как кнопками, так и непосредственно при помощи соответствующих этим кнопкам методов **(Play, Pause, Stop, Next, Previous, Step, Back, StartRecording, Eject).** В этом случае сам компонент **TMedia-Player** может быть сделан невидимым.

Тип устройства мультимедиа **(dtWaveAudio** или **dtVideodisc)** определяется свойством **DeviceType.** Если устройство хранит объект воспроизведения в файле, то имя файла задается свойством **FileName.** Если свойство **DeviceType** задано равным **dtAutoSelect,** то проигрыватель пытается определить тип устройства, исходя из расширения имени файла **FileName.** Если для свойства **AutoOpen** установлено значение **true,** то проигрыватель пытается открыть устройство, указанное свойством **DeviceType,** автоматически во время своего создания в процессе выполнения приложения.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Что такое анимация? Какими способами можно создать в приложении Delphi анимированное изображение?
2. Каково отличие звуковых файлов форматов *.wav и *.midi?
3. Сравните возможности функций **Beep, MessageBeep** и **PlaySound.**
4. Сравните назначение и возможности компонентов **Animate** и **MediaPlayer.**

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Создайте приложение, в окне которого отображается анимация — движение мяча, брошенного под углом к горизонту.
2. Создайте приложение-мультипликацию, в котором на фоне дороги перемещается автомобиль (вариант: на фоне звездного неба перемещается космический корабль).

3. Создайте приложение, которое позволит прослушивать звуковые файлы формата *.wav, открывая их с помощью диалогового окна **Открыть звуковой файл,** вызываемого командой **Файл ► Открыть.** Для завершения работы приложения используйте команду **Выход** в главном меню.