МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОУ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

СЕВЕРО-КАВКАЗСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ

Е.Х.Бежанова

ОБЪЕКТНО – ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Лабораторный практикум для обучающихся 2 курса направления подготовки 01.03.04 Прикладная математика

Черкесск
2018

УДК 004.4
ББК 32.973-018
Б38

Рассмотрено на заседании кафедры «Математика»
Протокол № 2 от « 21 » сентября 2018 г.
Рекомендовано к изданию редакционно-издательским советом СевКавГА.
Протокол № 15 от «30» октября 2018 г.

Рецензенты: Кочкаров А.М. – д.ф-м.н., проф. кафедры математики
Темирова Л.Г. – к.ф-м.н., доц. кафедры математики

Б38 **Бежанова, Е.Х.** Объектно – ориентированное программирование: Лабораторный практикум для обучающихся 2 курса направления подготовки 01.03.04 Прикладная математика / Е.Х.Бежанова – Черкесск: БИЦ СевКавГА, 2018. – 48 с.

Лабораторный практикум содержат теоретический материал для выполнения лабораторных работ, перечень индивидуальных заданий к лабораторным работам, теоретические вопросы по закреплению материала.

УДК 004.4
ББК 32.973-018

Содержание

Лабораторная работа № 1
Знакомство с визуальной средой Lazarus, создание простейшего приложения

Цель: Изучение среды Lazarus, освоение основных приемов работы в среде. Изучение основных компонентов. Работа со свойствами компонентов. Проектирование простейших обработчиков событий.

Рабочее задание: Провести исследование рабочей среды Lazarus и основных элементов управления. Создать простейшее приложение в среде Lazarus, в котором пользователь вводит произвольный текст и при щелчке на командной кнопке выводит этот текст на экран, добавляя перед ним постоянную фразу “Привет! Моя первая программа”. Пользователь должен иметь возможность управления цветом и параметрами шрифта отображаемого текста.

Пример выполнения работы

1. Создать на диске папку для лабораторной работой № 1, присвоив этой папке имя Lab1.
2. Войти в среду Lazarus. Будет автоматически создан новый проект с единственной (пустой) формой. Этот проект уже готов к выполнению. Запустить его, щелкнув кнопку Запуск Запустить (F9). Легко убедиться, что появившееся на экране окно-форма обладает всеми свойствами окна Windows: его можно перемещать по экрану, изменять размеры, сворачивать в значок и разворачивать на весь экран. Окно имеет стандартные интерфейсные элементы: заголовок, оконное меню, кнопки управления размером и кнопку закрытия. Однако на этом возможности окна и исчерпываются. Это естественно, ведь в проекте еще ничего, кроме формы не создано.
3. Изменить значение свойства Name (имя формы) Form1 в соответствии с изложенными выше требованиями. Например, L1_Ivanov_Form1. В свойство Caption занести текст, который будет показываться в заголовке формы при выполнении приложения. Например, “Лаб 1. Выполнил Иванов И.И., группа ПМ-171”.
4. Свойству BorderStyle формы (стиль рамки окна) присвоить значение bsDialog. Это значение определяет окно как диалоговое, его размеры на этапе прогона приложения (в процессе работы приложения) не могут быть изменены.
5. Поместить на форму компонент Button1 (командная кнопка) и написать на ней “Вывод текста” (т.е. занести этот текст в свойство Caption).
6. Аналогично создать кнопку Button2 и написать на ней “Запуск формы”.

7. Поместить на форму компонент BitBtn1 из палитры Additional. Для свойства Kind выбрать из предлагаемого перечня значение bkClose. При этом автоматически на кнопку помещается надпись “Закрыть” и добавляется значок «косой крестик».
8. Поместить на форму компонент Memo1 (многострочный редактор). Двойным щелчком на свойстве Lines вызвать окно «Диалог ввода строк» (String List Editor) и удалить из него все строки.
9. Поместить на форму компонент GroupBox и в свойстве Caption написать “Стиль шрифта”. Это компонент-контейнер.
10. Разместить в контейнере GroupBox четыре компонента CheckBox1, …, CheckBox4, подписав их как “Полужирный”, ”Курсивный”, ”Подчеркнутый” и “Перечеркнутый”.
11. Поместить на форму компонент Edit1 и очистить его свойство Text.
12. Слева от компонента Edit1 поместить компонент Label1 с подписью (свойство Caption) “Введите текст”.
13. Поместить на форму компонент RadioGroup и в свойстве Caption этого компонента написать “Цвет шрифта”. Это компонент-контейнер.
14. Разместить в контейнере RadioGroup пять компонентов-радиокнопок RadioButton1, …, RadioButton5, подписав их как “Красный”, ”Синий”, ”Зеленый”, “Желтый” и “Черный”. В соответствии с этими подписями установить значение свойства Color (цвет фона) каждого из этих компонентов: clRed, clBlue, clGreen, clYellow, clBlack. Чтобы на этом фоне смотрелись надписи, для компонентов синего, зеленого и черного цвета изменить цвет символов (подсвойство Color сложного свойства Font) на белый (clWhite). Особенностью контейнера RadioGroup является то, что только один из размещенных на нем компонентов RadioButton может быть включенным. При включении любого из них все остальные автоматически выключаются.
15. Используя команду Файл□Создать форму создать новую форму Form2, определив ее имя (свойство Name) в соответствии с требованиями, изложенными ранее (в разделе 1). В свойство Caption этого компонента занести текст “О программе”. Цветовое оформление, стиль и т.п. выбрать по своему усмотрению. С помощью компонента Shape из панели Additional можно создать различные элементы оформления. Например, на рис. 1. с помощью этого компонента создан эллипс (свойство Shape установлен в stEllipse).
16. Поместить на новую форму компонент Button1 (кнопка). На кнопке написать «Скрыть форму».
17. Поместить на форму Form2 компонент Image1 (отображение картинок) из палитры Additional и загрузить в него соответствующий тематике рисунок (можно загрузить файл lab1.jpg, прилагающийся к электронному варианту рабочего задания). Загрузка рисунка производится при двойном щелчке на свойстве Picture. Можно также щелкнуть на кнопке с многоточием. В обоих случаях откроется окно

Диалог загрузки изображения (Picture Editor), с помощью которого и загружается изображение. Если изображение не помещается в отведенную область компонента Image1, то свойству Stretch необходимо присвоить значение true, т.е. «подгонка» рисунка под отведенную область.

18. В результате проведенных операций должно получиться две формы примерно такого вида, как приведено на рис.6 (показан вид, которые формы будут иметь на этапе выполнения). Формы не должны обязательно иметь представленный выше вид, но общая идея лабораторной работы, т.е. знакомство с основными компонентами и их свойствами должна быть соблюдена.

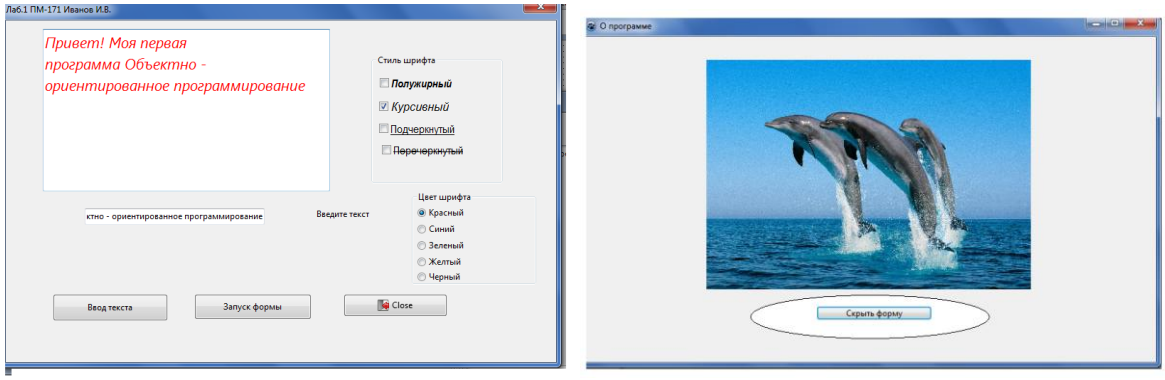


Рисунок 1 – Вид форм для лабораторной работы.

Запустить приложение на выполнение. Несмотря на наличие в форме всевозможных элементов управления, они не выполняют никаких действий. Например, можно выбрать любой стиль шрифта или цвет, но этот выбор нигде не отразится. Это естественно: в проекте еще не разработан ни один обработчик события, поэтому ни одно событие не обрабатывается, т.е. приложение не реагирует еще ни на одно событие. Форма Form2 вообще никогда не сможет появиться на экране, несмотря на то, что она зарегистрирована в проекте. Наличие командной кнопки “Запуск формы” еще не обеспечивает ее показ на экране. Необходим соответствующий программный код, который бы в ответ на щелчок на этой кнопке показывал форму. Работает только один единственный компонент – командная кнопка “Закрыть”, так как в этом компоненте автоматически генерируется соответствующий код для завершения выполнения приложения. С помощью этой кнопки закрыть приложение.

20. Сделать первую форму активной. Щелчком на компоненте Button2 (командная кнопка “Запуск формы”) выбрать (активизировать) его. Кстати, выбрать компонент можно и из списка компонентов в верхней части Инспектора объектов. В окне Инспектора объектов (Object Inspector) отображаются свойства и события выбранного компонента, т.е. при активизации Button2 будут показаны свойства и события именно этого объекта. В окне Object Inspector перейти на вкладку События (Events). Найти строку с именем события OnClick (щелчок). Двойной щелчок в

правой колонке этой строки переводит в окно ввода кода обработчика этого события. Кстати, для объекта типа TButton событие OnClick является событием по умолчанию, код которого раскрывается двойным щелчком на самом объекте. Если обработчик события еще не разработан, Lazarus создает заготовку для его создания. Создать следующую процедуру обработки этого события (эта процедура показывает форму Form2, делая ее видимой путем установки в true свойства Vsible):

```
procedure TL1_Ivanov_Form1.Button2Click (Sender: TObject);
begin
L1_Ivanov_Form2.Visible:=True;
end;
```

В этой процедуре делается ссылка на форму Form2. В модуле формы Form1 такой компонент неизвестен. Поэтому в таком виде проект не откомпилируется. Необходимо в модуль формы Form1 добавить ссылку на модуль Unit2. Однако перед этим модулю должно быть присвоено имя в соответствии с разделом 1: L1_Ivanov_Unit2. Для этого модуль необходимо сохранить в папке размещения проекта, указав при сохранении требуемое имя. Указанное имя будет занесено и в первую строку модуля. Сама ссылка на модуль Unit2 задается путем добавления в объектном коде первой формы в разделе реализации модуля implementation перед директивой {$R *.LFM} команды присоединения второго модуля:

```
uses L1_Ivanov_Unit2;
```

21. Сделать активной вторую форму (Form2). В окне Инспектора объектов перейти на вкладку События. Из списка компонентов, установленных на форме выбрать Button1. Найти событие OnClick. Двойным щелчком по этому событию перейти в окно ввода кода. Создать процедуру обработки этого события, закрывающую (делающую невидимой) форму Form2:

```
procedure TL1_Ivanov_Form2.Button1Click (Sender: TObject);
begin
L1_Ivanov_Form2.Visible:=False;
end;
```

22. Запустить приложение на выполнение. Теперь при щелчке на командной кнопке Button2 (“Запуск формы”) на экране появится форма Form2, т.к. событие Щелчок на этой кнопке будет обработано и этот обработчик покажет форму (сделает ее видимой). Аналогично щелчок на соответствующей кнопке формы Form2 уберет с экрана эту форму (сделает ее невидимой). Закрыть приложение.

23. Активизировать форму Form1 непосредственно. В окне Инспектора объектов (Object Inspector) найти событие OnActivate, наступающее в

момент активации формы. Двойным щелчком на этом событии перейти в окно кода, где создать процедуру, которая при запуске проекта на выполнение будет выделять RadioButton5 (т.е. «включать» черный цвет):

```
procedure TL1_Ivanov_Form1.FormActivate (Sender: TObject);
begin
RadioButton5.Checked:=True;
end;
```

24. Сделать окно первой формы активным. Щелчком на компоненте Button1 (командная кнопка “Вывод текста”) выбрать (активизировать) его. В окне Инспектора объектов перейти на вкладку События. Найти строку с именем события OnClick (Щелчок). Двойной щелчок в правой колонке этой строки (события) переводит в окно ввода кода обработчика этого события. Создать следующую процедуру обработки этого события:

```
procedure TL1_Ivanov_Form1.Button1Click(Sender: TObject);
var i_str,str:string;i_int:integer;k:TColor;
begin
// Очищаем многострочный редактор.
Memo1.Lines.Clear;
// Определяем, какая радиокнопка нажата, и в зависимости
// от этого выбираем цвет для отображения текста в
// редакторе. Выбранный цвет присваиваем свойству
// Font.Color объекта Memo1.
if RadioButton1.Checked then k:=clRed;
if RadioButton2.Checked then k:=clBlue;
if RadioButton3.Checked then k:=clGreen;
if RadioButton4.Checked then k:=clYellow;
if RadioButton5.Checked then k:=clBlack;
Memo1.Font.Color:=k;
// Формируем текст для вывода в многострочном редакторе.
// Текст компонуется из двух частей: постоянный текст
// (константа) и текст, введенный в поле однострочного
// редактора Edit1.
str:='Привет! Моя первая программа '+Edit1.Text;
// Запрашиваем размер шрифта и присваиваем его свойству
// Font.Size объекта Memo1.
i_str:= InputBox ('Блок ввода', 'Введите размер
шрифта','10');
i_int:=StrToInt(i_str);
Memo1.Font.Size:=i_int;
// Формируем стиль шрифта в зависимости от установок
// соответствующих переключателей на форме.
Memo1.Font.Style:=[];
```

```
if CheckBox1.State = cbChecked then
Memo1.Font.Style:=[fsBold];
if CheckBox2.State = cbChecked then
Memo1.Font.Style:=Memo1.Font.Style+[fsItalic];
if CheckBox3.State = cbChecked then
Memo1.Font.Style:=Memo1.Font.Style+[fsUnderline];
if CheckBox4.State = cbChecked then
Memo1.Font.Style:=Memo1.Font.Style+[fsStrikeOut];
// Помещаем скомпонованную строку в многострочный
// текстовый редактор Memo1.
Memo1.Lines.Add(str);
end;
```

25. Нажав клавишу F9, запустить приложение на выполнение. Объяснить логику функционирования приложения. В случае необходимости выполнить отладку.
26. Сохранить форму и сам проект в папке размещения проекта, присвоив им имена, как указано в разделе 1.
27. По результатам работы составить отчет.

Лабораторная работа № 2

Условные операторы. Вычисление значения функции, заданной условно.

Цель: Создание простого законченного приложения. Знакомство с возможностями Lazarus. Проектирование разветвляющегося вычислительного процесса.

Рабочее задание: Создать приложение для вычисления и вывода на экран значения функции:

$$y=\begin{cases} f_1(x), \text{если} & x \le 0 \\ f_2(x), \text{если} & 0 < x \le a \\ f_3(x), \text{если} & x > a \end{cases}$$

Создать приложение для вычисления значения функции:

$$y=\begin{cases} \sin(x^2+ax), & \text{если} \quad x \le 0 \\ 1-\dfrac{1+\sqrt{(x^2+ax)}}{e^{\sin(x)}(1+x)}, & \text{если} \quad 0 < x \le a \\ \dfrac{\cos(x^2-a^2)}{\sqrt{1-\sin(a-x)}}-\dfrac{1-\sin(a-x)}{e^{\sin(x)}}, & \text{если} \quad x > a \end{cases}.$$

1. Создать на диске папку для лабораторной работой № 2, присвоив этой папке имя Lab2.
2. Войти в среду Lazarus. Создать новый проект. Присвоить соответствии с требованиями, изложенными в разделе 1.
3. Используя палитру компонентов (элементов управления), спроектировать форму Form1, изменить значение свойства Name в соответствии с требованиями, изложенными в разделе 1 (L2_<ФИО>_Forma1). Для свойства BorderStyle лучше установить bsSingle. Это значение запрещает изменять с помощью мыши размеры формы в процессе выполнения приложения. Для BorderIcons biSystemMenu установить True, а для остальных трех свойств из раздела BorderIcons установить False. Такие установки оставляют в заголовке окна приложения только кнопку закрытия, убирая остальные, не позволяя тем самым изменять размеры формы с помощью кнопок в заголовке окна. Таким образом, во время работы приложения размеры формы изменить нельзя никаким способом. Примерный вид формы показан на рисунке 2.

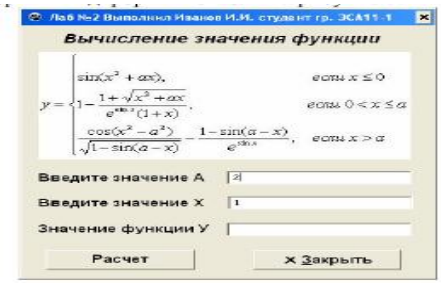


Рисунок 2 – Примерный вид проектируемой формы

4. Поместить на форму компонент Label, присвоив ему имя (свойство Name) Label0. В свойство Caption занести текст «Вычисление значения функции».
5. Поместить на форму компонент Image1 из палитры Additional и загрузить в него заранее подготовленный рисунок, отображающий функцию. Загрузка рисунка производится с помощью стандартного окна диалога, открываемого при щелчке на кнопке с многоточием в свойстве Picture. Подготовить такой рисунок можно, используя редактор формул Microsoft Equation из состава Microsoft Word и любой графический редактор (например, Photoshop или Paint). Вначале набирается формула в редакторе формул, а затем при помощи буфера обмена она помещается в графический редактор, после чего файл рисунка необходимо сохранить в папку, отведенную для текущего проекта.
6. Поместить на форму три компонента типа TEdit: Edit1, Edit2, Edit3. Первые два предназначены для ввода соответственно значения параметра A, и значения переменной X. В свойства Text этих компонентов ввести какие-либо значения – значения по умолчанию. Эти значения будут показываться

при запуске приложения на выполнение. При выполнении приложения их можно будет заменить другими. Компонент Edit3 будет использован для вывода результата вычисления функции. Поэтому необходимо запретить ввод данных в него пользователем. Для этого свойству ReadOnly (только чтение) присвоить значение true, запрещающее пользователю заносить в компонент какие-либо данные. Можно поступить другим способом: свойству Enabled этого компонента присвоить значение false, что сделает компонент недоступным для пользователя. При этом, правда, значение будет показываться более бледным цветом.

7. Поместить на форму три компонента типа TLabel: Label1, Label2, Label3 для подписей к полям ввода. В поле свойства Caption написать "Введите значение A", "Введите значение X" и "Значение функции Y" соответственно. При этом свойство Color лучше установить таким же, как и у формы. Параметры шрифта (сложное свойство Font) установить по своему усмотрению.

8. Включить в форму две командные кнопки: элемент управления типа TButton с именем Button1 (свойство Name), и кнопку типа TBitBtn из палитры Additional с именем BitBtn1. Написать (в свойстве Caption) на первой кнопке «Расчет». Для свойства Kind второй кнопки в окне Инспектор объектов установить значение bkClose. Подобрать удобный для просмотра размер шрифта (свойство Font□Size).

9. Щелчком на кнопке «Переключить форму/модуль» (F12) на панели инструментов главного окна интегрированной среды перейти к окну для набора программного кода модуля Unit1. В разделе реализации модуля implementation сразу после директивы {$R *LFM} задать свою функцию, например:

```
function f(x,a:real):real;
begin
  if x <= 0 then
    f:= sin(x*x + a*x)
  else if x<=a then
    f:=1-(1 + sqrt(x*x+a*x))/(exp(sin(x))*(1+x))
  else
    f:=cos(x*x-a*a)/sqrt(1-sin(a-x))-
      (1-sin(a-x))/exp(sin(x));
end;
```

10. Поскольку проект рассчитан на многократный расчет значения функции, то при вводе новых значений параметра X (или A) результат предыдущего расчета (У) уже не будет соответствовать исходным данным. Поэтому необходимо скрывать «старый» ответ. Для этого необходимо с событиями OnEnter (Получение фокуса) для однострочных окон

редактирования Edit1 и Edit2 связать программные коды, которые делают невидимым однострочное текстовое поле (Edit3):

```
procedure TL2_Ivanov_Form1.Edit1Enter(Sender: TObject);
begin
  Edit3.Visible:=False;
end;

procedure TL2_Ivanov_Form1.Edit2Enter(Sender: TObject);
begin
  Edit3.Visible:=False;
end;
```

11. Также необходимо проверять корректность входных данных, т.е. их соответствие числовому представлению исходных значений. Для этого добавляем код, обрабатывающий события OnExit (Потеря фокуса) для однострочных окон редактирования Edit1 и Edit2:

```
procedure TL2_Ivanov_Form1.Edit1Exit(Sender: TObject);
begin
  if Edit1.Text<>'' then
  try
    StrToFloat(Edit1.Text);
  except
    on EConvertError do
      begin
        ShowMessage('Введено неверное значение A');
        Edit1.SetFocus;
      end;
  end;
end;

procedure TL2_Ivanov_Form1.Edit2Exit(Sender: TObject);
begin
  if Edit2.Text<>'' then
  try
    StrToFloat(Edit2.Text);
  except
    on EConvertError do
      begin
        ShowMessage('Введено неверное значение X');
        Edit2.SetFocus;
      end;
  end;
```

```
end;
```

12. С событием OnClick кнопки Button1 связать программный код, который вычисляет значение функции и выводит его в поле однострочного редактора Edit3:

```
procedure TL2_Ivanov_Form1.Button1Click(Sender: TObject);
begin
  Edit3.Visible:=True;
  Edit3.Text:=FloatToStr(f(StrToFloat(Edit2.Text),
          StrToFloat(Edit1.Text)));
end;
```

13. Нажав клавишу F9, запустить приложение на выполнение. Объяснить логику функционирования приложения. В случае необходимости выполнить отладку.
14. Сохранить отлаженный проект в папке проекта.
15. По результатам работы составить отчет.

Математические функции

Функция	**Описание**	**Аргумент**
Abs(X)	абсолютное значение	целое или действительное выражение
Ceil(X)	округление до наименьшего целого, превышающего или равного аргументу	действительное выражение
Compare Value (A, B)	сравнение двух значений; возвращает —1, 0 или +1, если A соответственно меньше, равно или больше B	целые или действительные выражения
DivMod (Dividend, Divisor, Result, Remainder)	целочисленное деление: **Result** — результат, **Remainder** — остаток	целые выражения
EnsureRange (AValue, AMin, AMax)	возвращает ближайшее к **AValue** в диапазоне **AMin** — **AMax**	целые или действительные выражения
Exp(X)	экспонента	действительное выражение
Floor(X)	округление до наибольшего целого, меньшего или равного аргументу	действительное выражение
Frac(X)	дробная часть аргумента: ***X-Int(X)***	действительное выражение
Frexp(X, Mantissa,	выделяет мантиссу и показатель степени 2:	действительное выражение

Функция	Описание	Аргумент
Exponent)	$X = Mantissa * 2^{Exponent}$	
InRange (AValue, AMin, AMax)	определяет, лежит ли **AValue** в диапазоне **AMin** – **AMax**	целые или действительные выражения
Int(X)	целая часть аргумента	действительное выражение
IntPower (X, E)	возведение **X** в целую степень **E**: X^E	действительное и целое выражения
IsInfinite(X)	определяет, не равен ли аргумент бесконечности	действительное выражение
IsNan(X)	определяет, не равен ли аргумент **NaN** - нечисловой величине	действительное выражение
IsZero(X, Epsilon)	определяет, не отличается ли аргумент от нуля менее чем на **Epsilon**	целые или действительные выражения
Ldexp(X,P)	умножение **X** на 2 в целой степени **P**: $X*2^P$	действительное и целое выражения
Ln(X)	натуральный логарифм от **X**	действительное выражение
LnXPl(X)	натуральный логарифм от **X+1**	действительное выражение
Logl0(X)	десятичный логарифм от **X**	действительное выражение
Log2(X)	логарифм от **X** по основанию **2**	действительное выражение
LogN (N, X)	логарифм от **X** по основанию **N**	действительное выражение
Max(A,B)	максимум двух чисел	действительное или целое выражение
Min(A,B)	минимум двух чисел	действительное или целое выражение
Pi	число Пи: **3.1415926535897932385**	–
Poly(X, C)	вычисляет полином **X** с массивом коэффициентов **C**	действительные выражение и массив
Power(X, E)	возведение **X** в произвольную степень **E**: X^E	действительное выражение
Round(X)	ближайшее целое аргумента	действительное выражение
RoundTo (AValue,	округляет действительное число до заданного десятичного порядка	действительные и целые выражения

Функция	Описание	Аргумент
ADigit)		
SameValue (A, B, Epsilon)	сравнивает **A** и **B** с точностью до **Epsilon**	действительные выражения
Sign(X)	определяет знак аргумента	действительные и целые выражения
SimpleRoundTo (A Value, ADigit)	округляет действительное число до заданного десятичного порядка	действительные и целые выражения
Sqr(X)	квадрат аргумента: **X * X**	действительное выражение
Sqrt(X)	квадратный корень	действительное выражение
Trunc(X)	возвращает целую часть действительного выражения	действительное выражение

Комментарий:

Математические функции описаны в модуле Math. Этот модуль должен быть подключен к приложению оператором uses. Функции Ln, LnXPl, Logl0, Log2, LogN вычисляют логарифмы по различным основаниям. Если аргумент отрицательный, генерируется исключение EInvalidOp.
Функция Sqrt при извлечении корня из отрицательного числа возвращает значение NaN.

Тригонометрические и гиперболические функции

Функция	Описание	Модуль
ArcCos(X)	арккосинус	*Math*
ArcCosh(X)	арккосинус гиперболический	*Math*
ArcCot(X)	арккотангенс	*Math*
ArcCotH(X)	арккотангенс гиперболический	*Math*
ArcCsc(X)	арккосеканс	*Math*
ArcCsc(X)	арккосеканс гиперболический	*Math*
ArcSec(X)	арксеканс	*Math*
ArcSecH(X)	арксеканс гиперболический	*Math*
ArcSin(X)	арксинус	*Math*

Функция	Описание	Модуль
ArcSinh(X)	арксинус гиперболический	*Math*
ArcTan(X)	арктангенс	*System*
ArcTan2(Y, X)	арктангенс от **Y / X**	*Math*
ArcTanh(X)	арктангенс гиперболический	*Math*
Cos(X)	косинус	*System, Math*
Cosecant(X)	косеканс	*Math*
Cosh(X)	косинус гиперболический	*Math*
Cot(X)	котангенс	*Math*
Cotan(X)	котангенс	*Math*
CotH(X)	котангенс гиперболический	*Math*
Csc(X)	косеканс	*Math*
CscH(X)	косеканс гиперболический	*Math*
Hypot(X, Y)	вычисление гипотенузы по заданным катетам **X** и **Y**	*Math*
Sec(X)	секанс	*Math*
Secant(X)	секанс	*Math*
SecH(X)	секанс гиперболический	*Math*
Sin(X)	синус	*System, Math*
SinCos(X, S, C)	синус и косинус	*Math*
Sinh(X)	синус гиперболический	*Math*
Tan(X)	тангенс	*Math*
Tanh(X)	тангенс гиперболический	*Math*

Комментарии:

Во всех функциях тип аргумента и тип возвращаемого значения - Extended. Во всех тригонометрических функциях аргумент X — угол в радианах. Обратные тригонометрические функции возвращают главное значение в радианах. В функциях ArcSin и ArcCos аргумент должен лежать в пределах от -1 до 1. Функции ArcSin и ArcTan возвращают результат в пределах [-Pi/2 .. Pi/2], ArcCos — в пределах [0 .. Pi].

Индивидуальные задания:

Вариант	$f_1(x)$	$f_2(x)$	$f_3(x)$
1	$tg(2ax)$	$\sin(3x+\sqrt[3,5]{\lvert a\rvert})$	$\cos(x-2a)$
2	$5ax+2$	$5/\left(x+0,5\lvert a\rvert^{1,5}\right)$	$0,5/\sin(ax+1)$
3	$\sqrt[3]{\lvert x-1\rvert}-a$	$x^4/7+a$	$\sin^3(2ax+\pi)$
4	$\sqrt[3]{\sin^2(ax)+\cos^4(ax)}$	$ctg(ax+0,4)$	$\ln(2x+0,5a)$

5	$ax^3 - \ln\lvert x-1\rvert$	$\ln^3(ax+4)$	$x^4 - x^{2-a}$
6	$\sin(x^2 + a^3)$	$e^{-x} + 1/ax$	$\ln(ax^3 + x^2)$
7	$(3ax - 1)/x^5$	$\ln^2\lvert\sqrt{x+5a}\rvert$	$\sqrt{1+(ax)^{2,5}}$
8	$a^2 ch(x) + x$	$1/(arctg(2x)+a)$	$ax^2 e^{-x}$
9	$\lvert ax-1\rvert^{x^2} \sin(3x)$	$sh^2(x) + ax$	$\arcsin(ax) + x^{0,25}$
10	$th(x) + x^{\sin(ax)}$	$\sin(a/(2x^2+1))$	$\ln^2(x^2) + \sqrt{x^{1,25}}$
11	$ax + 1/(ax+2)$	$\left(\sqrt{\lvert x^3 + a\rvert}\right)\cos(x+1)$	$x^2 + \sin(5ax)$
12	$(a+5)\sin(4x)$	$\sqrt[5]{6x - x^2 + a^2}$	$\sin(\pi a/x)$
13	$x^2 + \arcsin(ax) - 1$	$\sin(ax+\pi) - \cos^2(x)$	$ax^2 - 5x$

Лабораторная работа № 3

Циклический алгоритм. Табулирование функции и поиск экстремумов.

Цель: Изучение операторов цикла. Создание циклических алгоритмов. Приобретение навыков создания проекта, работы с несколькими формами.

Рабочее задание: Создать приложение для вывода на экран в отдельную форму таблицы значений функции, заданной в лабораторной работе № 2. Диапазон и шаг изменения аргумента задавать при выполнении приложения. Определить максимальное и минимальное значения в заданном диапазоне методом перебора. При создании проекта использовать проект, созданный в лабораторной работе №2, не уничтожая последнего.

Пример выполнения работы

1. На диске создать папку для проекта лабораторной работы №3. Присвоить ей имя Lab3.
2. Создаваемый проект рассчитывает таблицу значений функции, которая была задана в лабораторной работе № 2. Чтобы заново не создавать на форме необходимые компоненты – откроем предыдущий проект, выделим все размещенные на нем компоненты и скопируем их (например, Ctrl+C). После этого закроем проект второй лабораторной работы и создадим новый (Проект□Создать проект), на форму которого поместим скопированные компоненты.
3. Изменить имя формы применительно к лабораторной работе № 3. Например, L3_Ivanov_Form1. Создать соответствующую подпись в строке заголовка формы (свойство Caption).

4. В свойство Caption компонента Label0 занести новый текст, соответствующий новой работе “Табулирование функции”.
5. Внести изменения в исходную форму и привести ее к виду, показанному на рис.8. Для этого поместить на форму (добавить) однострочное окно редактирования с именем Edit4 и метку Label4. Надписи в метках изменить в соответствии с рис. 8. Свойству ReadOnly однострочного окна редактирования Edit3 вернуть значение true. Очистить свойства Text всех однострочных окон редактирования, а лучше ввести в них числовые значения – значения по умолчанию.

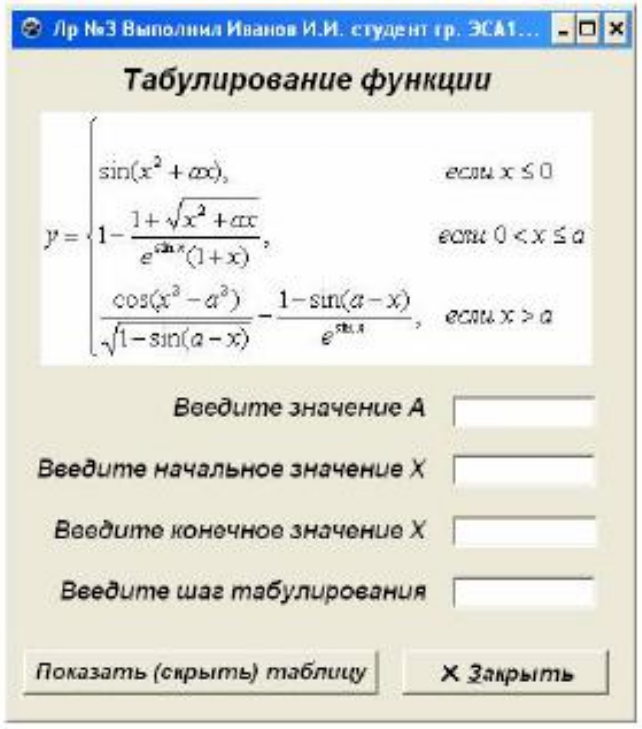


Рисунок 3– Примерный вид проектируемой формы.

6. В Элементе управления Button1 изменить значение свойства Caption на “Показать (скрыть) таблицу”. Подобрать удобный для просмотра размер шрифта (Font□Size).
7. Компонент Image из палитры Additional с загруженным в него рисунком, отображающим исследуемую функцию, не меняется.
8. Создать новую форму Файл□Создавать форму. Дать ей имя в соответствии с требованиями, изложенными в разделе 1. В поле свойства Caption написать “Таблица значений”. Значение свойства BorderStyle принять bsSingle.
9. Взаимно связать обе формы, поместив в их модули ссылки друг на друга так, как это делалось в лабораторной работе № 1.
10. Поместить на новую форму компонент StringGrid1 из палитры Additional. Этот компонент предназначен для создания текстовых таблиц.

Будем выводить в него формируемую таблицу значений функции. Свойству ColCount (количество колонок) присвоить значение 2. Количество строк RowCount задать 2. Это, впрочем, не имеет никакого значения, т.к. реальное количество строк в таблице будет определено и уточнено в процессе выполнения программы в зависимости от количества точек в таблице. Ширина столбцов по умолчанию DefaultColWidth равна 64. Высота строки по умолчанию DefaultRowHeight равна 24. Свойство ScrollBars установить равным ssVertical (только вертикальная полоса прокрутки). СвойствуFixedCols задать значение 0 (снять выделение начального столбца).

11. Изменить размеры второй формы таким образом, чтобы большую часть ее площади занимала полученная таблица (рисунок 4).

Рисунок 4 – Примерный вид формы для вывода таблицы значений.

12. На вторую форму поместить компоненты Label1 и Label2, в свойство Caption которых занести: “Максимум” и “Минимум” соответственно.

13. На вторую форму поместить также однострочные окна редактирования Edit1 и Edit2. Они будут использованы только для вывода максимального и минимального значений функции, вычисленных в процессе работы приложения. Ввод данных пользователем в эти окна не допускается. Поэтому свойству ReadOnly каждого компонента присвоить значение true (запрет ввода информации). Значение свойства BorderStyle принять bsNone для обоих компонентов. Цвет компонентов установить таким же, как и у формы, на которую выводится таблица значений (Color□clForm). Значение свойства Text можно очистить.

14. Отредактировать программный код для модуля первой формы (Unit1). Подпрограмма-функция f(x), осуществляющая вычисление значения функции не изменяется по сравнению с предыдущей лабораторной работой. Однако чтобы эта функция была доступна в модуле второй формы, где она и будет использована, в интерфейсный раздел модуля первой формы (interface) необходимо добавить описание этой функции:

function f(x,a:real):real;

В обработчиках событий OnExit однострочных окон редактирования, перешедших из предыдущей лабораторной работы подкорректировать текст сообщения об ошибке в соответствии с новым назначением окон. Для нового однострочного окна редактирования Edit4 и для окна редактирования Edit3 создать обработчики событий OnEnter и OnExit, аналогичные соответствующим обработчикам для остальных окон редактирования. При этом в момент исполнения события OnEnter должна становиться невидимой вторая форма. Обработчик события OnClick для командной кнопки Button1 (“скрыть/показать таблицу”) переписать заново. В этом обработчике сначала делается проверка исходных данных. Если какой-то параметр не задан или конечное значение диапазона табулирования окажется меньше начального – об этом выводится сообщение и фокус передается на соответствующий компонент для ввода данных. Если все данные введены, значение свойства Visible второй формы изменяется на противоположное. Таким образом, при каждом щелчке на кнопке Button1 состояние формы попеременно изменяется: она то показывается, то скрывается. Расчет таблицы значений выполняется непосредственно при показе второй формы (формы с таблицей).

```
procedure TL3_Ivanov_Form1.Button1Click(Sender: TObject);
begin
  If  (Edit1.Text='') Then
    Begin
      ShowMessage('Отсутствует значение A');
      Edit1.SetFocus;
    End
  Else If (Edit2.Text='') Then
    Begin
      ShowMessage('Отсутствует начальное значение X');
      Edit2.SetFocus;
    End
  Else If  (Edit3.Text='') Then
    Begin
      ShowMessage('Отсутствует конечное значение X');
      Edit3.SetFocus;
```

```
    End
  Else If  (Edit4.Text='') Then
    Begin
      ShowMessage('Отсутствует шаг H');
      Edit4.SetFocus;
    End
  Else if (StrToFloat(Edit3.Text))<=
         (StrToFloat(Edit2.Text)) Then
    Begin
       ShowMessage('Начальное значение X должно быть' +
                ' меньше конечного!');
       Edit3.SetFocus;
    End
  Else
    L3_Ivanov_Form2.Visible:= not L3_Ivanov_Form2.Visible
end;
```

15. Создать программный код для второго модуля второй формы (Unit2). Этот код содержит единственную процедуру – процедуру обработки события OnShow (показ формы), в которой и осуществляется расчет таблицы значений. Организуется цикл для расчета каждого значения. Результаты выводятся в компонент StringGrid1 (таблица строк). Количество строк в таблице формируется динамически: после формирования очередной строки общее количество строк (свойство RowCount) увеличивается на 1. Первоначально устанавливается количество строк, равное 2, т.е. на 1 больше, чем надо (первая строка – заголовок, составляет фиксированную зону). Это вызвано тем, что таблица должна иметь хотя бы одну информационную строку. Поэтому по завершению цикла таблица усекается на одну строку.

```
procedure TL3_Ivanov_Form2.FormShow(Sender: TObject);
  var  H,Xn,Xk,A,X,Y,Ymax,Ymin,Xmax,Xmin:Real;
begin
  // Формируем заголовок грида.
  StringGrid1.Cells[0,0]:='        X';
  StringGrid1.Cells[1,0]:='       f(X)';
  // Толщина разграничительных линий.
  // Можно было установить на этапе проектирования.
  StringGrid1.GridLineWidth:=2;
  // Количество строк устанавливаем 2, потом после
добавления
  // каждой строки будем наращивать по 1. В итоге получим
  // на одну строку больше.
  StringGrid1.RowCount:=2;
  // Переносим параметры из формы в рабочие переменные.
```

```
  A:=StrToFloat(L3_Ivanov_Form1.Edit1.Text);
  Xn:=StrToFloat(L3_Ivanov_Form1.Edit2.Text);
  Xk:=StrToFloat(L3_Ivanov_Form1.Edit3.Text);
  H:=StrToFloat(L3_Ivanov_Form1.Edit4.Text);
  // Присваиваем начальные значения для цикла.
  X:=Xn; Ymax:=-1e30;Ymin:=1e30;
  // Организуем цикл.
  // В цикле вычисляем значение функции в очередной точке,
  // Уточняем мах и min.
  // Заносим данные в таблицу и увеличиваем счетчик строк.
  While X<=Xk Do
  Begin
    Y:=f(X,A);
    If Y>Ymax Then Begin Ymax:=Y;Xmax:=X; End;
    If Y<Ymin Then Begin Ymin:=Y;Xmin:=X; End;
    StringGrid1.Cells[0,StringGrid1.RowCount-1]:=
        FloatToStrF(X,ffGeneral,6,6);
    StringGrid1.Cells[1,StringGrid1.RowCount-1]:=
        FloatToStrF(Y,ffGeneral,6,6);
    StringGrid1.RowCount:=StringGrid1.RowCount+1;
    X:=X+H;
  End;
  // По концу цикла уничтожаем лишнюю строку.
  StringGrid1.RowCount:=StringGrid1.RowCount-1;
  // Выводим значения мах и min.
  Edit1.Text:='Y='+FloatToStrF(Ymax,ffGeneral,6,6)+
        ' при '+'X='+FloatToStrF(Xmax,ffGeneral,6,6);
  Edit2.Text:='Y='+FloatToStrF(Ymin,ffGeneral,6,6)+
        ' при '+'X='+FloatToStrF(Xmin,ffGeneral,6,6);
end;
```

16. Нажав клавишу F9, запустить приложение на выполнение. Объяснить логику его функционирования. В случае необходимости выполнить отладку.
17. Сохранить отлаженный проект в папке размещения проекта лабораторной работы.
18. По результатам работы составить отчет.

Индивидуальные задания:

Функции из лабораторной работы №3, начальные и конечные значения переменной x и шаг табулирования приведены ниже.

1. Начальное значение - 2, конечное - 6, шаг-0,5
2. Начальное значение - 1, конечное - 10, шаг-1

3. Начальное значение - 2, конечное - 9, шаг-0,4
4. Начальное значение - 4, конечное - 14, шаг-0,5
5. Начальное значение - 2, конечное - 6, шаг-0,3
6. Начальное значение - 1, конечное-18, шаг-1
7. Начальное значение - 2, конечное - 9, шаг-0,7
8. Начальное значение - 5, конечное 20, шаг-1
9. Начальное значение - 3, конечное - 16, шаг-1
10. Начальное значение - 2, конечное - 8, шаг-0,9
11. Начальное значение- 2, конечное - 9, шаг -0,4
12. Начальное значение - 2, конечное - 8, шаг-0,5
13. Начальное значение - 2, конечное - 16, шаг-1

Лабораторная работа № 4
Элементы графики

Цель: Дальнейшее освоение работы с объектами Lazarus. Использование специальных графических компонентов (элементов управления) для рисования графиков функций.

Рабочее задание: На основе проекта Lab3 разработать новый проект, включив в него блок вывода на экран графика функции. Предусмотреть масштабирование графика, в зависимости от размеров окна.

Краткие теоретические сведения

Методы некоторых объектов, имеющих канву (точнее, методы самой канвы), например, форма, позволяют формировать графические изображения непосредственно, в том числе и рисовать графики. Однако среди многочисленных компонентов Lazarus имеется специальный объект типа TChart для графического представления числовых данных. Этот объект является панелью, на которой можно создавать диаграммы и графики различных типов. Компонент служит контейнером для объектов Series типа TLineSeries – серий данных, характеризующихся различными стилями отображения. Каждый компонент может включать несколько серий. Каждая серия имеет собственное имя, ее можно задавать программно как переменную. Многочисленные свойства самого компонента и его серий можно установить с помощью Окна Свойств или программно. Компонент содержит большое количество специфических свойств, событий и методов. Для программного создания графиков в лабораторной работе используются такие методы объектов Chart и Series:

- ClearSeries (очистка всех серий от занесенных ранее данных);
- AddXY (добавление в график новой точки).

Для вывода графика и таблицы значений можно было бы организовать отдельные формы, как в предыдущей работе, где была создана отдельная форма для вывода таблицы значений. В данной лабораторной работе, однако, применен другой прием: вся информация размещена в одной форме. А для

того, чтобы иметь возможность попеременно переключаться с просмотра таблицы на просмотр графика, и на ввод данных использован компонент типа TPageControl (Набор страниц с закладками). Назначение этого компонента – создание нескольких, перекрывающих друг друга страниц (панелей) класса TTabSheet. Каждая панель содержит свой набор компонентов. Доступ к конкретной странице осуществляется через связанный с ней корешок (закладку) – небольшой выступ над страницей, содержащий краткое название страницы.

Пример выполнения работы

1. Войти в среду Lazarus. Создать новый проект.
2. Используя палитру компонентов (элементов управления), спроектировать форму Form1, изменить значение ее свойства Name в соответствии с требованиями, описанными в разделе 1 (L4_<ФИО>_Form1). Для свойства BorderStyle лучше оставить значение bsSizeable (устанавливается по умолчанию), что позволяет изменять размеры формы (окна) в процессе выполнения приложения.
3. В нижней части формы поместить элемент управления BitBtn1 (командная кнопка) из страницы Additional с надписью «Выход». Растянуть его пошире, чтобы он имел ширину примерно такую, как форма. Чтобы при расширении формы этот компонент тоже растягивался, в сложном свойстве Anchors установит в true значения параметров akLeft и akRight. Установка в true значения параметра akBottom в этом же свойстве привяжет кнопку к нижнему краю формы: при изменении вертикального размера формы кнопка всегда будет находиться в нижней части формы.
4. Поместить на форму компонент PageControl1 из палитры Common Controls. Расположить его так, чтобы он занимал всю оставшуюся после размещения командной кнопки часть формы.
5. Привязать созданный компонент PageControl1 ко всем четырем сторонам формы, чтобы при изменении размеров формы соответственно изменялись и размеры компонента. Для этого в сложном свойстве Anchors значения всех параметров установить в true.
6. Щелкая правой клавишей мыши на компоненте PageControl1, и выбирая пункт контекстного меню Добавить страницу (New Page), создать три новых страницы.
7. Оставить названия полученных страниц TabSheet1… TabSheet3 или присвоить новые по своему усмотрению. В полях свойств Caption написать: “Ввод исходных данных”, ”Таблица значений” и “График функции” соответственно.
8. Страница TabSheet1 оформляется примерно как форма Form1 в лабораторной работе № 3 (рис.5). С помощью буфера обмена можно скопировать все элементы этой формы на страницу TabSheet1:

9. Страница TabSheet2 также оформляется аналогично форме Form2 в лабораторной работе №3 (рис.6). На ней размещается компонент StringGrid1 (текстовая таблица) из страницы Additional для размещения таблицы значений функции, два компонента Label1 и Label2, а также два компонента Edit1 и Edit2 для размещения максимального и минимального значений функции. Определить свойства этих компонентов как в лабораторной работе №3. Для StringGrid1 дать соответствующую привязку с помощью сложного свойства Anchors.

Рисунок 5 – Примерный вид первой страницы компонента **PageControl** проектируемой формы.

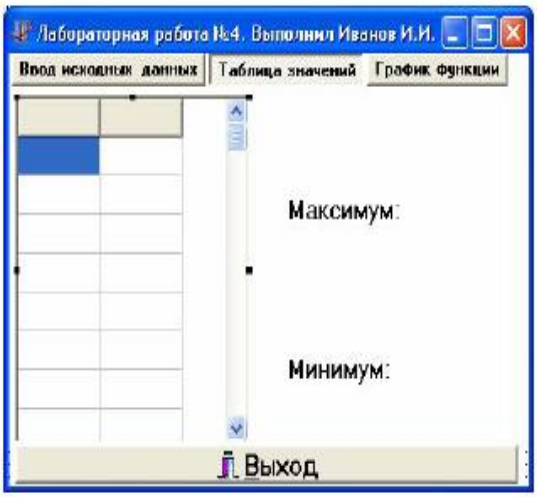


Рисунок 6 – Примерный вид второй страницы компонента PageControl проектируемой формы.

10. На третью страницу формы поместить компонент Chart1 из палитры Chart, который осуществляет рисование графиков функций. Перейти на свойство Titles и ввести в подсвойстве Text заголовок: “График функции, построенный по таблице значений”. Сделать заголовок видимым: Visible□true. Двойным щелчком на объекте Chart1 открываем окно редактора графиков (Edit series), добавляем (Add) новый график (Chart1LineSeries). Для данного графика с помощью окна свойств избавляемся от легенды (Legend□Visible□False). Остальные параметры графика будут в дальнейшем задаваться программно. Привязать компонент Chart1 ко всем четырем сторонам страницы компонента PageControl путем установки в true всех параметров свойства Anchors, установив предварительно необходимые размеры компонент Chart1 (точнее расстояния от краев компонента PageControl). Удобнее в этом случае воспользоваться свойством Align, определив его значение равным alClient (растянуть на всю клиентскую область). Привязка компонента к границам контейнера (которым является компонент PageControl) означает, что при изменении размеров контейнера (страницы) будут соответственно изменяться и размеры компонента Chart1. А поскольку сам контейнер (компонент PageControl), в свою очередь, привязан к границам формы, то при изменении размеров формы будут автоматически изменяться и размеры компонента PageControl, а значит и размеры компонента Chart1. При этом каждое изменение размеров компонента Chart1 будет автоматически инициировать перерисовку содержащегося в этом компоненте графика. Страница TabSheet3 будет иметь вид, примерно такой, как показано на рис.7

.

Рисунок 7 – Примерный вид третьей страницы компонента **PageControl** проектируемой формы.

11. Создать или скопировать с предыдущих лабораторных работ функцию f(x) для вычисления значения функции.
12. Создать или скопировать с предыдущих лабораторных работ обработчики событий OnExit для однострочных окон редактирования.
13. Написать обработчик события OnChange для компонента PageControl1. В нем сначала необходимо выполнить проверку исходных данных. Эту часть кода можно скопировать из обработчика Button1Click предыдущейлабораторной работы, внеся в него небольшие коррективы: при передаче фокуса на компонент, в котором необходимо исправить (ввести) данные следует, кроме того, активизировать первую страницу компонента PageControl1. Вторую часть данного обработчика можно скопировать из обработчика FormShow предыдущей лабораторной работы. В любом случае обработчик события OnChange должен иметь примерно следующий вид:

```
procedure  TL4_Ivanov_Form1.PageControl1Change(Sender:
TObject);
   var H,Xn,Xk,A,X,Y,Ymax,Ymin,Xmax,Xmin:Real;
   var MySerie: TLineSeries;
begin
  If (Edit1.Text='') Then
     Begin
       ShowMessage('Отсутствует значение A');
       PageControl1.ActivePageIndex:=0;
       Edit1.SetFocus;
     End
  Else If (Edit2.Text='') Then
     Begin
       ShowMessage('Отсутствует начальное значение X');
       PageControl1.ActivePageIndex:=0;
       Edit2.SetFocus;
     End
  Else If (Edit3.Text='') Then
     Begin
       ShowMessage('Отсутствует конечное значение X');
       PageControl1.ActivePageIndex:=0;
       Edit3.SetFocus;
     End
  Else If (Edit4.Text='') Then
     Begin
       ShowMessage('Отсутствует шаг H');
       PageControl1.ActivePageIndex:=0;
       Edit4.SetFocus;
     End
```

```
Else if (StrToFloat(Edit3.Text))<=
       (StrToFloat(Edit2.Text)) Then
  Begin
    ShowMessage('Начальное значение X должно быть' +
            ' меньше конечного!');
    PageControl1.ActivePageIndex:=0;
    Edit3.SetFocus;
  End
Else
  begin
    StringGrid1.RowCount:=1;
    StringGrid1.Cells[0,0]:='       X';
    StringGrid1.Cells[1,0]:='      f(X)';
    StringGrid1.GridLineWidth:=2;
    A:=StrToFloat(Edit1.Text);
    Xn:=StrToFloat(Edit2.Text);
    Xk:=StrToFloat(Edit3.Text);
    H:=StrToFloat(Edit4.Text);
    X:=Xn;Ymax:=-1e30;Ymin:=1e30;
    // Очищаем компонент Chart1. Там могут
    // остаться данные от предыдущего просмотра.
    // Создаем свою серию в компоненте Chart1
    // Определяем ее цвет (красный)
 Chart1.ClearSeries;
    MySerie:=TLineSeries.Create(Chart1);
    MySerie.LinePen.Color:=clRed;
    Chart1.AddSeries(MySerie);
    //Организуем цикл, вычисляющий f(x) в каждой точке
    // и строящий график функции.
    While X<=Xk Do
     Begin
      Y:=f(X,A);
      If Y>Ymax Then Begin Ymax:=Y;Xmax:=X; End;
      If Y<Ymin Then Begin Ymin:=Y;Xmin:=X; End;
      StringGrid1.RowCount:=StringGrid1.RowCount+1;
      StringGrid1.Cells[0,StringGrid1.RowCount-1]:=
          FloatToStrF(X,ffGeneral,6,6);
      StringGrid1.Cells[1,StringGrid1.RowCount-1]:=
          FloatToStrF(Y,ffGeneral,6,6);
      // Добавляем очередную точку в график функции.
      MySerie.AddXY(X,Y);
      X:=X+H;
     End;
    Edit6.Text:='Y='+FloatToStrF(Ymax,ffGeneral,6,6)+
```

```
        ' при '+'X='+FloatToStrF(Xmax,ffGeneral,6,6);
      Edit5.Text:='Y='+FloatToStrF(Ymin,ffGeneral,6,6)+
        ' при '+'X='+FloatToStrF(Xmin,ffGeneral,6,6);
      if (PageControl1.ActivePageIndex = 1) Then
        StringGrid1.SetFocus;
    End;
end;
```

14. При запуске проекта должна активироваться первая вкладка PageControl1 и фокус передаваться однострочному текстовому полю для ввода A. Поэтому создаем обработчик OnActivate для формы:

```
procedure TL4_Ivanov_Form1.FormActivate(Sender: TObject);
begin

  PageControl1.ActivePageIndex:=0;
  Edit1.SetFocus;
end;
```

15. Нажав на клавиатуре клавишу F9, запустить приложение на выполнение. В случае необходимости выполнить отладку. Объяснить работу приложения.
16. Сохранить отлаженный проект в папке размещения проекта лабораторной работы.
17. По результатам выполнения работы составить отчет.

Индивидуальные задания: функции берутся из лабораторной работы №4, по этим функциям построить графики.

Лабораторная работа №5
Одномерный массив

Цель: Изучение дополнительных возможностей объектно – ориентированного программирования в среде Lazarus.

Краткие теоретические сведения.

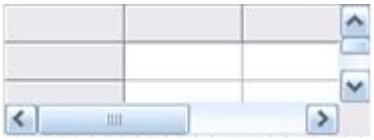
При работе с массивами обычно используется цикл с параметром.

Для работы с массивами используются следующие компоненты:

1. StringGrid – таблица строк, предназначена для создания таблиц, в ячейках которых располагаются произвольные текстовые строки. Компонент находится на странице Additional. Таблица делится на фиксированную и рабочую (рис.).

Фиксированная служит для показа заголовков строк (столбцов) и выделена темно-серым цветом; рабочая – это остальная часть

таблицы. Если рабочая часть целиком не помещается в пределах окна, то в этом случае автоматически появляются полосы прокрутки.



Основные свойства компонента *StringGrid*

FixedCols	Количество фиксированных столбцов
FixedRows	Количество фиксированных строк. Если эти свойства равны 0, то таблица не содержит фиксированной части
ColCount	Количество столбцов
RowCount	Количество строк
Cells	Двумерный массив ячеек
+Option	Параметры. «+» указывает на наличие вложенных свойств. Из всех вложенных свойств выбирается *GoEditing – true* (разрешено редактирование ячеек)

В компоненте StringGrid строки и столбцы нумеруются с 0.
К каждой ячейке таблицы можно обратиться
Cells[<номер столбца>,<номер строки>].

2. BitBtn – командная кнопка с изображением. Находится на странице Additional.

Основные свойства компонента *BitBtn*

Caption	Надпись на кнопке
Kind	Предлагает на выбор десять предопределенных типов кнопок *(bkOk, bkYes, bkCancel* и т.д.*)*
Glyph	Определяет связанное с кнопкой изображение

Задание 1. Дан массив A, состоящий из элементов целого типа. Найти количество положительных элементов этого массива.
Определим переменные задачи:
входные данные: a[i,0]: integer – элементы массива; n: integer – количество элементов массива промежуточные переменные: i: integer – параметр цикла результат: k: integer – количество положительных элементов массива Расположим следующие компоненты на форме .

Свойства выбранных компонент:
Form1 – Caption – Положительные элементы
Button1 – Caption – Сформировать массив
Edit1, 2 – Text – пусто
BitBtn – Kind – bkOk
Label1 – Caption – Введите количество элементов массива

StringGrid1 – ColCount – 100
RowCount – 1
Visible – False
FixedCols – 0
FixedRows – 0
Option – GoEditing – True

Процедура для кнопки «Сформировать массив»:

```
procedure TForm1.Button1Click(Sender: TObject);
var n:integer;
begin
n:=strtoint(edit1.text);
stringgrid1.ColCount:=n;
stringgrid1.Visible:=true;
end;
```

Процедура для кнопки «OK»:

```
procedure TForm1.BitBtn1Click(Sender: TObject);
var
 i,k:integer;
 a:array[0..10,0..10] of integer;
begin
for i:=0 to n-1 do
a[i,0]:=strtoint(stringGrid1.cells[i,0]);
k:=0;
for i:=0 to n-1 do
 if a[i,0]>0
  then k:=k+1;
edit2.text:=inttostr(k);
end;
```

В задачах на массивы часто используется генератор случайных чисел. Для этого необходимо запустить процедуру Randomize, которая инициализирует (запускает) генератор случайных чисел.
Чтобы получить случайное число, нужно воспользоваться функцией Random.
Пример.

```
Randomize;
    for i:=0 to n-1 do
      a[i,0]:=random(20)-10;
```

Таким образом, диапазон значений массива будет (–10; 9).
Задание 2. В одномерном массиве положительные элементы, кратные 3, заменить на нуль. Вывести полученный массив. Заполнение исходного массива осуществить с использованием генератора случайных чисел.

Определим переменные задачи:
входные данные: a[i,0]: integer – элементы массива; n: integer – количество элементов массива промежуточные переменные: i: integer – параметр элемента результат: a[i,0]: integer – элементы массива

Расположим следующие компоненты на форме .

Свойства выбранных компонент:
Form1 – Caption – Замена элементов
Label1 – Caption – Введите количество элементов
Button1 – Caption – Сформировать массив
Button2 – Caption – Новый массив
Edit1 – Text – пусто
StringGrid2 – ColCount – 100
RowCount – 1
Visible – False
FixedCols – 0
FixedRows – 0
Option – GoEditing – True
StringGrid1 – ColCount – 100
RowCount – 1
Visible – False
FixedCols – 0
FixedRows – 0
Option – GoEditing – True

Процедура для кнопки «Сформировать массив»:

```
procedure TForm1.Button1Click(Sender: TObject);
var n,i:integer;
a:array[0..100,0..100] of integer;
begin
n:=strtoint(edit1.text);
stringgrid1.ColCount:=n;
stringgrid1.Visible:=true;
Randomize;
for i:=0 to n-1 do
```

```
 begin
   a[i,0]:=random(20)-10;
   stringgrid1.cells[i,0]:=inttostr(a[i,0]);
 end;
end;
```

Процедура для кнопки «Новый массив»:

```
procedure TForm1.Button2Click(Sender: TObject);
var
 i,n:integer;
 a:array[0..10,0..10] of integer;
begin
n:=strtoint(edit1.text);
for i:=0 to n-1 do
a[i,0]:=strtoint(stringGrid1.cells[i,0]);
for i:=0 to n-1 do
 if (a[i,0]>0) and (a[i,0] mod 3 =0)
  then
    a[i,0]:=0;
 stringgrid2.Visible:=true;
 StringGrid2.ColCount:=n;
for i:=0 to n-1 do
StringGrid2.Cells[i,0]:=IntToStr(a[i,0]);
end;
```

Индивидуальные задания:

Вариант №1	1. Дан одномерный массив и число k. Найти сумму квадратов элементов массива, кратных заданному числу k. 2. Из элементов одномерного массива C сформировать массив A той же размерности по правилу: если номер элемента четный, то $A_i = C_i^2$; если номер элемента нечетный, то $A_i = 2C_i$.
Вариант №2	1. Дан одномерный массив и число b. Найти количество элементов массива, равных числу b. 2. Из элементов одномерного массива A сформировать массив D той же размерности по правилу: первые k элементов - $D_i = A_i + i$, остальные - $D_i = A_i - i$.
Вариант №3	1. Дан одномерный массив и число n. Найти количество элементов массива, значения которых не превосходят заданного числа n. 2. Из элементов одномерного массива A сформировать массив C той же размерности по правилу: первые k элементов - $A_i = -C_i^2$, остальные - $A_i = C_i - 1$.
Вариант №4	1. Дан одномерный массив. Найти количество элементов, больших среднего арифметического всех элементов массива. 2. Из элементов одномерного массива C сформировать массив A той же размерности по правилу: если номер элемента четный, то $A_i = C_i$; если номер элемента нечетный, то $A_i = 2$.
Вариант №5	1. Дан одномерный массив и число m. Найти произведение отрицательных элементов массива, кратных заданному числу m. 2. Из элементов одномерного массива A сформировать массив D той же размерности по правилу: $D_i = A_i^i$.

Вариант №6	1. Дан одномерный массив. Найти произведение ненулевых элементов и количество нулевых элементов. 2. Из элементов одномерного массива M сформировать массив P той же размерности по правилу: если номер элемента четный, то $P_i = iM_i$; если номер элемента нечетный, то $P_i = 5M_i$.
Вариант №7	1. Дан одномерный массив. Найти минимальный элемент и его номер. 2. Из элементов одномерного массива C сформировать массив A той же размерности по правилу: если номер элемента четный, то $A_i = C_i + 5$; если номер элемента нечетный, то $A_i = C_i - 5$.
Вариант №8	1. Дан одномерный массив. Найти разность максимального и минимального элементов массива. 2. Из элементов одномерного массива A сформировать массив C той же размерности по правилу: первые k элементов - $A_i = 10C_i$, остальные - $A_i = 2C_i$.

Лабораторная работа №6

Работа с матрицами

Цель: Получить практические навыки при работе с матрицами при создании консульного и визуального приложений в Lazarus.

Краткие теоретические сведения.

Задание 1. Написать программу, вычисляющую количество и произведение ненулевых элементов матрицы.

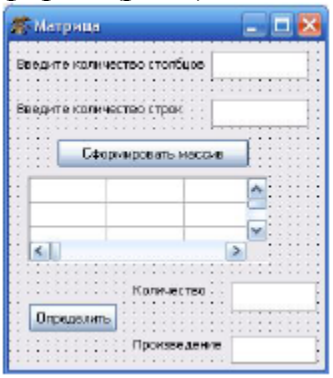
Определим переменные задачи:

входные данные: a[i,j]: integer – элементы массива; n, m: integer – количество столбцов и строк соответственно

промежуточные переменные: i, j: integer – параметры элементов

результат: P: integer – произведение ненулевых элементов; k: integer – количество ненулевых элементов

Расположим следующие компоненты на форме (рис.).



Свойства выбранных компонент:

Form1 – Caption – Матрица

Label1 – Caption – Введите количество столбцов
Label2 – Caption – Введите количество строк
Label3 – Caption – Произведение
Label4 – Caption – Количество
Button1 – Caption – Сформировать массив
Button2 – Caption – Определить
Edit1..4 – Text – пусто
StringGrid1 – ColCount – 100
RowCount – 100
Visible – False
FixedCols – 0
FixedRows – 0
Option – GoEditing – True
Процедура для кнопки «Сформировать массив»:

```
procedure TForm1.Button1Click(Sender: TObject);
var n,m:integer;
begin
n:=strtoint(edit1.Text);
m:=strtoint(edit2.Text);
stringgrid1.ColCount:=n;
stringgrid1.RowCount:=m;
stringgrid1.visible:=true;
end;
```

Процедура для кнопки «Определить»:

```
procedure TForm1.Button2Click(Sender: TObject);
var i,j,n,m,k,p:integer;
    a:array[0..100,0..100] of integer;
begin
n:=strtoint(edit1.text);
m:=strtoint(edit2.text);
for i:=0 to n-1 do
for j:=0 to m-1 do
a[i,j]:=strtoint(stringgrid1.cells[i,j]);
p:=1;
k:=0;
for i:=0 to n-1 do
for j:=0 to m-1 do
 if a[i,j]<>0
  then
   begin
    p:=p*a[i,j];
    k:=k+1;
   end;
edit3.text:=inttostr(k);
```

edit4.text:=inttostr(p);
end;

Задание 2. В матрице, размерность которой nxn, отрицательные элементы заменить на их квадраты, а положительные элементы уменьшить на 10. Вывести полученный массив. Заполнение исходного массива осуществить с использованием генератора случайных чисел.
Определим переменные задачи: входные данные: a[i,j]: integer – элементы массива; n: integer – количество столбцов и строк промежуточные переменные: i, j: integer – параметры элемента результат: a[i,j]: integer – элементы массива. Расположим следующие компоненты на форме(рис.).
Свойства выбранных компонент:
Form1 – Caption – Замена элементов
Label1 – Caption – Введите количество элементов
StringGrid2 – ColCount – 100
RowCount – 100
Visible – False
FixedCols – 0
FixedRows – 0

Option – GoEditing –True
Button1 – Caption – Сформировать массив
Button2 – Caption – Новый массив
Edit1 – Text – пусто
StringGrid1 – ColCount – 100
RowCount – 100
Visible – False
FixedCols – 0
FixedRows – 0
Option – GoEditing – True
Процедура для кнопки «Сформировать массив»:
procedure TForm1.Button1Click(Sender: TObject);
var n,i,j:integer;
a:array[0..100,0..100] of integer;
begin

```
n:=strtoint(edit1.text);
stringgrid1.ColCount:=n;
stringgrid1.RowCount:=n;
stringgrid1.Visible:=true;
Randomize;
for i:=0 to n-1 do
for j:=0 to n-1 do
 begin
   a[i,j]:=random(20)-10;
   stringgrid1.cells[i,j]:=inttostr(a[i,j]);
 end;
end;
```

Процедура для кнопки «Новый массив»:

```
procedure TForm1.Button2Click(Sender: TObject);
var  i,j,n:integer;
       a:array[0..100,0..100] of integer;
begin
56
n:=strtoint(edit1.text);
for i:=0 to n-1 do
for j:=0 to n-1 do
a[i,j]:=strtoint(stringGrid1.cells[i,j]);
for i:=0 to n-1 do
for j:=0 to n-1 do
 if a[i,j]<0
  then a[i,j]:=sqr(a[i,j])
    else
      if a[i,j]>0
        then a[i,j]:=a[i,j]-10;
 stringgrid2.Visible:=true;
 StringGrid2.ColCount:=n;
for i:=0 to n-1 do
for j:=0 to n-1 do
StringGrid2.Cells[i,j]:=inttostr(a[i,j]); end;
```

Индивидуальные задания:

1. Определить номера строки и столбца максимального простого числа прямоугольной матрицы A(n,m). Подсчитать количество нулевых элементов матрицы и напечатать их индексы.

2. Найти среднее геометрическое значение элементов квадратной диагонали, если это возможно. Если среднее геометрическое вычислить

невозможно, то поменять местами максимальный и минимальный элементы матрицы.

3. Сформировать вектор D, каждый элемент которого представляет собой среднее арифметическое значение элементов строк матрицы C(k,m), и вектор G – любой его компонент должен быть равен произведению элементов соответствующего столбца матрицы C.

4. Задана матрица A(n,m), в каждом столбце которой максимальный элемент необходимо заменить произведением отрицательных элементов этого же столбца.

5. Задана матрица A(n,n). Определить максимальный элемент среди элементов матрицы, расположенных выше главной диагонали, и минимальный элемент среди тех, что находятся ниже побочной диагонали. После этого выполнить сортировку каждого столбца матрицы по возрастанию.

6. Заменить строку матрицы P(n,m) с минимальной суммой элементов на строку, где находится максимальный элемент матрицы.

7. Переместить максимальный элемент матрицы F(k,p) в правый верхний угол, а минимальный элемент – в левый нижний.

8. Проверить, является ли матрица A(n,n) диагональной (все элементы нули, кроме главной диагонали), единичной (все элементы нули, на главной диагонали только единицы) или нулевой (все элементы нули).

9. Сформировать из некоторой матрицы A(n,n) верхнетреугольную матрицу B(n,n) (все элементы ниже главной диагонали нулевые), нижнетреугольную матрицу C(n,n) (все элементы выше главной диагонали нулевые) и диагональную матрицу D(n,n) (все элементы нули, кроме главной диагонали) .

10. Заданы матрицы A(m,n) и B(n,m). Найти матрицу C=(A·B)4.

11. Проверить, является ли матрица B(n,n) обратной к A(n,n). Произведением матриц A и B в этом случае должна быть единичная матрица.

12. Определить количество простых чисел, расположенных вне диагоналей матрицы B(n,n).

13. Проверить, лежит ли на главной диагонали максимальный отрицательный элемент матрицы A(n,n).

14. Переписать простые числа из матрицы A в массив B. Массив упорядочить по убыванию.

15. Переписать положительные числа из матрицы целых чисел A в массив B. Из массива B удалить числа, в двоичном представлении которых единиц больше, чем нулей.

16. Даны четыре квадратные матрицы A(n,n), B(n,n), C(n,n), D(n,n), в которых хранятся целые числа. Найти матрицу, в которой находится максимальное простое число.

17. Заданы четыре квадратные матрицы A(n,n), B(n,n), C(n,n), D(n,n), в которых хранятся целые числа. Найти матрицы, в которых на диагоналях есть простые числа.

18. Заданы три прямоугольные матрицы A(n,m), B(r,p), C(k,q). Найти матрицы, в которых по периметру расположены только отрицательные числа.

19. Проверить, лежит ли на побочной диагонали минимальный положительный элемент матрицы A(n,n).

20. Заданы матрицы D(n,n), A(m,n) и B(n,m). Найти матрицу C=(B·A). Проверить, является ли матрица C(n,n) обратной к D(n,n). Произведением матриц C и D в этом случае должна быть единичная матрица.

21. Заданы четыре квадратные матрицы A(n,n), B(n,n), C(n,n), D(n,n), в которых хранятся целые числа. Найти, в какой из матриц на побочной диагонали есть числа, состоящие из восьмерок.

22. Заменить столбец матрицы P(n,m) с максимальной суммой элементов на столбец, где находится максимальное число, состоящее из единиц.

23. Заданы четыре квадратные матрицы A(n,n), B(n,n), C(n,n), D(n,n), в которых хранятся целые числа. Определить, есть ли среди них матрицы, в которых на побочной диагонали находятся только числа, состоящие из единиц и двоек.

24. Переписать простые числа из матрицы целых чисел A в массив B. Из массива B удалить числа, расположенные между максимальным и минимальным элементами.

25. В матрице целых чисел A(n,n) упорядочить те строки, в которых диагональные элементы не содержат семерок.

Лабораторная работа №7

Работа с файлами, типизированными файлами

Цель: Получить практические навыки при работе с различными файлами, уметь связывать различные файлы при создании визуальных приложений в Lazarus.

Рабочее задание: Во всех заданиях составить две программы. Первая должна формировать типизированный файл. Вторая – считать данные из этого файла, выполнить соответствующие вычисления и записать их результаты в текстовый файл.

1. Создать типизированный файл, куда записать n целых чисел. Из исходного файла сформировать массивы четных и нечетных чисел. Определить наибольший отрицательный компонент файла и наименьший положительный.

2. Создать типизированный файл, куда записать n целых чисел. На основе исходного файла создать массив утроенных четных чисел. Упорядочить его по убыванию элементов.

3. Создать типизированный файл, куда записать n целых чисел. Сформировать массив положительных чисел, делящихся на семь без остатка, используя элементы исходного файла. Упорядочить массив по возрастанию элементов.

4. Создать типизированный файл, куда записать n вещественных чисел. Из компонентов исходного файла сформировать массивы, из чисел, больших 10 и меньших двух. Вычислить количество нулевых компонентов файла.

5. Создать типизированный файл, куда записать n целых чисел. Из файла создать массив, элементы которого являются простыми числами и расположены после максимального элемента.

6. Создать типизированный файл, куда записать n целых чисел. Из файла целых чисел сформировать массив, записав в него только четные компоненты, находящиеся до минимального элемента.

7. Создать типизированный файл, куда записать n вещественных чисел. Сделать массив из элементов исходного файла, внеся в него числа, превосходящие среднее значение среди положительных значений файла.

8. Создать типизированный файл, куда записать n целых чисел. Из исходного файла сформировать массив, записав в него числа, расположенные в файле до максимального элемента и после минимального.

9. Создать типизированный файл, куда записать n целых чисел. Массив создать из исходного файла. Внести в него простые и совершенные числа, расположенные в файле между минимальным и максимальным элементами.

10. Создать типизированный файл, куда записать n целых чисел. Из исходного файла сформировать массив, в котором вначале расположить четные, а затем нечетные числа. Определить номера наибольшего нечетного и наименьшего четного компонентов.

11. Создать типизированный файл, куда записать n целых чисел. В файле поменять местами минимальный среди положительных элементов и третий по счету простой элемент.

12. Создать типизированный файл, куда записать n целых чисел. Из файла переписать все простые, расположенные после максимального элемента в новый файл.

13. Создать типизированный файл, куда записать n целых чисел. Найти среднее арифметическое среди положительных чисел, расположенных до второго простого числа.

14. Создать типизированный файл, куда записать n целых чисел. Поменять местами последнее совершенное и третье отрицательное числа в файле.

15. Создать типизированный файл, куда записать n целых чисел. Все совершенные и простые числа из исходного файла записать в массив, который упорядочить по возрастанию.

16. Создать типизированный файл, куда записать n целых чисел. Последнюю группу расположенных подряд положительных чисел из исходного файла переписать в текстовый файл.

17. Создать типизированный файл, куда записать n целых чисел. Найти в нем группу подряд расположенных простых элементов наибольшей длины.

18. Создать типизированный файл, куда записать n целых чисел. Из исходного файла сформировать массивы простых и отрицательных чисел. Определить наименьшее простое число в файле и наибольшее совершенное.

19. Создать типизированный файл, куда записать n целых чисел. Из файла создать массив, элементы которого не являются простыми числами и расположены до максимального значения файла.

20. Создать типизированный файл, куда записать n целых чисел. Из файла целых чисел сформировать массив, записав в него только кратные 5 и 7 значения, находящиеся после максимального элемента файла.

21. Создать типизированный файл, куда записать n вещественных чисел. Сделать массив из элементов исходного файла, внеся в него числа, превосходящие среднее значение среди положительных значений файла.

22. Создать типизированный файл, куда записать n вещественных чисел. Поменять местами последнее отрицательное число в файле с четвертым по счету числом.

23. Создать типизированный файл, куда записать n вещественных чисел. Найти сумму третьей группы подряд расположенных отрицательных элементов.

24. Создать типизированный файл, куда записать n целых чисел. Удалить из него четвертую группу, состоящую из подряд расположенных простых чисел.

Лабораторная работа №8

Строки, списки. Работа с ними.

Цель: Получить практические навыки при работе со строками списками при создании визуальных приложений в Lazarus.

1. Посчитать количество запятых в строке.
2. Заменить в строке все цифры на пробел. Вывести количество замен.
3. Посчитать в строке количество цифр.
4. Удалить из строки все запятые.
5. Посчитать в строке количество слов.

6. Удалить из строки все слова, начинающиеся на букву «о».

7. После каждого пробела вставить символ *.

8. Найти в строке самое длинное слово.

9. Перед каждым пробелом вставить пробел и символ +.

10. Посчитать сумму всех чисел, которые встречаются в строке.

11. Посчитать в строке количество слов, начинающихся на «Ав».

12. Заменить в строке двойные пробелы на одинарный пробел. Вывести количество замен.

13. Вставить после каждого слова запятую.

14. После каждого слова вставить символ «;».

15. Посчитать в строке количество символов «:» и «;».

16. Удалить из строки все цифры.

17. Посчитать в строке количество слов, заканчивающихся символами «ая».

18. Найти в строке самое короткое слово.

19. Вставить после каждого слова, заканчивающегося на букву «о», слово «Ого».

20. Удалить из строки все слова, состоящие из пяти букв.

21. Найти в строке количество слов, начинающихся на букву «а» и заканчивающихся буквой «т».

22. Удалить из строки второе, третье и пятое слова.

23. Перед каждой цифрой вставить символ №.

24. Посчитать в строке количество гласных букв.

25. Удалить из строки все слова, начинающиеся и заканчивающиеся на «о».

Лабораторная работа №9

Записи. Создание баз данных в Lazarus.

Цель: Получить практические навыки при работе с записями, отработать все способы обращения к полям записи, уметь создавать небольшие базы данных в приложении Lazarus.

1. Создать структуру с данными по таблице

Город	Прирост населения, тыс. чел.				
	1999	2000	2001	2002	2003
Макеевка	2,5	1,3	-0,2	-0,1	0,6
...					

Добавить и вычислить в структуре поле «Средний прирост». Определить количество городов с отрицательным приростом в 2003 году. Упорядочить записи по возрастанию среднего прироста.

2. Создать структуру с данными по таблице варианта №1. Добавить и вычислить в структуре поле «Минимальный прирост». Определить количество городов с приростом в 2003 году более 2 тыс. чел. Выполнить сортировку записей по полю «Город». Названия городов упорядочить по алфавиту.

3. Создать структуру с данными по таблице

Название	Фабрика	Цена	Дата выпуска	Количество
Паровозик	Игрушка	125,00	01.02.2007	
...				

Добавить и вычислить в структуре поле «Цена со скидкой», вводя процент скидки с формы. Найти общее количество игрушек с фабрики «Игрушка». Упорядочить записи по убыванию поля «Цена».

4. Создать структуру с данными по таблице варианте №3. Добавить и вычислить в структуре поле «Сумма продажи». Найти количество названий игрушек, у которых цена меньше общей средней цены всех игрушек. Упорядочить записи по названию игрушек.

5. Создать структуру с данными по таблице

Фамилия	Имя	Дата рождения	Школа	Класс
Сергеев	Сергей	05.05.1994	112	9-А
...				

Добавить и вычислить в структуре поле «Возраст», вводя текущую дату с формы. Определить количество школьников с именем Сергей. Упорядочить записи по номеру школы.

6. Создать структуру с данными по таблице варианта №5. Добавить и вычислить в структуре поле «Год обучения», убрав из названия класса букву. Найти количество учеников 9-х классов. Выполнить сортировку записей по полю «Фамилия». Фамилии упорядочить по алфавиту.

7. Создать структуру с данными по таблице

Принтер	Количество, шт.			Цена, $
	Январь	Февраль	Март	
Samsung CLP-310	25	20	26	550
...				

Добавить и вычислить в структуре поле «Выручка». Найти среднюю цену принтеров. Упорядочить записи по возрастанию поля «Цена».

8. Создать структуру с данными по таблице варианта №7. Добавить и вычислить в структуре поле «Среднее количество». Найти среднее количество принтеров в каждом месяце. Упорядочить записи по названию принтера.

9. Создать структуру с данными по таблице варианта №7. Добавить и вычислить в структуре поле «Общее количество». Найти общее количество

проданных принтеров в каждом месяце. Упорядочить записи по возрастанию поля «Общее количество».

10. Создать структуру с данными по таблиц варианта №7. Добавить и вычислить в структуре поле «Цена со скидкой», вводя процент скидки с формы. Найти количество принтеров с ценой более 500$. Упорядочить записи по убыванию цены.

11. Создать структуру с данными по таблице

ФИО	Дата рождения	Должность	Стаж	Оклад
Сергеев С. И.	12.03.1966	Менеджер	2	1250
...				

Добавить и вычислить в структуре поле «Премия», рассчитав ее по следующему принципу: 20% от оклада, если стаж более 10 лет, в противном случае 10%. Найти количество сотрудников со стажем более 10 лет. Упорядочить записи по должности.

12. Создать структуру с данными по таблице варианта №11. Добавить и вычислить в структуре поле «Возраст», текущую дату вводить с формы. Найти средний оклад всех сотрудников. Упорядочить записи по ФИО.

13. Создать структуру с данными по таблице варианта №11. Добавить и вычислить в структуре поле «Возраст», текущую дату вводить с формы. Определить количество молодых специалистов (моложе 25 лет). Упорядочить записи по возрастанию оклада.

14. Создать структуру с данными по таблице

Место отдыха	Количество, шт.			Цена, $
	Июль	Август	Сентябрь	
Геленджик	255	203	198	510
...				

Добавить и вычислить в структуре поле «Среднее количество».
Найти общее количество путевок в каждом месяце. Упорядочить записи по месту отдыха.

15. Создать структуру с данными по таблице варианта №14. Добавить и вычислить в структуре поле «Доход от путевок». Найти среднюю цену путевки. Упорядочить записи по возрастанию цены.

16. Создать структуру с данными по таблице

ФИО	Дата рождения	Должность	Пол	Оклад
Сергеев С. И.	12.03.1966	Менеджер	Муж.	1250
...				

Добавить и вычислить в структуре поле «Зарплата», рассчитав ее по следующему принципу: к окладу добавить премию в размере 15% от оклада. Упорядочить записи по ФИО.

17. Создать структуру с данными по таблице варианта №16. Добавить и вы-

числить в структуре поле «Возраст», текущую дату вводить с формы. Определить количество мужчин и женщин. Упорядочить записи по должности.

18. Создать структуру с данными по таблице

Фамилия	Инициалы	Ученая степень	Год рождения	Количество статей
Сергеев	С. А.	Доцент	1971	25
...				

Добавить и вычислить в структуре поле «Активность» по следующему принципу: если количество статей более 10, то в поле записать пробел, в противном случае – фразу «Работать лучше». Упорядочить записи по фамилии.

19. Создать структуру с данными по таблице варианта №17. Удалить сотрудника с фамилией, которая вводится с формы. Определить количество доцентов. Упорядочить записи по должности.

20. Создать структуру с данными по таблице

Название	Автор	Издательство	Год издания	Цена, $	Тираж
Вий	Гоголь Н. В.	Правда	1971	6,5	15000
...					

Добавить и вычислить в структуре поле «Стоимость тиража». Найти общий тираж книг 2005 года. Упорядочить записи по автору.

21. Создать структуру с данными по таблице варианта №20. Удалить все записи книг тиража 2000 года. Найти среднюю цену книг типографии «Правда». Упорядочить записи по году издания.

22. Создать структуру с данными по таблице

ФИО абонента	Номер	Дата звонка	Город	Стоимость 1 минуты разговора	Количество минут
Моль Р. Ю.	956-25-78	12.05.2003	Казань	3,65	2
...					

Добавить и вычислить в структуре поле «Стоимость звонка». Найти общую стоимость звонков в город, вводимый по запросу. Упорядочить записи по ФИО абонента.

23. Создать структуру с данными по таблице варианта №22. Удалить все записи звонков с номерами, начинающимися с цифры 3. Упорядочить записи по названию города.

Контрольные вопросы:

1. Понятие алгоритма. Свойства алгоритмов. Формы записей алгоритмов. Общие принципы построения алгоритмов.
2. Основные алгоритмические конструкции: линейные, разветвляющиеся, циклические.
3. Данные: понятие и типы. Основные базовые типы данных и их характеристика. Структурированные типы данных и их характеристика. Методы сортировки данных.
4. Этапы решения задач на ЭВМ.
5. Основы алгебры логики. Логические операции с высказываниями: конъюнкция, дизъюнкция, инверсия. Законы логических операций. Таблицы истинности.
6. Эволюция языков программирования. Классификация языков программирования. Элементы языков программирования. Понятие системы программирования.
7. Исходный, объектный и загрузочный модули. Интегрированная среда программирования.
8. Общие принципы разработки программного обеспечения. Жизненный цикл программного обеспечения. Типы приложений. Консольные приложения. Оконные Windows приложения. Web-приложения. Библиотеки. Web-сервисы.
9. История развития ООП. Базовые понятия ООП: объект, его свойства и методы, класс, интерфейс. Основные принципы ООП: инкапсуляция, наследование, полиморфизм.
10. Событийно-управляемый модель программирования. Компонентно-ориентированный подход. Классы объектов. Компоненты и их свойства
11. Требования к аппаратным и программным средствам интегрированной среды разработчика.
12. Интерфейс среды разработчика: характеристика, основные окна, инструменты, объекты.
13. Форма и размещение на ней управляющих элементов. Панель компонентов и их свойства.
14. Окно кода проекта. Состав и характеристика проекта. Выполнение проекта. Настройка среды и параметров проекта.
15. Проектирование объектно-ориентированного приложения. Создание интерфейса пользователя. Программирование приложения.
16. Тестирование, отладка приложения. Создание документации
17. Основные компоненты (элементы управления) интегрированной среды разработки, их состав и назначение. Дополнительные элементы управления
18. Свойства компонентов (элементов управления). Виды свойств. Синтаксис определения свойств. Категория свойств.

19. Назначение свойств и их влияние на результат. Управление объектом через свойства. События компонентов (элементов управления), их сущность и назначение.
20. Создание процедур на основе событий. Процедуры, определенные пользователем: синтаксис, передача аргументов. Вызов событий.
21. Разработка функционального интерфейса приложения. Создание интерфейса приложения. Разработка функциональной схемы работы приложения.
22. Создание процедур обработки событий.
23. Компиляция и запуск приложения.
24. Понятие класса в Delphi. Отличие класса Delphi от записей Pascal.
25. Свойства и методы базового класса Delphi TObject.
26. Создание и уничтожение экземпляра класса в Delphi.
27. Понятие свойства класса. Синтаксис свойств и их достоинства.
28. Описание классов в Delphi. Области видимости и их отличительные особенности.
29. События и делегирование.
30. Библиотека компонент Delphi. Визуальные и невизуальные компоненты.
31. Иерархия классов Delphi. Краткая характеристика основных классов Delphi и их назначение.
32. Стандартные события (события мыши, клавиатуры, системные события) визуальных компонент.
33. Организация текстового диалога. Обзор стандартных окон и стандартных компонент.
34. Работа с многострочным текстом. Компонент TMemo, классы Tstrings, TStringList.
35. Обзор стандартных компонент управления (выключатели, переключатели, списки, контейнеры). Их взаимодействие.
36. Реализация механизма буксировки Drag&Drop.
37. Типовые окна диалога.
38. Организация меню. Главное меню приложения. Контекстное меню.
39. Использование графики. Вспомогательные графические классы. Возможности класса TCanvas.
40. Классы TGraphic, TPicture, компоненты для работы с графикой.
41. Работа с буфером обмена. Работа с принтером.
42. Таблицы строк и таблицы изображений.
43. Классы TComponent, TPersistent. Динамическое создание компонент.
44. Обработка исключительных ситуаций. Защищенные блоки.
45. Организация многопоточных приложений.
46. Технологии OLE. Пример создания OLE-контейнера.
47. COM – технологии. Пример OLE-контроллера. Автоматизация MS Office. Раннее и позднее связывание.
48. Средства Delphi для создания SDI-приложений.

49. Средства Delphi для создания MDI-приложений.
50. Программный интерфейс ОС Windows. Понятие API-интерфейса. Основные модули ядра Windows и их функции.
51. Динамически компонуемые библиотеки, их создание и использование в Delphi.

Литература :

1. Букунов, С.В. Основы объектно-ориентированного программирования [Электронный ресурс]: учебное пособие/ С.В. Букунов, О.В. Букунова. — Электрон. текстовые данные. — СПб.: Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2017. — 196 c. — 978-5-9227-0713-8. — Режим доступа: http://www.iprbookshop.ru/74339.html
2. Мейер, Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Б. Мейер. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 285 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/39552.html
3. Николаев, Е.И. Объектно-ориентированное программирование [Электронный ресурс]: учебное пособие/ Е.И. Николаев. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2015. — 225 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/62967.html
4. Николаев, Е.И. Объектно-ориентированное программирование. Часть 1 [Электронный ресурс]: лабораторный практикум/ Е.И. Николаев. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2015. — 183 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/62966.html
5. Николаев, Е.И. Объектно-ориентированное программирование. Часть 2 [Электронный ресурс]: лабораторный практикум/ Е.И. Николаев. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2015. — 156 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/63218.html
6. Новиков, П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ П.В. Новиков. — Электрон. текстовые данные. — Саратов: Вузовское образование, 2017. — 124 c. — 978-5-4487-0011-8. — Режим доступа: http://www.iprbookshop.ru/64650.html
7. Пышкин, Е.В. Основы концепции и механизмы объектно-

ориентированного программирования [Текст]: учебник/ Е.В. Пышкин.- СПб.: БХВ-Петербург, 2005.- 640 с.
8. Сорокин, А.А. Объектно-ориентированное программирование. LAZARUS (Free Pascal) [Электронный ресурс]: лабораторный практикум/ А.А. Сорокин. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2014. — 216 с. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/63109.html
9. Хореев, П.Б. Технология объектно-ориентированного программирования [Текст]: учеб. пособие для студ. высш. учеб. заведений/ П.Б. Хорев.- 2-е изд., стер.- М.: Академия, 2008.- 448 с.