

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторных работ
по дисциплине
«ПРАКТИКУМ ПО РЕШЕНИЮ ЗАДАЧ НА ЭВМ»
для студентов направления подготовки
44.03.05 Педагогическое образование (с двумя профилями подготовки)
направленность (профиль):
«Информатика и математика»

Ставрополь, 2026

Содержание

ВВЕДЕНИЕ	Ошибка! Закладка не определена.
СТРУКТУРА И СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ ЗАНЯТИЙ	Ошибка! Закладка не определена.
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА	89

ВВЕДЕНИЕ

Цель и задачи освоения дисциплины

Цели дисциплины: формирование набора универсальных и общепрофессиональных компетенций будущего бакалавра (специалиста) по соответствующему направлению подготовки.

Задачи дисциплины: формирование умений и навыков применения методов информатики для решения задач с использованием персонального компьютера.

В результате освоения дисциплины студенты должны:

Знать:

- базовые определения информатики, основные и составные структуры данных, используемые в компьютерных технологиях;
- основы организации современных ПК и их общие характеристики;
- тенденции развития устройств компьютера и компьютерных сетей;
- понятия алгоритма решения задачи и языков программирования;
- возможности сети Интернет для поиска и обработки данных и организации информационного обмена;
- проблемы информационной безопасности компьютерных систем и методы защиты информации.

Уметь:

- ориентироваться в многообразии информационных процессов и средств их обработки;
- применять разнообразные методы для решения определенного класса задач;
- эффективно использовать возможности современных ПК, компьютерных сетей и программных средств для решения прикладных задач, возникающих в процессе обучения в вузе и в ходе будущей профессиональной деятельности;
- использовать знания по теории информации и теории кодирования в профессиональной деятельности.

Владеть:

- навыками работы с основными устройствами ПК;
- способами измерения и представления различных типов информации;
- основами вычислительной и алгоритмической культуры;
- приемами и методами кодирования информации;

навыками использования в профессиональной деятельности базовых знаний в области информатики и современных информационных технологий.

СТРУКТУРА И СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ ЗАНЯТИЙ

Лабораторная работа 1

Основные понятия среды Lazarus

Среда программирования Lazarus состоит из нескольких отдельно расположенных окон.

Окна могут перемещаться по экрану, частично или полностью перекрывая друг друга. После запуска Lazarus перед вами появятся основные окна (рис. 1):

1. Главное окно (Project1).
2. Окно формы (Form1).
3. Окно Инспектора Объектов (Object Inspector).
4. Окно кода программы или окно редактора кода (Unit1.pas).

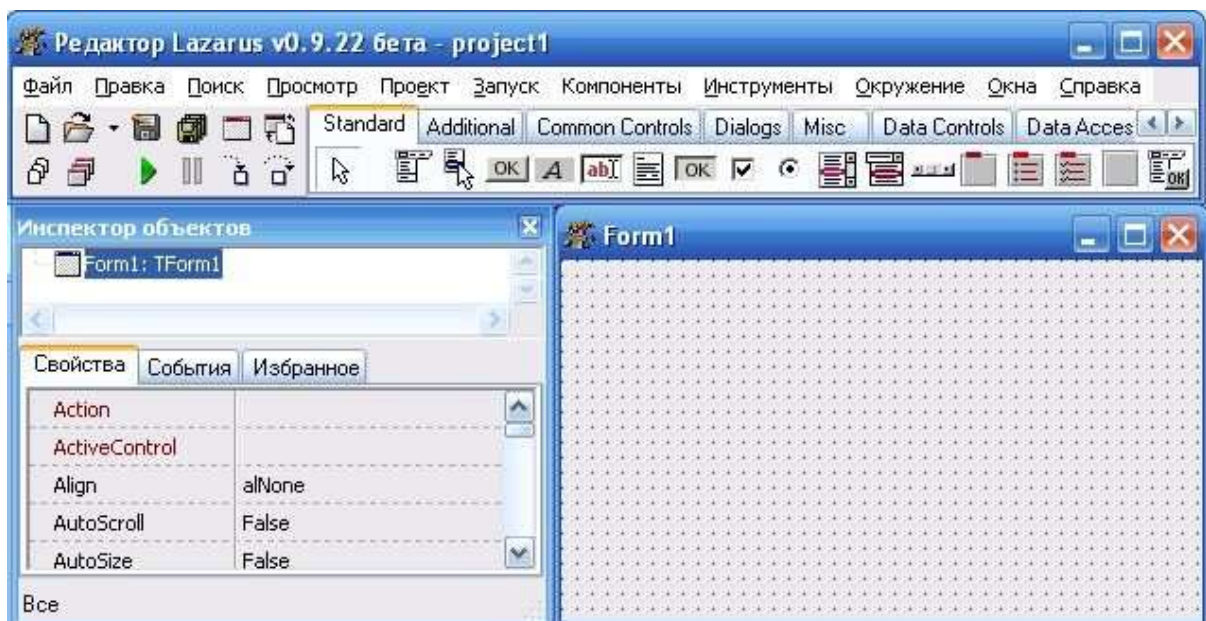


Рис. 1.

Первоначально окно кода перекрыто окном формы, для переключения между которыми осуществляется клавишей F12.

Lazarus – объектно-ориентированная среда

Объектно-ориентированное программирование – это методика разработки программ, в основе которой лежит понятие объекта.

Объект – это совокупность свойств (параметров) определенных сущностей и методов их обработки (программных средств).

Описание объекта рассмотрим на примере объекта автомобиль.

Каждый объект имеет:

Свойства – характеристики объекта (автомобиль характеризуется маркой, цветом, салоном, местонахождением руля, коробкой передач и т.д.).

Методы – действия объекта, что объект может делать (для автомобиля действиями являются движение вперед, назад, торможение, остановка, поворот и др.).

События – изменения в окружающей объект обстановке (события, которые влияют на объект автомобиль: сигнал светофора, знаки милиционера и др.).

Реакция на события – описания действий, которые необходимо совершить при данном событии.

Все объекты реализованы в Lazarus в виде палитры визуальных компонентов (рис. 2).



Рис. 2.

Компоненты сгруппированы в отдельные страницы, каждая из которых снабжена закладкой. Если щелкнуть мышью на одну из закладок, то можно перейти на соответствующую ей страницу. При подведении курсора мыши к компоненте (или нажатие правой кнопки мыши на компоненте) появляется подсказка – название компоненты.

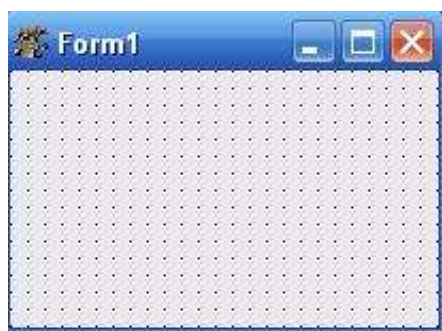


Рис. 3.

Форма. Инспектор объектов. Окно кода программы

Форма – окно будущей программы (рис.3)

. Окно формы представляет собой проект окна на Windows: имеет заголовок, кнопку вызова системного меню, кнопку максимизации, минимизации и закрытия окна, рамку окна. На форме

размещаются компоненты, тем самым, формируя заготовку (макет) окна будущего приложения.

Окно **Инспектора Объектов** содержит страницы: *Свойства (Properties)* и *События (Events)*, каждая из которых разделена на две части.

Страница *Свойства* служит для указания свойств объекта. На этой странице указаны все свойства активного (выделенного) объекта. В левой части страницы находится название свойства, а в правой – его значение. «+» слева от названия свойства указывает на то, что свойство состоит из нескольких значений. Значениями свойств могут быть слова, числа,

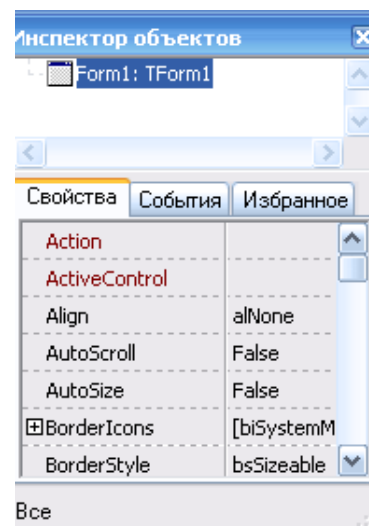


Рис. 4.

значения из раскрывающегося списка. Нажав на «▼» справа от поля значения свойства, можно получить раскрывающийся список значений. При нажатии на «...» вызывается специальное диалоговое окно.

Если значением свойства является число или текст, то после его набора необходимо нажать *Enter*, иначе оно может быть не зафиксировано. При нажатии на *Esc* ввод отменяется.

Страница *События* используется для задания реакции на событие. Страница также состоит из двух частей. В первой – название события, а во второй – название процедур, обрабатывающих данное событие. Если в правой части ничего не написано, то программа на данное событие не реагирует. Название процедуры состоит из двух частей: названия объекта + название события.

Для создания новой реакции на событие необходимо дважды щелкнуть в правой половине напротив нужного события. В этом месте появится название новой процедуры, а на экране автоматически открывается окно редактора кода.

Окно кода программы предназначено для создания и редактирования текста программы (рис. 5). Этот текст составляется по определенным правилам.

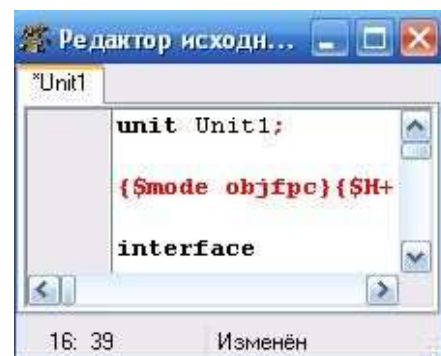


Рис. 5.

Для написания программы необходимо:

1. Разместить необходимые компоненты на форме.
2. Задать свойства выбранных компонент.
3. Определить событие.
4. Задать реакцию на событие.

Для размещения компонент на форме необходимо щелкнуть мышью на компоненте, после щелкнуть на форме.

Краткая характеристика некоторых компонентов

У всех компонент есть свойство *Name* – имя компонента. Имена создаются средой Lazarus по такому принципу: сначала идет название компонента (*Form*), а за ним порядковый номер размещенного на форме компонента (1). То есть если добавить еще одну форму, то она получит имя *Form2*, следующее – *Form3* и т.д. Имя, заданное по умолчанию можно изменить, но при этом желательно использовать только английские буквы и цифры.

Запомните! Свойство *Name* задается первоначально, это имя переменной с которой вы будете работать при написании программы.

1. *Компонент Form* (экранная форма). Форма представляет не только внешний вид окна приложения, но и сама является полноценным компонентом с собственными свойствами и событиями, хотя на палитре компонентов ее нет.

Таблица 1

Основные свойства компонента *Form*

<i>Caption</i>	Заголовок окна формы
<i>Color</i>	Цвет формы
<i>Font</i>	Атрибуты шрифта заголовка окна формы: Color – цвет шрифта Size – размер шрифта Style – стиль шрифта

Таблица 2

Основные события компоненты *Form*

<i>OnCreate</i>	Создание формы
<i>OnClose</i>	Закрытие формы

2. *Компонент Label* (надпись или метка) . Можно использовать для вывода ответа или пояснения вводимых данных.

Таблица 3

Основные свойства компоненты *Label*

<i>Caption</i>	Заголовок надписи, выводимой на форму
<i>AutoSize</i>	Позволяет автоматически менять размеры метки, чтобы соответствовать размерам надписи (по умолчанию принимает значение True).
<i>Font</i>	Шрифт, используемый для отображения текста в метке
<i>Visible</i>	Видимость надписи на экране. Имеет два значения: True – надпись видна на форме, False – надпись не видна на форме.
<i>Alignment</i>	Выравнивание текста по горизонтали относительно границ: taRightJustify – прижат к правой границе taCenter – выровнен по центру taLeftJustify – прижат к левой границе

3. *Компонент Edit* (однострочное окно ввода/вывода) . Используется для ввода/вывода чисел и текста в программу.

Таблица 4

Основные свойства компонента *Edit*

<i>BorderStyle</i>	Стиль обрамления: bsNone – нет обрамления bsSingle – обрамление одной линией
<i>MaxLength</i>	Определяет максимальную длину текстовой строки. Если имеет значение 0, длина строки не ограничена
<i>Text</i>	Содержимое строки редактирования

Таблица 5

Основные методы компонента *Edit*

<i>Clear</i>	Удаляет весь текст
<i>SetFocus</i>	Устанавливает фокус ввода

4. *Компонент Panel* (контейнер общего назначения) . Используется для размещения на ней сгруппированных компонент.

Таблица 6

Основные свойства компонента *Panel*

<i>BevelOuter</i>	Внешняя рамка	bvRaised – выпуклая bvLowered – вдавленная bvNone – отсутствует
<i>BevelInner</i>	Внутренняя рамка	
<i>BevelWidth</i>	Ширина рамки	

5. *Компонент Button* (командная кнопка) . Используется для задания реакции на событие.

Таблица 7

Основные свойства компонента *Button*

<i>Caption</i>	Надпись на кнопке
<i>Height</i>	Высота кнопки
<i>Width</i>	Ширина кнопки

Основные события компонента *Button*

<i>OnClick</i>	Происходит, когда пользователь щелкает основной (левой) кнопкой мыши на объекте
<i>OnDbClick</i>	Происходит, когда пользователь дважды щелкает основной (левой) кнопкой мыши на объекте
<i>OnClose</i>	Закрытие окна формы

Задание 1. Изменить заголовок окна формы с *Form1* на «Первая программа», используя свойство *Caption*.

Задание 2. Изменить цвет формы на *clAqua*, используя свойство *Color*.

Задание 3. Разместить в центре формы компоненту *Label*. Задать: надпись метки – СВЕТОФОР, цвет метки – серый. Изменить свойство *Font*: шрифт – Times New Roman, начертание – жирный, размер – 14.

Задание 4. Расположить на форме компоненты *Panel1*, *Panel2*, *Panel3*, для которых поочередно задать свойство *Caption* пустым.

Задание 5. Расположить на форме три командные кнопки *Button1*, *Button2*, *Button3*. Задать надписи на этих кнопках «Красный», «Желтый», «Зеленый». В результате должна получиться форма, согласно рисунка 6.



Рис. 6.

События. Реакция на события. Окно кода программы

События устанавливаются в окне *Инспектора Объектов* на странице *События (Events)*.

Задание 7. Для изменения цвета компоненты *Panel1* на красный необходимо выполнить следующую процедуру:

```

Procedure TForm1.Button1Click (Sender:TObject);
Begin
    Panel1.Color:=clRed;

```

```
Panel2.Color:=clWhite;  
Panel3.Color:=clWhite;  
End;
```

Задание 8. Создать событие и реакцию на событие для командной кнопки *Button2*: цвет компоненты *Panel2* будет меняться на желтый, а цвет компонент *Panel1* и *Panel3* будет меняться на белый. Если вы не знаете, как записать название цвета, посмотрите возможные цвета свойства *Color* в *Инспекторе Объектов*.

Задание 9. Создать событие и реакцию на событие для командной кнопки *Button3*: цвет компоненты *Panel3* будет меняться на зеленый, а цвет компонент *Panel1* и *Panel2* будет меняться на белый.

Запуск и сохранение программы

Перед первым запуском программы ее необходимо сохранить.

Проект Lazarus представляет собой набор программных единиц – модулей, которые хранятся в отдельных файлах. Поэтому каждую программу сохраняют в отдельной папке.

Для сохранения программы надо выбрать *Файл* → *Сохранить как*. Появится диалоговое окно *Сохранить Unit1 (*.pas)*, в котором на первом этапе сохраняется программа с расширением *.pas*.

После задания имени программы и нажатия кнопки *Сохранить* появляется диалоговое окно *Переименовать файл?* (рис. 7).

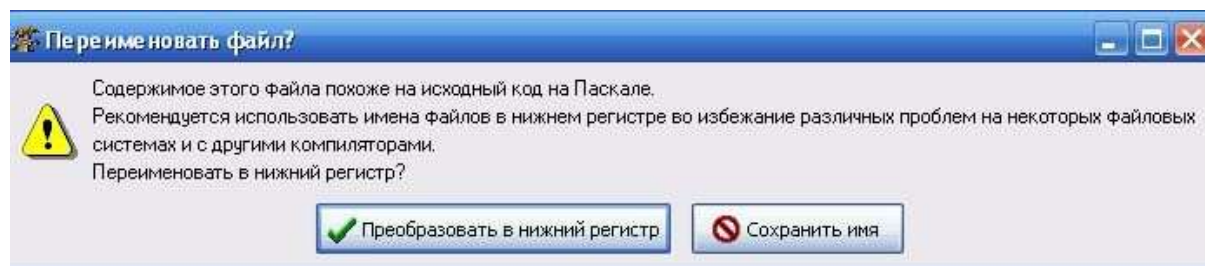


Рис. 7

В данном окне надо нажать кнопку *Преобразовать в нижний регистр*.

Далее появляется диалоговое окно *Сохранить проект project1 (*.lpi)*, в котором сохраняется проект, работающим с программой, с расширением *.lpi*.

Запомните! Чтобы открыть программу, необходимо открыть проект с расширением *.lpi*.

В среде Lazarus программу можно сохранить и командой *Файл* → *Сохранить все*.

Если после первого сохранения программы в нее были внесены изменения, то для их сохранения можно нажать на панели инструментов кнопку  или выбрать команду *Сохранить все*.

Запустить программу на исполнение можно одним из следующих способов:

1. Выбрать команду *Запуск* → *Запуск* (*Run* → *Run*).
2. Нажать клавишу F9.
3. Нажать кнопку  на панели инструментов.

После запуска программы она может зависнуть. Для остановки зависшей программы необходимо нажать комбинацию клавиш *Ctrl+F2* или выбрать команду *Запуск* → *Остановить*.

Задание 10. На диске *C:* создать папку с номером вашей группы (в скобках указать номер подгруппы). В папке с номером группы создать папку с именем *Первая программа*.

Задание 11. Сохранить свою программу в папке *Первая программа*.

Задание 12. Добавить появление на компоненте *Panel1* при нажатии на командную кнопку *Button1* информации «СТОЙТЕ», цвет – белый, начертание – жирный, размер шрифта – 12.

Для этого необходимо в имеющуюся процедуру добавить следующие действия:

```
Procedure TForm1.Button1Click (Sender:TObject);
```

```
Begin
```

```
    Panel1.Color:=clRed;
```

```
    Panel2.Color:=clWhite;
```

```
    Panel3.Color:=clWhite;
```

```
    Panel1.Caption:='СТОЙТЕ';
```

```
{задание на панели надписи}
```

```
    Panel1.Font.Color:=clWhite;
```

```
{задание цвета шрифта}
```

```
    Panel1.Font.Size:=12;
```

```
{задание размера шрифта}
```

```
    Panel1.Font.Style:=[fsBold];
```

```
{задание начертания шрифта}
```

```
End;
```

Задание 13. Добавить появление на компоненте *Panel2* при нажатии на командную кнопку *Button2* информации «ВНИМАНИЕ», цвет – белый, начертание – жирный, размер шрифта – 12.

Задание 14. Добавить появление на компоненте *Panel3* при нажатии на командную кнопку *Button3* информации «ИДИТЕ», цвет – белый, начертание – жирный, размер шрифта – 12.

Задание 15. Сохранить изменения в программе и запустить ее на исполнение.

Окна сообщения

При работе с любой программой можно выводить окна сообщения. Для этого существует команда:

ShowMessage('текст сообщения');

Задание 16. На форме добавить командную кнопку *Button4*. Задать для нее надпись «Автор». При нажатии на кнопку должно выводиться сообщение об авторе программы. Для реализации данного задания задать для нее реакцию на событие :

```
Procedure TForm1.Button4Click(Sender:TObject);  
begin  
  ShowMessage('Программа разработана Ивановым С.');
```

```
end;
```

Задания для самостоятельного выполнения

Составьте программу, которая при выборе соответствующей командной кнопки, изменяет:

1. Содержание надписи на «Язык программирования», «Система программирования», «Алгоритм»; дополнительно выводится сообщение «Спасибо за работу»при завершении работы программы.

2. Цвет формы на голубой, фиолетовый, серый; дополнительно выводится сообщение при запуске программы об условии задачи.

3. Вид отображения рамки панели «выпуклая», «вдавленная», «отсутствует»; дополнительно выводится сообщение о назначении компонентов при щелчке мыши на данной компоненте.

Программирование задач линейной структуры

При программировании задач линейной структуры необходимо знать основные типы данных, операции и функции их обработки.

Среди существующих типов данных можно выделить три основные:

1. *Целые типы* данных используются для представления целых чисел. Есть несколько различных целых типов, которые могут хранить целые значения. Различные целые типы имеют существенно различные диапазоны хранимых значений, затраты памяти растут с ростом допустимого диапазона значений. Наиболее распространенный из них – тип *Integer*.

2. *Вещественные типы* данных предназначены для хранения чисел, имеющих дробную часть. Наиболее распространенный из них – тип *Real*.

3. *Логический тип* данных – один из простейших и часто используемых. Примером логического типа является *Boolean*. Переменные типа *Boolean* представляют собой логические значения *True* или *False*.

В программе каждая переменная перед использованием должна быть описана. С помощью описания устанавливается факт существования переменной, задается ее тип.

Для описания переменных существует раздел **var**.

Var имя переменной: тип данных;

Пример.

var a: real;

i: integer;

Если в программе несколько переменных одного типа, то можно через запятую перечислить имена переменных, относящихся к одному типу, и после имени последней переменной через двоеточие указать тип.

Пример.

var a,b,c: real;

x1,x2: integer;

Для присваивания переменной вычисленного значения используется оператор присваивания. Это наиболее часто используемый оператор языка. Синтаксис оператора присваивания:

Имя переменной := выражение;

Символы «:=» всегда пишутся слитно, без разделяющих пробелов, хотя перед двоеточием и после знака равенства можно для лучшей читаемости программы вставлять пробелы. Как и любой другой оператор языка, оператор присваивания завершается точкой с запятой. Переменные и результат выражения должны быть одного типа.

Выражение состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций, например $a+b*\sin(\cos(x))$.

При вычислении значений выражений соблюдается приоритет операций. Порядок выполнения операций – слева направо, при этом сначала выполняются действия в скобках, после умножение, деление, арифметические функции, а дальше сложение, вычитание. Числа записываются при помощи цифр, причем целая часть от дробной отделяется точкой.

Таблица 9

Арифметические операции

Операция	Действие	Тип операндов	Тип результата	Пример
+	сложение	целый/вещественный	целый/вещественный	$4+5 = 9$
+	сцепление строк	строковый	строковый	
–	вычитание	целый/вещественный	целый/вещественный	$9 - 4 = 5$
*	умножение	целый/вещественный	целый/вещественный	$2 * 2 = 4$
/	деление	целый/вещественный	вещественный	$10/4 = 2.5$
div	целочисленное деление	целый	целый	$9 \text{ div } 4 = 2$
mod	остаток от деления	целый	целый	$9 \text{ mod } 4 = 1$

Таблица 10

Арифметические функции

Обозначение	Действие	Тип аргументов	Тип результата	Пример
abs(x)	модуль числа	целый/вещественный	целый/вещественный	$\text{abs}(-5.3)=5.3$
sqr(x)	квадрат числа	целый/вещественный	целый/вещественный	$\text{sqr}(4)=16$
sqrt(x)	квадратный корень	целый/вещественный	вещественный	$\text{sqrt}(4)=2$
cos(x)	косинус	вещественный	вещественный	
sin(x)	синус	вещественный	вещественный	

arctan(x)	арктангенс	вещественный	вещественный	
exp(x)	экспонента e^x	вещественный	вещественный	
ln(x)	натуральный логарифм	вещественный	вещественный	
pi	число π	вещественный	без аргумента	
int(x)	целая часть числа	вещественный	вещественный	int(5.4)=5
frac(x)	дробная часть числа	вещественный	вещественный	frac(5.6)=0,6
round(x)	округление числа	вещественный	целый	round(4.5)=5
trunc(x)	отсекание дробной части числа	вещественный	целый	trunc(4.5)=4
random(n)	случайное число от 0 до n	целый	целый	random(7)=4
exp(n*ln(x))	возведение числа x в степень n (x^n)	целый/вещественный	вещественный	exp(5*ln(x))=x5
Функции, определенный в программном модуле Math*				
arccos(x)	арккосинус	вещественный	вещественный	
arcsin(x)	арксинус	вещественный	вещественный	
arccot(x)	арккотангенс	вещественный	вещественный	
log10(x)	десятичный логарифм	вещественный	вещественный	log10(10)=1
log2(x)	логарифм по основанию 2	вещественный	вещественный	log2(4)=2
logn(n,x)	логарифм от x по основанию n	вещественный	вещественный	logn(3,27)=3

* – эти функции будут работать только в том случае, если в тексте основной программы после ключевого слова *uses* (список модулей) указать имя *Math*.

Пример. uses **Math**, Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, StdCtrls;

Задание 1. Вычислить значение функции $f = \frac{1}{x+1)^2 + 2} + x^5 - \sqrt{x}$.

Решение любой задачи начинается с определения входных и выходных переменных также определения типа переменных.

входные переменные: x : integer;

выходные переменные: f : real.

Размещаем компоненты на форме (рис. 8).

Свойства выбранных компонент:

Form1 – Caption – Значение функции

Label1 – Caption – Введите значение x

Label2 – Caption – Значение функции f

Button1 – Caption – Вычислить

Edit1 – Text – пусто

Edit2 – Text – пусто

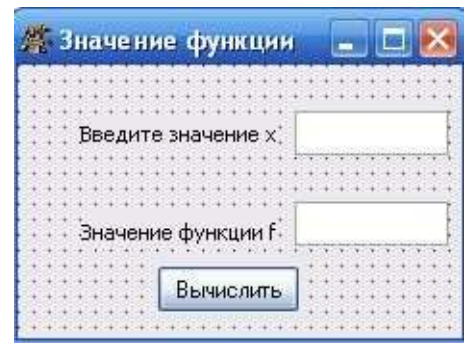


Рис. 8.

При написании процедур необходимо помнить, что все вводимые данные воспринимаются как строки. Поэтому необходимо переводить вводимые числовые данные из строки в числа, а выводимый результат переводить из числа в строку.

StrToInt – перевод строки в целое число

StrToFloat – перевод строки в вещественное число

IntToStr – перевод целого числа в строку

FloatToStr – перевод вещественного числа в строку

Процедура и блок-схема (рис. 9) для вычисления значения функции будет иметь ниже приведенный вид (обратите внимание, что при написании программы все служебные слова выделяются автоматически жирным шрифтом):

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

```
var x:integer;
```

```
    f:real;
```

```
begin
```

```
    x:=strtoint(edit1.text);
```

```
    f:=1/(sqr(x+1)+2)+exp(5*ln(x))-sqrt(x);
```

```
    edit2.text:=floattostr(f);
```

```
end;
```

Задание 2. Добавить на форму командную кнопку *Button2*. Задать для нее надпись *Очистить*. Задать для кнопки *Очистить* реакцию на событие – при нажатии на кнопку должны очищаться компоненты *Edit1* и *Edit2* и курсор помещаться в компоненту *Edit1*.

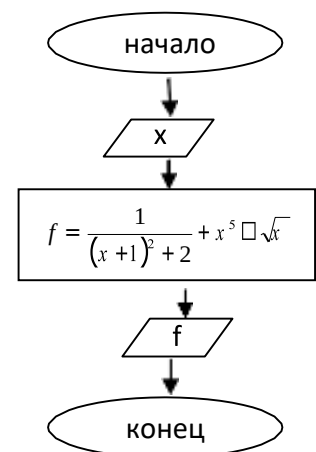


Рис. 9.

Задание 3. Известны длины сторон треугольника a , b и c (рис. 10). Вычислить площадь S , периметр P и величины углов α , β и γ (в градусах) треугольника.

Для вычисления площади треугольника применим формулу Герона: $S = \sqrt{r(r-a)(r-b)(r-c)}$, где $r = \frac{a+b+c}{2}$ – полупериметр.

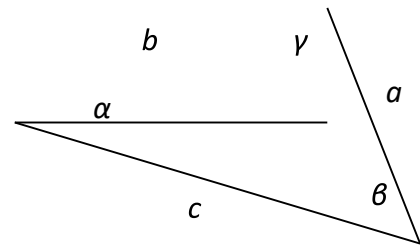


Рис. 10.

Первый угол α найдем по теореме косинусов: $\cos \alpha = \frac{a^2 + b^2 - c^2}{2ab}$.

Второй угол β найдем по теореме синусов: $\sin \beta = \frac{b}{a} \sin \alpha$

Третий угол γ найдем по формуле: $\gamma = 180^\circ - (\alpha + \beta)$

Необходимо помнить, что найденные углы по приведенным формулам будут вычислены в радианах. Для перевода радиан в градусы надо воспользоваться формулой: $x = \frac{x \cdot 180}{\pi}$, где x – угол в радианах.

Определим в программе входные и выходные переменные:

входные переменные: a , b , c (стороны): integer;

выходные переменные: α , β , γ (углы), S (площадь), r (полупериметр): real; P (периметр): integer.

Размещаем компоненты на форме (рис. 11).

Свойства выбранных компонент:

Form1 – Caption – Параметры треугольника

Label1 – Caption – Введите длины сторон

Label2 – Caption – a=

Label3 – Caption – b=

Label4 – Caption – c=

Label5 – Caption – Величины углов

Label6 – Caption – alpha=

Label7 – Caption – beta=

Label8 – Caption – gamma=

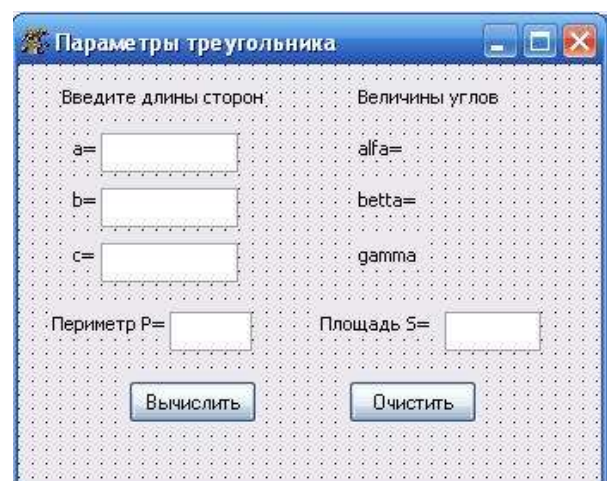


Рис. 11.

Label9 – Caption – Периметр P=
 Label10 – Caption – Площадь S=
 Button1 – Caption – Вычислить
 Button2 – Caption – Очистить
 Edit1 – Text – пусто

Edit2 – Text – пусто
 Edit3 – Text – пусто
 Edit4 – Text – пусто
 Edit5 – Text – пусто

Перед написанием процедуры необходимо в список модулей добавить модуль *Math*.

Процедура и блок-схема (рис. 12) для кнопки *Вычислить* будет иметь вид:

procedure TForm1.Button1Click(Sender: TObject);

var a,b,c,P:integer;

alfa,betta,gamma,S,r:real;

begin

a:=strtoint(edit1.text);

b:=strtoint(edit2.text);

c:=strtoint(edit3.text);

P:= a+b+c;

r:=P/2;

S:=sqrt(r*(r-a)*(r-b)*(r-c));

alfa:=arccos((sqr(b)+sqr(c)-
 sqr(a))/(2*b*c));

betta:=arcsin(b/a*sin(alfa));

gamma:=pi-(alfa+betta);

alfa:=alfa*180/pi;

betta:=betta*180/pi;

gamma:=gamma*180/pi;

Label6.caption:='alfa='+floattostr(alfa);

Label7.caption:='betta='+floattostr(betta);

Label8.caption:='gamma='+floattostr(gamma);

Edit4.text:=inttostr(P);

Edit5.text:=floattostr(S);

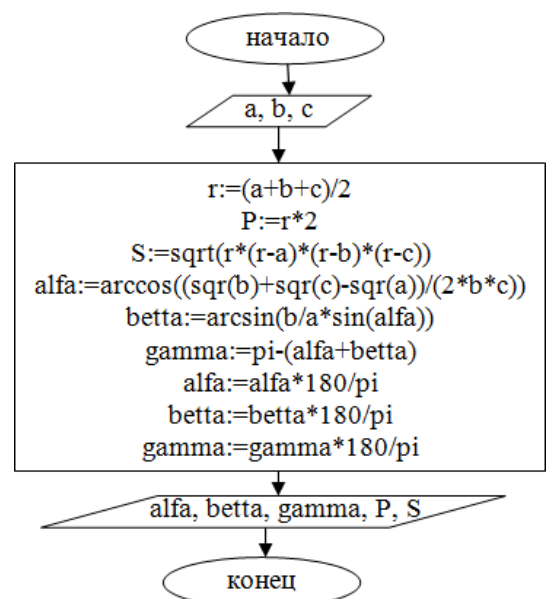


Рис. 12.

end;

Процедура для кнопки *Очистить* будет иметь вид:

procedure TForm1.Button2Click(Sender: TObject);

begin

edit1.clear;

edit2.clear;

edit3.clear;

edit4.clear;

edit5.clear;

edit1.setfocus;

label6.caption:='alfa=';

label7.caption:='beta=';

label8.caption:='gamma=';

end;

Задания для самостоятельного выполнения

1. Найти значение функции (все переменные принимают действительные значения).

2. Решить задачу.

Вариант	Задание
1	1. $f = \frac{1 + \sin^2(x + y)}{2 + \left x - \frac{2x}{1 + x^2 y^2} \right } + x.$ 2. В прямоугольном треугольнике известны катет b и площадь S. Найти величину гипотенузы c, второго катета a и углов и
2	1. $f = \ln \left \left(y \cdot \sqrt{ x } \right)^x \cdot \frac{y}{x + \frac{x^2}{4}} \right .$ 2. Даны две стороны треугольника и угол между ними. Определить третью сторону, площадь треугольника и радиус описанной окружности.

3	1. $f = \frac{x+1}{x-1} + 18xy^2$. 2. Известны координаты вершин квадрата $ABCD$: $A(x_1, y_1)$ и $C(x_2, y_2)$. Найти его площадь и периметр.
4	1. $f = \sqrt{\frac{a+b}{a^2+b^2}} - ab^3$. 2. Известна площадь квадрата S. Вычислить сторону квадрата a, диагональ d и площадь круга S_1, описанного вокруг квадрата.
5	1. $f = e^{at} \sin(at+b) - \sqrt{bt+a}$ 2. В равнобедренном треугольнике известно основание c, угол при основании равен α. Найти площадь треугольника S и величину боковой стороны a.
6	1. $f = \frac{a_2^x + e^{bx} \cos(bx)}{bx - e^{bx} \sin(bx) + 1}$. 2. Вычислить диагональ и площадь прямоугольника, вписанного в окружность радиуса R, если отношение его сторон равно n.
7	1. $f = x^3 \lg^2(x+b)^2 + \frac{a}{\sqrt{x+b}}$. 2. Задан периметр квадрата P. Вычислить сторону квадрата a, диагональ d и площадь S.
8	1. $f = 2 \operatorname{ctg}(3x) - \frac{\ln \cos x}{\ln(1+x^2)}$. 2. Задан первый член a_1 и разность d арифметической прогрессии. Вычислить сумму n членов арифметической прогрессии и значение n-го члена.

Лабораторная работа №3

Оператор ветвления

Ветвление – это алгоритмическая структура, в которой проверяется некоторое условие и в зависимости от результатов проверки выполняется то или иное действие.

Для программирования проверки условия и выбора действия в зависимости от этого условия используются *условные операторы*.

Условие в языке программирования может принимать два значения: истина (*True*) и ложь (*False*). Условие записывается с помощью операций отношений (табл. 11) и логических операторов (табл. 12).

Таблица 11

Операции отношения

Оператор	Действие	Оператор	Действие
>	Больше	<>	Не равно
<	Меньше	>=	Больше или равно
=	Равно	<=	Меньше или равно

Таблица 12

Логические операторы

Оператор	Действие	Оператор	Действие
and	логическое И	or	логическое НЕ
not	логическое НЕ	xor	логическое исключающее ИЛИ

Использование операторов сравнения позволяет записывать простые условия. Из простых условий, которые являются выражениями логического типа, можно строить сложные условия с применением к ним, как к операндам, логических операторов.

Примеры задания условий с помощью операций отношений и логических операторов представлены на рисунке 13.

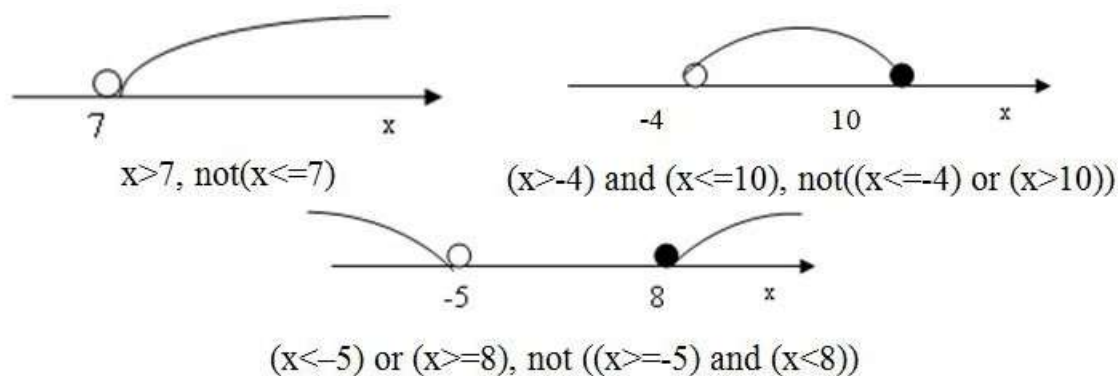


Рис. 13.

Существует две формы условного оператора:

1. Полное ветвление (развилка).

Синтаксис:

If <i>условие</i> Then <i>оператор1</i> Else <i>оператор2</i> ;	если <i>условие</i> то <i>действие 1</i> иначе <i>действие 2</i> все	} <i>запись полного ветвления выбора на алгоритмическом языке</i>
--	---	---

После *условие* и *оператора1* «;» не ставится.

Пример.

```
If  $x \geq 4$   
Then  $y := \text{sqr}(x)$   
Else  $y := \text{abs}(x)$ ;
```

Иногда после служебных слов *Then* или *Else* может стоять более одного оператора. Тогда необходимо использовать *составной оператор*.

Составной оператор может состоять из произвольного количества операторов, разделенных между собой «;».

Синтаксис:

```
Begin  
  Оператор1;  
  Оператор2;  
  .....  
  Оператор n;  
End;
```

Пример.

```
If  $(x \geq 4) \text{ and } (x \leq 5)$   
Then Begin  
   $y := \text{sqr}(x)$ ;  
   $z := \text{sqrt}(x)$ ;  
end  
Else  $y := x + 8$ ;
```

2. Неполное ветвление.

Синтаксис:

If условие
Then оператор;

если условие
то действие
все

запись неполного ветвления
выбора на алгоритмическом языке

Блок-схемы полного и неполного ветвлений представлены на рисунках 14 и 15.

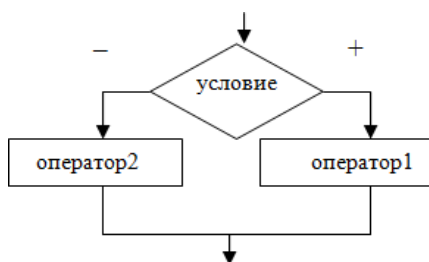


Рис. 14.

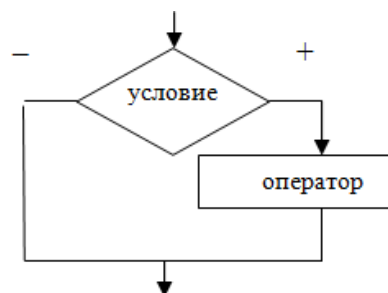


Рис. 15

Задание 1. Составить программу нахождения действительных и комплексных корней квадратного уравнения $ax^2 + bx + c = 0$.

Определим переменные задачи:

входные переменные: a, b, c (коэффициенты уравнения): real;

промежуточные переменные d (дискриминант), y_1, y_2 (вспомогательные переменные): real;

результат: x_1, x_2 (корни уравнения): real.

Можно выделить следующие этапы решения задачи:

- 1) Ввод коэффициентов квадратного уравнения a, b и c .
- 2) Вычисление дискриминанта d по формуле $d = b^2 - 4ac$.
- 3) Проверка знака дискриминанта.

Если $d \geq 0$, то корни уравнения действительные и находятся по формулам:

$$x_{1,2} = \frac{-b \pm \sqrt{d}}{2a}.$$

Если $d < 0$, то корни уравнения комплексные и находятся по формулам:

$$x_{1,2} = \frac{-b \pm \sqrt{|d|}}{2a}.$$

Для нахождения корней будем использовать вспомогательные перемен-

ные: $y1 = \frac{-b}{2a}$, $y2 = \frac{\sqrt{|d|}}{2a}$. Тогда корни уравнения примут вид:

действительные – $x1 = y1 + y2$, $x1 = y1 - y2$;

комплексные – $x1 = y1 + iy2$, $x1 = y1 - iy2$

Размещаем компоненты на форме (рис. 16).

Свойства выбранных компонент:

Form1 – Caption – Решение квадратного уравнения

Label1 – Caption – Введите коэффициенты уравнения

Label2 – Caption – Коэффициент a=

Label3 – Caption – Коэффициент b=

Label4 – Caption – Коэффициент c=

Label5 – Caption – Корни уравнения

Label6 – Caption – x1=

Label7 – Caption – x2=

Button1 – Caption – Вычислить

Button2 – Caption – Очистить

Edit1...5 – Text – пусто



Рис. 16.

Процедура и блок-схема (рис. 17) для вычисления значения функции будет иметь вид:

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

```
var a,b,c,d,x1,x2,y1,y2:real;
```

```
begin
```

```
a:=strtoint(edit1.text);
```

```
b:=strtoint(edit2.text);
```

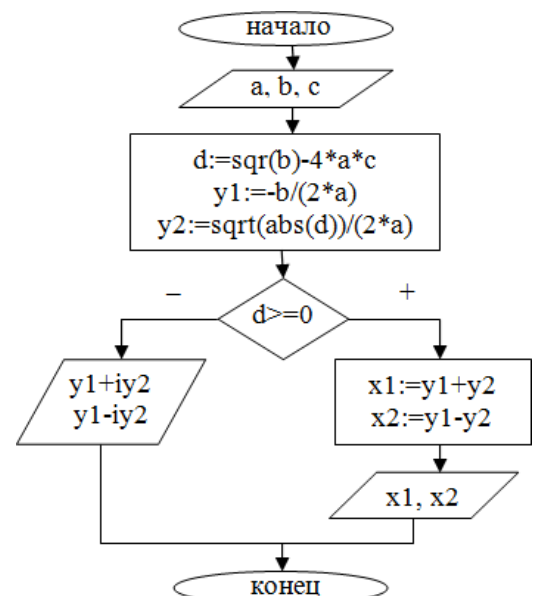
```
c:=strtoint(edit3.text);
```

```
d:=sqr(b)-4*a*c;
```

```
y1:=-b/(2*a);
```

```
y2:=sqrt(abs(d))/(2*a);
```

```
if d>=0
```



then
begin

Puc. 17.

```

x1:=y1+y2;
x2:=y1-y2;
edit4.text:=floattostr(x1);
edit5.text:=floattostr(x2);
end
else
begin
edit4.text:=floattostr(y1)+'+i'+floattostr(y2);
edit5.text:=floattostr(y1)+'-i'+floattostr(y2);
end;
end;
end;

```

Задание 2. Процедуру для кнопки «Очистить» написать самостоятельно.

Задание 3. Дано вещественное число x . Для функции, представленной графиком, вычислить $y = f(x)$ (рис. 18).

Определим переменные задачи:

входные переменные: x : real;

результат: y : real.

Аналитически функцию, представленную на рисунке 18, можно записать следующим образом:

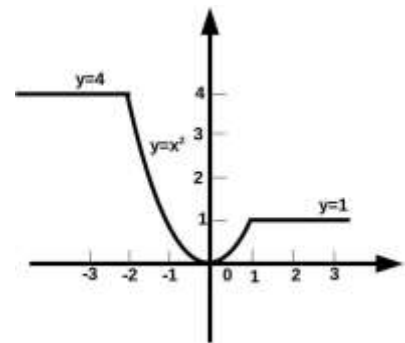


Рис. 18.

$$y = \begin{cases} 4, & x < -2 \\ x^2, & -2 \leq x < 1 \\ 1, & x \geq 1 \end{cases}$$

Размещаем компоненты на форме (рис. 19).

Свойства выбранных компонент:

Form1 – Caption – Значение функции

Label1 – Caption – Введите x

Label2 – Caption – Значение y

Button1 – Caption – Вычислить

Button2 – Caption – Очистить

Edit1 – Text – пусто

Edit2 – Text – пусто



Рис. 19.

Процедура и блок-схема (рис. 20) для вычисления значения функции будут иметь вид:

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

```
  var x,y:real;
```

```
  begin
```

```
    x:=strtoint(edit1.text);
```

```
    if x<=-2
```

```
    then
```

```
      y:=4
```

```
    else
```

```
      if (x>-2) and (x<1)
```

```
      then y:=sqr(x)
```

```
      else y:=1;
```

```
    edit2.text:=floattostr(y);
```

```
  end;
```

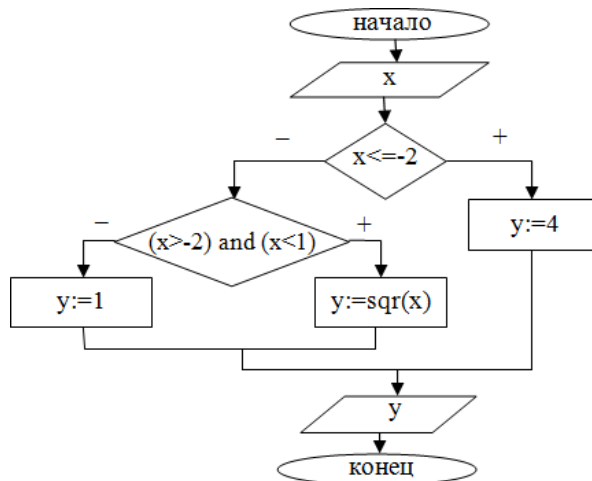


Рис. 20.

Задание 4. Процедуру для кнопки «Очистить» написать самостоятельно.

Компонент *CheckBox* ☒

При ветвлении кроме известных компонент используется компонента *CheckBox* (независимый переключатель). Позволяет пользователю выбрать/отменить определенную опцию.

Для задания состояния независимого переключателя используется свойство *State*.

CheckBox может находиться в трех состояниях:

1. Независимый переключатель выбран, в окошке компонента стоит галочка. Этому состоянию соответствует значение *State=cbChecked*.
2. Независимый переключатель не выбран, в окошке компонента галочка отсутствует. Этому состоянию соответствует значение *State=cbUnchecked*.
3. Независимый переключатель выбран, в окошке компонента стоит галочка, но само окошко и галочка серого цвета. Этому состоянию соответствует

значение *State=cbGrayed*. Это, так называемое, промежуточное состояние.

Свойство *AllowGrayed=[true, false]* включает (выключает) состояние *cbGrayed*.

Задание 5. Составить программу «Флажок» для вычисления квадрата и модуля числа.

Определим переменные задачи:

входные данные: *x: real*;

результат: *y1,y2: real* – модуль и квадрат числа соответственно.

Для решения задачи расположим компоненты на форме (рис. 21).

Свойства выбранных компонент:

Label1 – Caption – Введите *x*

CheckBox1 – Caption – Модуль

CheckBox2 – Caption – Квадрат

Edit1 – Text – пусто

Edit2 – Text – пусто

Edit3 – Text – пусто

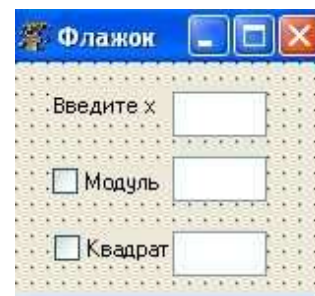


Рис. 21.

Процедуры для вычисления значений модуля и квадрата числа будут иметь вид:

```
procedure TForm1.CheckBox1Change(Sender: TObject);
```

```
var x,y1:real;
```

```
begin
```

```
x:=strtofloat(edit1.text);
```

```
y1:=abs(x);
```

```
if checkbox1.state=cbchecked
```

```
  then edit2.text:=floattostr(y1)
```

```
  else edit2.clear;
```

```
end;
```

```
procedure TForm1.CheckBox2Change(Sender: TObject);
```

```
var x,y2:real;
```

```
begin
```

```
x:=strtofloat(edit1.text);
```

```
y2:=sqr(x);
```

```
if checkbox2.state=cbchecked
```

```
  then edit3.text:=floattostr(y2)
```

else edit3.clear;

end;

Задание 6. Определить, принадлежит ли некоторая точка $M(x, y)$ закрашенной области (рис. 22).

Определим переменные задачи:

входные данные: x, y : real;

результат: выделяется один из двух флажков: принадлежит точка области или не принадлежит.

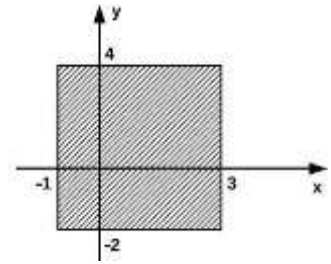


Рис. 22.

Для решения задачи расположим компоненты на форме (рис. 23).

Свойства выбранных компонент:

Form1 – Caption – Область

Label1 – Caption – Введите координаты точки M

Label2 – Caption – x:

Label3 – Caption – y:

Button1 – Caption – Вычислить

Button2 – Caption – Очистить

CheckBox1 – Caption – Точка M принадлежит области

CheckBox2 – Caption – Точка M не принадлежит области

Edit1 – Text – пусто

Edit2 – Text – пусто

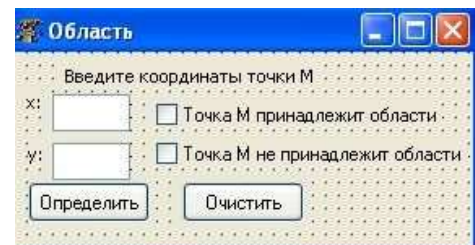


Рис. 23.

Процедуры для определения принадлежности точки области и очистки компонент будут иметь вид:

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

```
var x,y:real;
```

```
begin
```

```
x:=strtoint(edit1.text);
```

```
y:=strtoint(edit2.text);
```

```
if (x>=-1) and (x<=3) and (y>=-2) and (y<=4)
```

```
  then checkbox1.state:=cbchecked
```

```
    else checkbox2.state:=cbchecked;
```

```
end;
```

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

begin

```

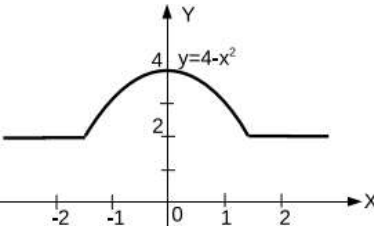
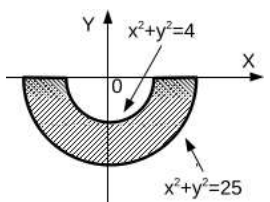
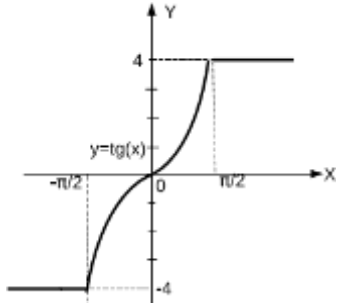
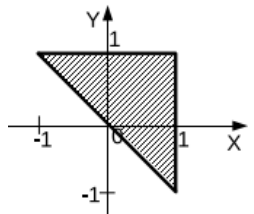
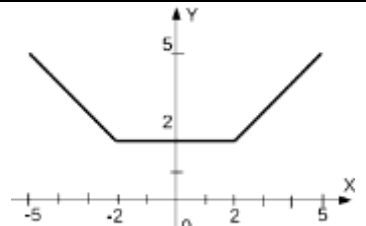
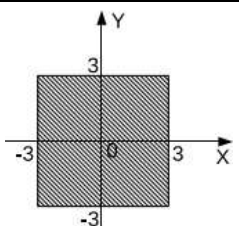
edit1.clear;
edit2.clear;
edit1.setfocus;
checkbox1.state:=cbunchecked;
checkbox2.state:=cbunchecked;
end;

```

Задания для самостоятельного выполнения

1. Дано вещественное число x . Для функции, представленной графиком, вычислить $y = f(x)$.
2. Найти максимальное (минимальное значение).
3. Выяснить, принадлежит ли точка $P(x, y)$ заштрихованной области.

Вариант	Задание		
	1	2	3
1		$\min(x^2 + y^2, z , xyz)$	
2		$\max(x^2 + y^2 - z^2, xyz, (x - y)^4)$	
3		$\min(x + y , \frac{z^2}{x}, \frac{45}{x^3})$	
4		$\min(\sin(x + y), 5 - 6y, \lg z)$	
5		$\max(\sin(x - y), \lg^2 z, \frac{x}{y})$	

6		$\min \left(\sin(x^2 + y^2), \operatorname{tg} z , \frac{x}{y} \right)$	
7		$\max \left(\sin(x + y), \operatorname{tg}(10 - 2z), 6y^3 \right)$	
8		$\max \left(\ln x \cdot y , \operatorname{tg} z , \frac{x}{y} \right)$	

Лабораторная работа №4

Оператор выбора

Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы. Параметром, по которому осуществляется выбор, служит *ключ выбора* s – выражение любого порядкового типа (кроме типов *Real* и *String*).

Синтаксис:

Case s of c_1 :оператор1; c_2 :оператор2; c_n :оператор n Else оператор; End;	выбор при условие 1: действия 1 при условие 2: действия 2 при условие n : действия n иначе действия $n+1$ все	} запись оператора выбора на алгоритмическом языке

где $c_1, c_2, \dots, c_n$ – все простые типы *Integer*, *Char*, за исключением *Real*. Это конкретное значение управляющей переменной и выражения s , при котором необходимо выполнить соответствующий оператор, игнорируя все остальные.

Если в наборе несколько значений, то они разделяются между собой «,». Можно указывать диапазоны значений между которыми ставятся «..». Между набором значений s и соответствующим ему оператором должно стоять «:». Значения $c_1 \dots c_n$ не должны повторяться.

Для работы с оператором выбора в *Lazarus* служат несколько компонент.

Компонент *RadioGroup*

Компонент представляет собой комбинацию *GroupBox* с набором *RadioButton*, т. е. контейнер для размещения зависимых переключателей, каждый переключатель помещается в специальный список *Items* и доступен по индексу.

Таблица 12

Основные свойства компонента *RadioGroup*

<i>Caption</i>	Заголовок
<i>Columns</i>	Количество столбцов переключателей
<i>ItemIndex</i>	Индекс выбранного переключателя (нумерация индексов начинается с нуля)
<i>Items</i>	Список строк с заголовками элементов

Если у компоненты *RadioGroup* свойство *ItemIndex* принимает значение (–1), то не один из переключателей не будет выделен.

Особенности компонента *RadioGroup*:

1. Щелчок мыши выделяет строку в виде радиокнопки с точкой.
2. В данный момент времени может быть включена максимально одна радиокнопка.

Понятие «зависимый переключатель» говорит о влиянии одной радиокнопки на другую. Если одна радиокнопка включена, то остальные выключены. При включение другой, предыдущая автоматически отключится.

Компонент *ListBox*

Компонент содержит список элементов, которые могут быть выбраны при помощи клавиатуры или мыши. Список может содержать строки и произвольное изображение. Список задается свойством *Items* и доступен по индексу.

Таблица 13

Основные свойства компонента *ListBox*

<i>Columns</i>	Количество столбцов элементов в списке
<i>ItemIndex</i>	Индекс выбранной строки (нумерация индексов начинается с нуля)
<i>Items</i>	Список строк с заголовками элементов

Если у компоненты *ListBox* свойство *ItemIndex* принимает значение (–1), то ни одна из строк не будет выделена.

Особенности компонента *ListBox*:

1. Щелчок мыши выделяет всю строку.
2. Выбор строки снимает выделение других строк.
3. Редактирование набора строк невозможно.

Компонент *Memo*

Компонент *Memo* – это многострочное окно ввода-вывода данных. Компонент представляет собой небольшой текстовый редактор, имеет ограничения на объем текста в 32Кб, что составляет 10-20 строк. Текст хранится в свойстве *Lines* класса *TStrings*. В целом компонент представляет собой пронумерованный набор строк.

Таблица 14

Основные свойства компонента *Memo*

<i>Lines</i>	Содержит строки текста (нумерация строк начинается с 0)
<i>ScrollBars</i>	Определяет наличие полос прокруток (<i>ssNone</i> , <i>ssHorizontal</i> , <i>ssVertical</i> , <i>ssBoth</i>)
<i>MaxLength</i>	Определяет максимальную длину текстовой строки. Если имеет значение 0, то длина строки не ограничена
<i>Text</i>	Содержит весь текст

Основные методы компонента *Мемо*

<i>Add(s)</i>	Добавляет строку в набор данных последней и возвращает ее индекс
<i>Clear</i>	Очищает набор данных.
<i>Delete(n)</i>	Удаляет строку с индексом n.
<i>Insert(n; s)</i>	Вставляет строку s в набор с индексом n.
<i>String[n]</i>	Получить текст в строке с номером n
<i>Count</i>	Получить количество строк в Мемо

Задание 1. Создать приложение (Калькулятор), обеспечивающее ввод двух целых чисел и выполнение над ними арифметических операций: сложения, вычитания, умножения и вещественного деления. Для выбора операции используется набор переключателей. Вывести сообщение об ошибке при вводе делителя, равного нулю.

Определим переменные задачи:

входные данные: a, b : integer;

результат: c : real.

Для решения задачи расположим компоненты на форме (рис. 24).

Свойства выбранных компонент:

Form1 – Caption – Калькулятор

RadioGroup1 – Caption – Операция

Items – сумма, разность, произведение, частное

Label1 – Caption – 1-ое число

Label2 – Caption – 2-ое число

Label3 – Caption – Результат

Edit1 – Text – пусто

Edit2 – Text – пусто

Edit3 – Text – пусто

Button1 – Caption – Вычислить

Button2 – Caption – Очистить

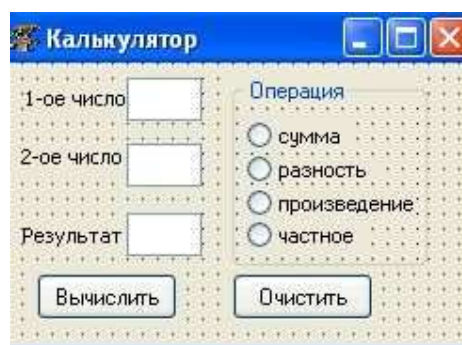


Рис. 24.

Процедура для кнопки «Вычислить» будет иметь вид:

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

```
var a,b:integer;
```

```
    c:real;
```

```
begin
```

```
    a:=strtoint(edit1.text);
```

```

b:=strtoint(edit2.text);
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;
edit3.text:=floattostr(c);
end;

```

Задание 2. Самостоятельно записать процедуру для кнопки «Очистить».

Задание 3. Имеется список периодических изданий. Для выбранного издания вывести день недели, в который он выходит. Вывод дня недели осуществить в многострочное окно ввода-вывода.

Определим переменные задачи:

промежуточная переменная: n (индекс выбранной строки): integer.

Для решения задачи расположим компоненты на форме (рис. 25).

Свойства выбранных компонент:

ListBox1 – Items – Комсомольская правда, АиФ, Тема, Моя мадонна, Телепорт,

Из рук в руки, Жизнь, Дважды два, Навигатор

Label1 – Caption – Издания

Label2 – Caption – Дни выхода издания

Memo1 – Lines – пусто

Button1 – Caption – Очистить

Form1 – Caption – Периодические издания

В программе будет осуществляться выбор одной из строк с названием периодического издания, поэтому процедура будет иметь вид:

```

procedure TForm1.ListBox1.Click (Sender: TObject);
var n:integer;
begin

```



Рис. 25.

```
n:= ListBox1.ItemIndex;
```

```
case n of
```

```
0: Memo1.lines.add('понедельник');
```

```
1: Memo1.lines.add('ежедневно, кроме воскресенья');
```

```
2: Memo1.lines.add('четверг');
```

```
3: Memo1.lines.add('четверг');
```

```
4: Memo1.lines.add('пятница');
```

```
5: Memo1.lines.add('суббота');
```

```
6: Memo1.lines.add('четверг');
```

```
7: Memo1.lines.add('пятница');
```

```
8: Memo1.lines.add('пятница');
```

```
end;
```

```
end;
```

Задание 4. Самостоятельно записать процедуру для кнопки «Очистить».

Задание 5. В городе имеется несколько кинотеатров: Зея, Октябрь, Благовещенск, Харбин, Амур, Восток, Кактус. Каждый из них работает в определенный день недели, который задается номером. В кинотеатрах идут фильмы: Мумия, Гарри Потер, Елки, Трансформеры, Люди в черном, Железный человек, Турист. Вводится номер дня недели и определяется кинотеатр, работающий в этот день. Затем из списка фильмов, идущих в кинотеатре, выбирается фильм и выводится время киносеансов.

Определим переменные задачи:

промежуточная переменная: *n* (индекс выбранной строки): integer.

Для решения задачи расположим компоненты на форме (рис. 26).

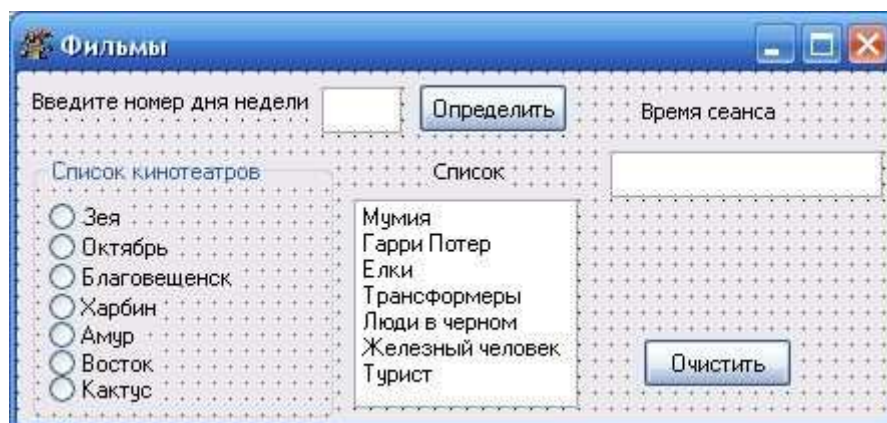


Рис. 26.

Свойства выбранных компонент:

Form1 – Caption – Фильмы

Edit1 – Text – пусто

Edit2 – Text – пусто

Label1 – Caption – Введите номер
дня недели

Label2 – Caption – Список

Label3 – Caption – Время сеанса

Button1 – Caption – Определить

Button2 – Caption – Очистить

RadioGroup1 – Items – Зоя, Октябрь,
Благовещенск, Харбин, Амур, Вос-
ток, Кактус

Процедуры для определения кинотеатра и для определения времени сеанса будут иметь вид:

```
procedure TForm1.Button1Click();
```

```
var n:integer;
```

```
begin
```

```
n:=strtoint(edit1.text);
```

```
case n of
```

```
1: RadioGroup1.ItemIndex:=1;
```

```
2: RadioGroup1.ItemIndex:=0;
```

```
3: RadioGroup1.ItemIndex:=4;
```

```
4: RadioGroup1.ItemIndex:=6;
```

```
5: RadioGroup1.ItemIndex:=2;
```

```
6: RadioGroup1.ItemIndex:=5;
```

```
7: RadioGroup1.ItemIndex:=3
```

```
else showmessage('Введите номер дня недели от 1 до 7');
```

```
end;
```

```
end;
```

```
procedure TForm1.ListBox1Click();
```

```
begin
```

```
case ListBox1.ItemIndex of
```

```
0: edit2.text:='12-00, 16-00, 20-00';
```

```
1: edit2.text:='12-30, 16-30, 20-30';
```

```
2: edit2.text:='11-00, 15-00, 21-00';
```

```
3: edit2.text:='11-30, 15-30, 21-30';
```

```
4: edit2.text:='12-00, 15-00, 22-00';
```

```
5: edit2.text:='11-00, 17-00, 21-00';
```

6: edit2.text:='12-00, 15-00, 21-30';

end;

end;

Задание 6. Самостоятельно записать процедуру для кнопки «Очистить».

Задания для самостоятельного выполнения

Вариант 1	1. Записать программу, которая по выбранному названию фигуры (треугольник, квадрат, ромб, прямоугольник и т.п.) выводит формулу, по которой вычисляется площадь этой фигуры. 2. Имеется расписание приема врача: понедельник 8-00 – 10-00, вторник 11-00 – 13-00, среда 9-00 – 11-00, четверг 8-00 – 10-00, пятница 11-00 – 13-00, суббота 9-00 – 10-00, воскресенье выходной. Необходимо для указанного дня недели выдать информацию о часах приема врача.
Вариант 2	1. Записать программу, которая на ввод времени суток выделяет соответствующее пожелание доброго утра, доброго дня, доброго вечера и спокойной ночи. 2. Пусть элементами круга являются радиус (первый элемент), диаметр (второй элемент) и длина окружности (третий элемент). Составить программу, которая по выбранному номеру элемента запрашивала бы его соответствующее значение и вычисляла бы площадь круга.
Вариант 3	1. Человек в течение своей жизни проходит несколько возрастных этапов. Требуется по введенному количеству лет определить возрастной промежуток для данного возраста. 2. Напишите программу стоимости телефонного разговора. Известно, что тариф днем, вечером и на выходных разный. Пользователь выбирает в списке время разговора и вводит количество минут, а на экране появляется стоимость всего разговора.
Вариант 4	1. Составьте программу стоимости услуг в парикмахерской. При выборе в списке вида услуг выводится ее стоимость. В программе учесть мужской и женский зал. 2. Написать программу, которая по введенному номеру месяца выделяет в списке название следующего за ним месяца.

Вариант 5	1. Составить программу подсчета стипендии в университете. Пользователь вводит количество студентов и в списке с радиокнопками выбирает категорию сдачи сессии: только с пятерками (M руб); только одна четверка (L руб); без троек (K руб); с тройками. В результате получает количество денег, расходуемых на стипендию данной группы студентов. 2. Определить остаток от деления целой части значения выражения $\ln x^2+ab$ на 7 и в зависимости от величины выделить в списке день недели с соответствующим номером.
Вариант 6	1. Для каждой введенной цифры (0 – 9) выделить в списке соответствующее ей название на английском языке. 2. Оклад служащего составляет X рублей. Для выбранного района (центральный, дальневосточный, северный) вычисляется з/плата. Выводится процентная надбавка (0%, 50%, 100%) и зарплата за месяц.
Вариант 7	1. Напишите программу телепередач. Пользователь вводит время просмотра, путем указания часа. В результате в списке выделяется соответствующие программы. 2. Средний ежеквартальный расход электроэнергии – P кВт/час. При этом: в 1 квартале расход на 15% выше среднего, во 2 и 4 кварталах – на 5% выше среднего, в 3 квартале – на 8% ниже среднего. Вводится номер квартала. Выделяется корректирующий сезонный процент и вычисляется расход электроэнергии в данном квартале.
Вариант 8	1. По введенному количеству лет вывести возможные виды кредита: до 1 года – экспресс-кредитование; до 5 лет – кредит на неотложные нужды; до 15 лет – ипотечное кредитование. Для предлагаемого кредита в Мето выводятся условия его получения. 2. Для суммы S в списке радиокнопок выбирается вид вклада («Пенсионный» – 17% годовых, «До востребования» – 3% годовых, «Молодежный» – 14% годовых и др.). Указать сумму вклада через один, два, три года с учетом начисляемых процентов. Суммы вкладов выводить в компонент Мето.

Лабораторная работа 5

Циклические структуры

Циклом называется многократно повторяемый участок вычислений.

1. Оператор цикла с параметром.

Данный оператор используется, если надо выполнить некоторую последовательность действий несколько раз, причем число повторений заранее известно.

Синтаксис:

For параметр цикла: =мл. знач. **to** ст. знач. **do**

For параметр цикла: =ст.. знач. **downto** мл. знач. **do**

Если параметр возрастает, то между границами его значений ставится *to*, если убывает, то *downto*.

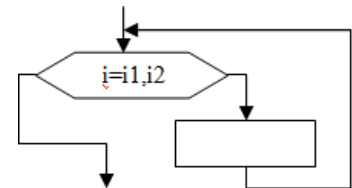


Рис. 27.

нц для *i* от *i1* до *i2*

тело цикла
(последовательность действий)

кц

} запись оператора выбора на алгоритмическом языке

Действия: параметру цикла присваивается начальное значение (*i1*) и выполняется тело цикла, затем параметру цикла присваивается следующее значение и вновь выполняется тело цикла и так до тех пор, пока не будут перебраны все значения параметра цикла (*i2*).

2. Оператор цикла с постусловием.

Данный оператор используется в программе, если надо провести некоторые повторяющиеся вычисления, однако число повторов заранее не известно и определяется самим ходом вычисления.

Синтаксис:

Repeat

тело цикла

Until условие;

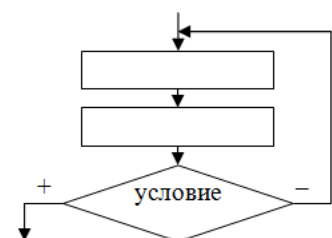


Рис. 28.

Действия: сначала выполняются операторы, потом проверяется условие. Если оно ложно, то снова выполняются операторы, и так до тех пор, при про-

верке условие не примет значение *True*.

3. Оператор цикла с предусловием.

Данный оператор, как и оператор *Repeat*, используется в программе, если надо провести некоторые повторяющиеся вычисления, однако число повторов заранее не известно и определяется самим ходом вычисления.

Синтаксис:

While *условие* **Do**
 тело цикла;

нц пока условие
 тело цикла
 (последовательность действий)
кц

запись оператора выбора на
алгоритмическом языке

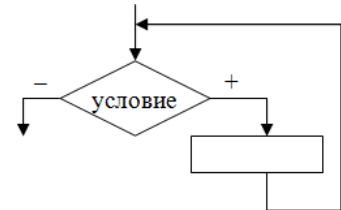


Рис. 29.

Действия: сначала проверяется условие, если оно истинно, то выполняется тело цикла. Затем снова проверяется условие и т. д. Если условие ложно, то цикл завершается и выполняется оператор, стоящий непосредственно после цикла.

Задание 1. Найти значение $n!$.

Определим переменные задачи:

входные данные: n : integer

промежуточные переменные: i : integer – параметр элемента, изменяется на промежутке $[1;n]$ с шагом 1.

результат: p : real

Задачу решим с использованием всех трех циклических структур. Расположим компоненты на форме (рис. 30).

Свойства выбранных компонент:

Form1 – Caption – Факториал

Label1 – Caption – Введите n

Label2 – Caption – Значение факториала

Button1 – Caption – С параметром

Button2 – Caption – С предусловием

Button3 – Caption – С постусловием

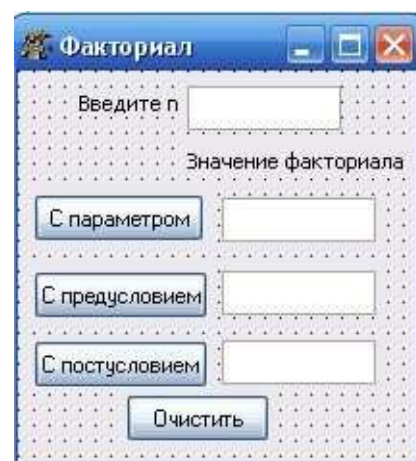


Рис. 30.

Button4 – Caption – Очистить

Edit1 – Text – пусто

Edit2 – Text – пусто

Edit3 – Text – пусто

Edit4 – Text – пусто

Запомните! При решении задач на циклические структуры первоначальное значение суммы $S=0$, а произведения $P=1$.

Процедуры и блок-схемы (рис. 31, 32, 33) для вычисления значения факториала будут иметь вид:

Цикл с параметром:

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

```
var i,n:integer;
```

```
    p:real;
```

```
begin
```

```
n:=strtoint(edit1.text);
```

```
P:=1;
```

```
for i:=1 to n do
```

```
    p:=p*i;
```

```
edit2.text:=floattostr(P);
```

```
end;
```

Цикл с предусловием:

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

```
var n,i:integer; P:real;
```

```
begin
```

```
n:=strtoint(edit1.text);
```

```
P:=1;
```

```
i:=1;
```

```
while i<=n do
```

```
begin
```

```
    P:=P*i;
```

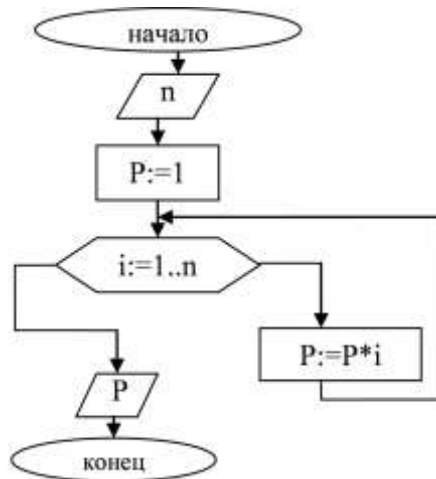


Рис. 31.

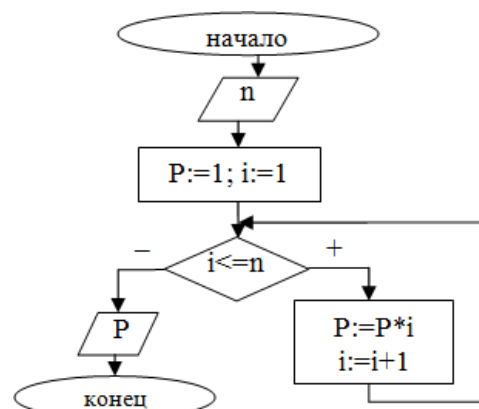


Рис. 32.

i:=i+1;

end;

edit3.text:=floattostr(P);

end;

Цикл с постусловием:

procedure TForm1.Button3Click(Sender: TObject);

var n,i:integer; P:real;

begin

n:=strtoint(edit1.text);

P:=1;

i:=1;

repeat

P:=P*i;

i:=i+1;

until i>n;

edit4.text:=floattostr(P);

end;

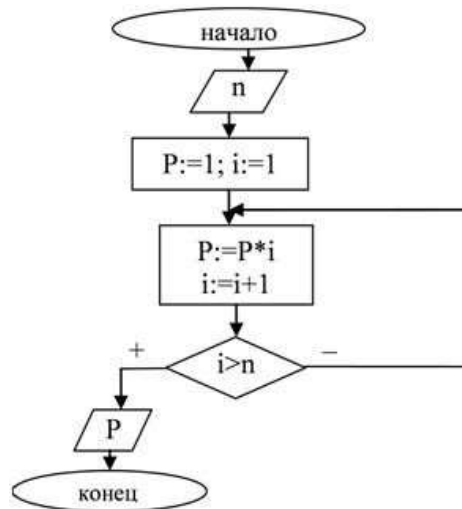


Рис. 33.

Задание 2. Процедуру для кнопки «Очистить» написать самостоятельно.

Задание 3. Составить программу, находящую сумму ряда $\sum_{i=1}^n \sin(i)$.

Условие задачи определяет ряд, вида $\sin(1) + \sin(2) + \sin(3) + \dots + \sin(n)$. Обозначим отдельный элемент переменной a . Можно сказать, что каждый элемент $a = \sin(i)$, где $i=1, 2, 3, \dots, n$.

Определим переменные задачи:

входные данные: n: integer

промежуточные переменные: i: integer; a: real

результат: S: real

Решение данной задачи рассмотрим для цикла с постусловием. Расположим следующие компоненты на форме (рис. 34).

Свойства выбранных компонент:

Form1 – Caption – Сумма ряда

Label1 – Caption – Введите n
 Button1 – Caption – Вычислить
 Button2 – Caption – Очистить
 Edit1 – Text – пусто
 Memo1 – Lines – пусто

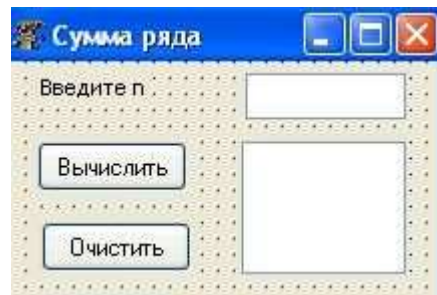


Рис. 34.

Процедура для кнопки «Вычислить»:

```

procedure TForm1.Button1.Click(Sender: TObject);
var n,i:integer; a,S:real;
begin
  n:=StrToInt (edit1.text);
  S:=0 ;
  i:=1;
  repeat
    a:=sin(i);
    S:=S+a;
    i:=i+1;
  until i>n;
  Memo1.Lines.Add (FloatToStr(s));
end;
  
```

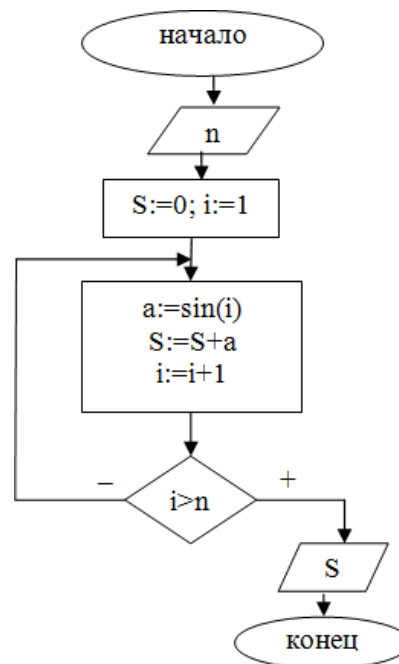


Рис. 35.

Задание 4. Процедуру для кнопки «Очистить» написать самостоятельно.

Задание 5. Составить программу для нахождения произведения ряда

$$\prod_{i=1}^n (3 + x^i).$$

Условие задачи определяет ряд, вида $(3 + x)(3 + x^2)(3 + x^3) \dots (3 + x^n)$.

Как и в задании 3 обозначим отдельный элемент переменной a , где $a = 3 + x^i$, где $i=1, 2, 3, \dots, n$.

Обозначим $t = x^i$. Тогда $a = 3 + t$. Первоначально $t=1$, а далее в цикле каждый раз увеличивается на степень $t = t \cdot x$.

Определим переменные задачи:

входные данные: n : integer; x : real

промежуточные переменные: i : integer; a, t : real

результат: P : real

Решение данной задачи рассмотрим на примере цикла с предусловием.

Расположим компоненты на форме (рис 36).

Свойства выбранных компонент:

Form1 – Caption – Произведение ряда

Label1– Caption – Введите n

Label2 – Caption – Введите x

Button1 – Caption – Вычислить

Button2 – Caption – Очистить

Edit1 – Text – пусто

Edit2 – Text – пусто

Memo1 – Lines – пусто

Процедура для кнопки «Вычислить»:

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

```
var n, i, x:integer; a, t, P:real;
```

```
begin
```

```
n:=StrToInt (edit1.text);
```

```
x:=StrToInt (edit2.text);
```

```
P:=1;
```

```
i:=1;
```

```
t:=1;
```

```
while i<=n do
```

```
  begin
```

```
    t:=t*x;
```

```
    a:=3+t;
```

```
    P:=P*a;
```

```
    i:=i+1;
```

```
  end;
```

```
Memo1.Lines.Add (FloatToStr(P));
```

```
end;
```

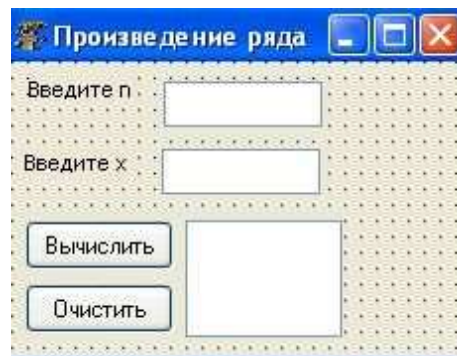


Рис. 36.

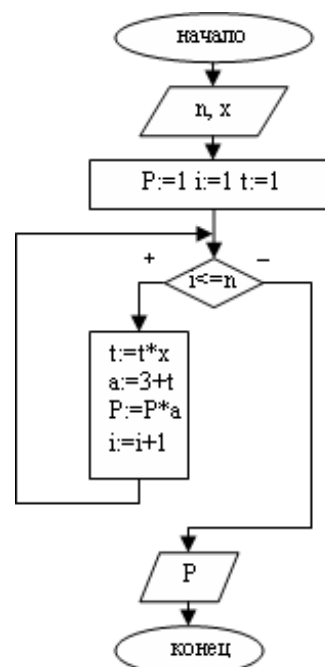


Рис. 37.

Задание 6. Процедуру для кнопки «Очистить» написать самостоятельно.

Компонент *Мето* часто используется, когда необходимо вывести значения функции на некотором интервале с заданным шагом. Задачи такого типа

называются задачами на табулирование, т. е. при решении таких задач результаты выводятся на каждом шаге.

Задание 7. Найти значение функции $y = e^x - e^{-x} - 2$ на отрезке $[a, b]$ с шагом h .

Определим переменные задачи:

входные данные: a, b, h : real

промежуточные переменные: x : real

результат: y : real

Решение данной задачи рассмотрим на примере цикла с предусловием. Расположим компоненты на форме (рис. 38).

Свойства выбранных компонент:

Form1 – Caption – Значение функции

Label1 – Caption – Табулирование функции

Label2 – Caption – a

Label3 – Caption – b

Label4 – Caption – h

Button1 – Caption – Выполнить

Button2 – Caption – Очистить

Memo1 – Lines – пусто

ScrollBars – ssVertical

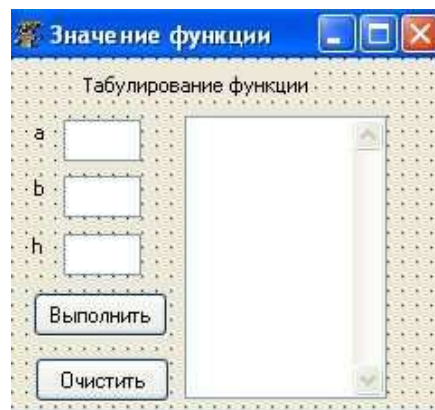


Рис. 38.

Процедура и блок-схема (рис. 39) для кнопки «Вычислить»:

procedure TForm1.Button1Click(Sender: TObject);

var a, b, h, y, x : real;

begin

$a := \text{StrToFloat}(\text{Edit1.Text});$

$b := \text{StrToFloat}(\text{Edit2.Text});$

$h := \text{StrToFloat}(\text{Edit3.Text});$

$x := a;$

while $x \leq b$ **do**

begin

$y := \exp(x) - \exp(-x) - 2;$

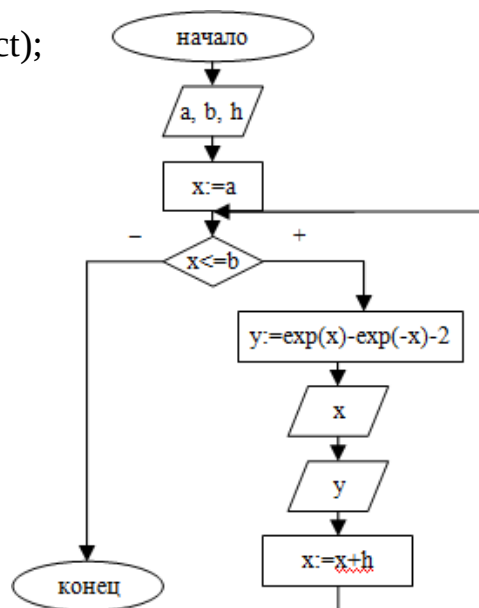


Рис. 39.

```
Memo1.Lines.Add('x='+FloatToStr(x)+' y='+FloatToStr(y));
```

```
x:=x+h;
```

```
end;
```

```
end;
```

Задание 8. Процедуру для кнопки «Очистить» написать самостоятельно.

Задания для самостоятельного выполнения

Вариант №1	1. Вычислить $Y = \sum_{k=1}^n \overline{k^2}$. 2. Вычислить $z = \sum_{i=1}^n \frac{x^i}{i}$. 3. Вычислить и вывести значения функции $y = \frac{x^2 - 3x + 2}{x^2 + 2x - 1}$ на интервале $[0; 2]$ с шагом $h=0,15$.	Вариант №5	1. Вычислить $Y = \sum_{k=1}^n \overline{(2k)^2}$. 2. Вычислить $z = \sum_{i=1}^n \frac{x^i}{(i+1)^2}$. 3. Вычислить и вывести значения функции $y = x^x (1 + \ln(x))$ на интервале $[0,1; 1,5]$ с шагом $h=0,15$.
Вариант №2	1. Вычислить $Y = \sum_{k=1}^n \frac{1}{k!}$. 2. Вычислить $z = \sum_{i=1}^n \frac{x^i}{i+2}$. 3. Вычислить и вывести значения функции $y = xe^x \sin(x)$ на интервале $[-1;1]$ с шагом $h=0,1$.	Вариант №6	1. Вычислить $Y = \sum_{k=1}^n \frac{1}{k^2}$. 2. Вычислить $z = \sum_{i=1}^n \frac{(x+1)^{2i+1}}{2i-1}$. 3. Вычислить и вывести значения функции $y = e^x \cos^2(x)$ на интервале $[-1; 1]$ с шагом $h=0,1$.
Вариант №3	1. Вычислить $Y = \sum_{k=1}^n \overline{k\sqrt{k}}$. 2. Вычислить $z = \sum_{i=1}^n \frac{ix^i}{2i-1}$. 3. Вычислить и вывести значения функции $y = x e^{-x^2}$ на интервале $[-2; -0,5]$ с шагом $h=0,15$.	Вариант №7	1. Вычислить $Y = \sum_{k=1}^n \frac{1}{k\sqrt{k}}$. 2. Вычислить $z = \sum_{i=1}^n \frac{1}{(2i-1)x^{2i-1}}$. 3. Вычислить и вывести значения функции $y = \frac{x}{x^4 + 3x^2 + 2}$ на интервале $[0; 3]$ с шагом $h=0,1$.

Вариант №4	1. Вычислить $T = \frac{1}{n} \sum_{k=1}^n \overline{k(k+1)}$. 2. Вычислить $z = \sum_{k=1}^n \frac{x^{i+1}}{i}$. 3. Вычислить и вывести значения функции $y = \frac{1}{x^2} \cdot \sin \frac{1}{x}$ на интервале $[1;3]$ с шагом $h=0,1$.	Вариант №8 1. Вычислить $T = \frac{1}{n} \sum_{k=1}^n \overline{k!}$. 2. Вычислить $z = \sum_{i=1}^n \frac{x^i}{i^2}$. 3. Вычислить и вывести значения функции $y = \frac{\ln^2 x}{x}$ на интервале $[6;8]$ с шагом $h=0,25$.
--------------------------	--	---

Лабораторная работа 6

Одномерный массив

Массив – это фиксированное количество элементов одного и того же типа, объединенные одним именем, где каждый элемент имеет свой номер.

Одномерный массив представляет собой вектор-строку или вектор столбец.

Описание массива

Массив описывается в разделе описания переменных.

Var имя массива: *array*[1..*t*₁] *of* *t*₂;

array – массив; *of* – из;

*t*₁ – количество элементов массива;

*t*₂ – тип элементов массива.

Пример. *Var A:array[1..30] of integer;*

Если в программе несколько массивов, например, *A*, *B*, *C* одного и того же типа, то они перечисляются через запятую.

Пример. *Var A,B,C:array[1..30] of integer;*

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. 40).

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.

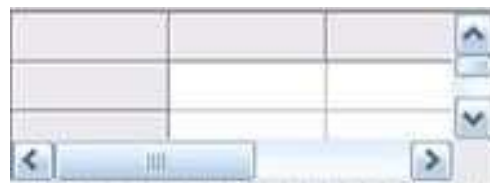


Рис. 40.

Основные свойства компонента *StringGrid*

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

<i>Caption</i>	Надпись на кнопке
<i>Kind</i>	Предлагает на выбор десять предопределенных типов кнопок (<i>bkOk</i> , <i>bkYes</i> , <i>bkCancel</i> и т.д.)
<i>Glyph</i>	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 (рис. 41).

Свойства выбранных компонент:

Form1 – *Caption* – Положительные элементы

Button1 – *Caption* – Сформировать массив

Edit1, 2 – *Text* – пусто

BitBtn – *Kind* – *bkOk*

Label1 – *Caption* – Введите количество элементов массива

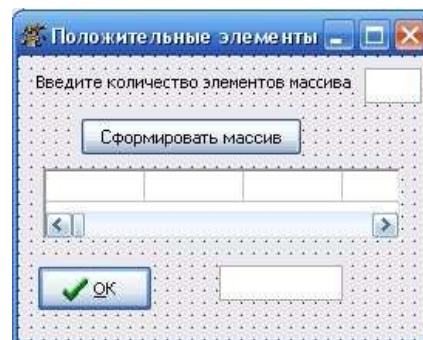


Рис. 41.

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 и блок-схема (рис. 42) для кнопки «ОК»:

```
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;
```

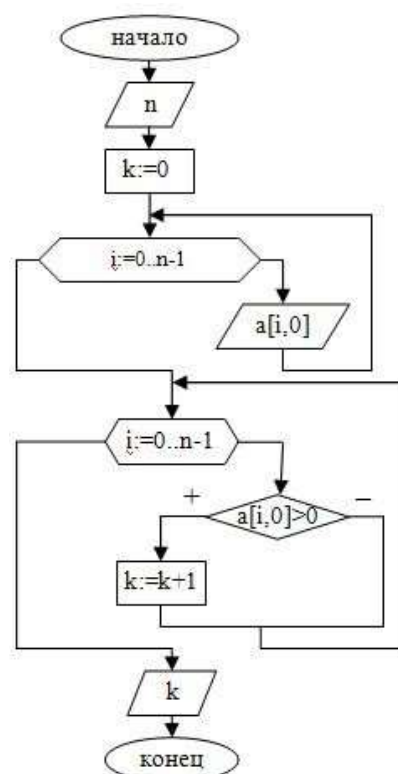


Рис. 42.

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 (рис. 43).

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

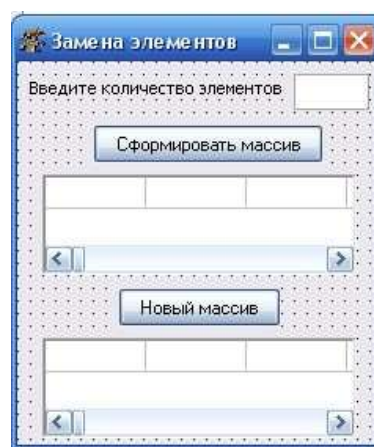


Рис. 43.

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 и блок-схема (рис. 44) 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;
```

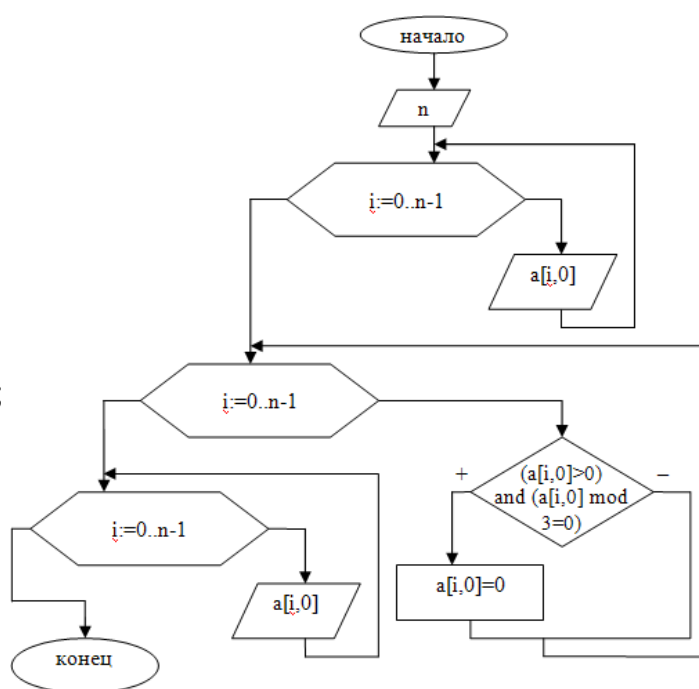


Рис. 44.

for i:=0 to n-1 do

StringGrid2.Cells[i,0]:=IntToStr(a[i,0]);

end;

Задания для самостоятельного выполнения

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 М сформировать массив Р 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 С сформировать массив А 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 А сформировать массив С той же размерности по правилу: первые k элементов - $A_i = 10C_i$, остальные - $A_i = 2C_i$.

Лабораторная работа 7

Многомерный массив

Многомерный массив представляет собой матрицу (таблицу).

Описание массива

Массив описывается в разделе описания переменных.

Var имя массива: *array*[1..*t*₁, 1..*t*₂] *of* *t*₃;

array – массив; *of* – из;

*t*₁ – количество столбцов массива;

*t*₂ – количество строк массива;

*t*₃ – тип элементов массива, допустимый в языке *Object Free Pascal*.

Пример. *Var* A:array[1..30, 1..30] *of* integer;

При работе с массивами обычно используется цикл с параметром.

Для работы с многомерными массивами используются те же самые компоненты, что и при работе с одномерными массивами.

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. 45).

Свойства выбранных компонент:

Form1 – Caption – Матрица

Label1 – Caption – Введите количество столбцов

Label2 – Caption – Введите количество строк

Label3 – Caption – Произведение

Label4 – Caption – Количество

Рис. 45.

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 и блок-схема (рис. 46) 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;

```

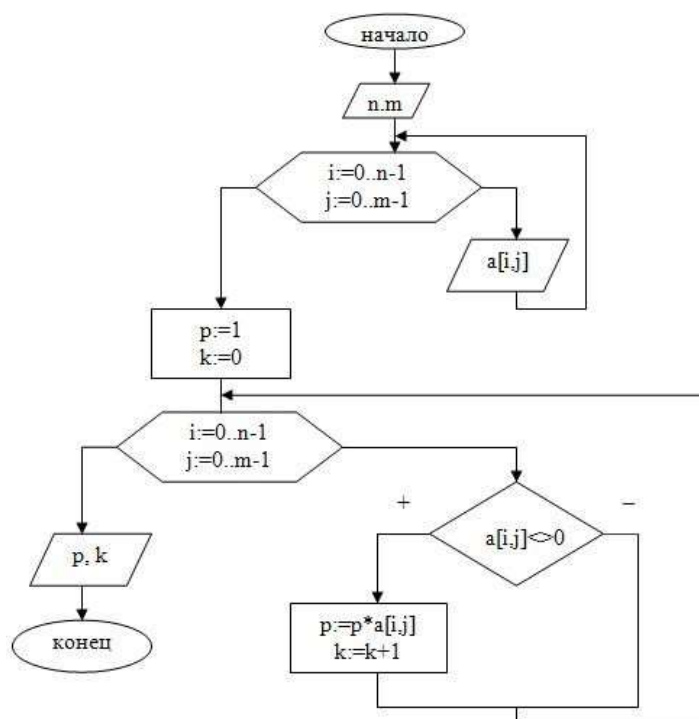


Рис. 46.

Задание 2. В матрице, размерность которой $n \times m$, 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. 47).

Свойства выбранных компонент:

Form1 – Caption – Замена элементов

Label1 – Caption – Введите количество элементов

StringGrid2 – ColCount – 100

RowCount – 100

Visible – False

FixedCols – 0

FixedRows – 0

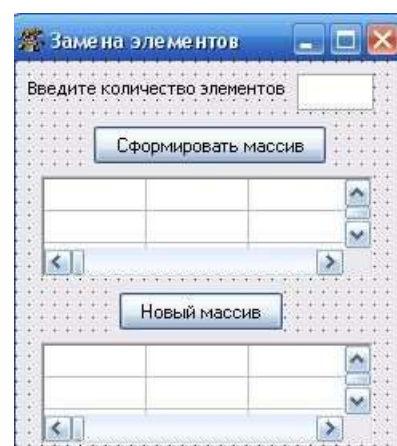


Рис. 47.

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 и блок-схема (рис. 48) для кнопки «Новый массив»:

procedure TForm1.Button2Click(Sender: TObject);

var i,j,n:integer;

a:array[0..100,0..100] of integer;

begin

```

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;

```

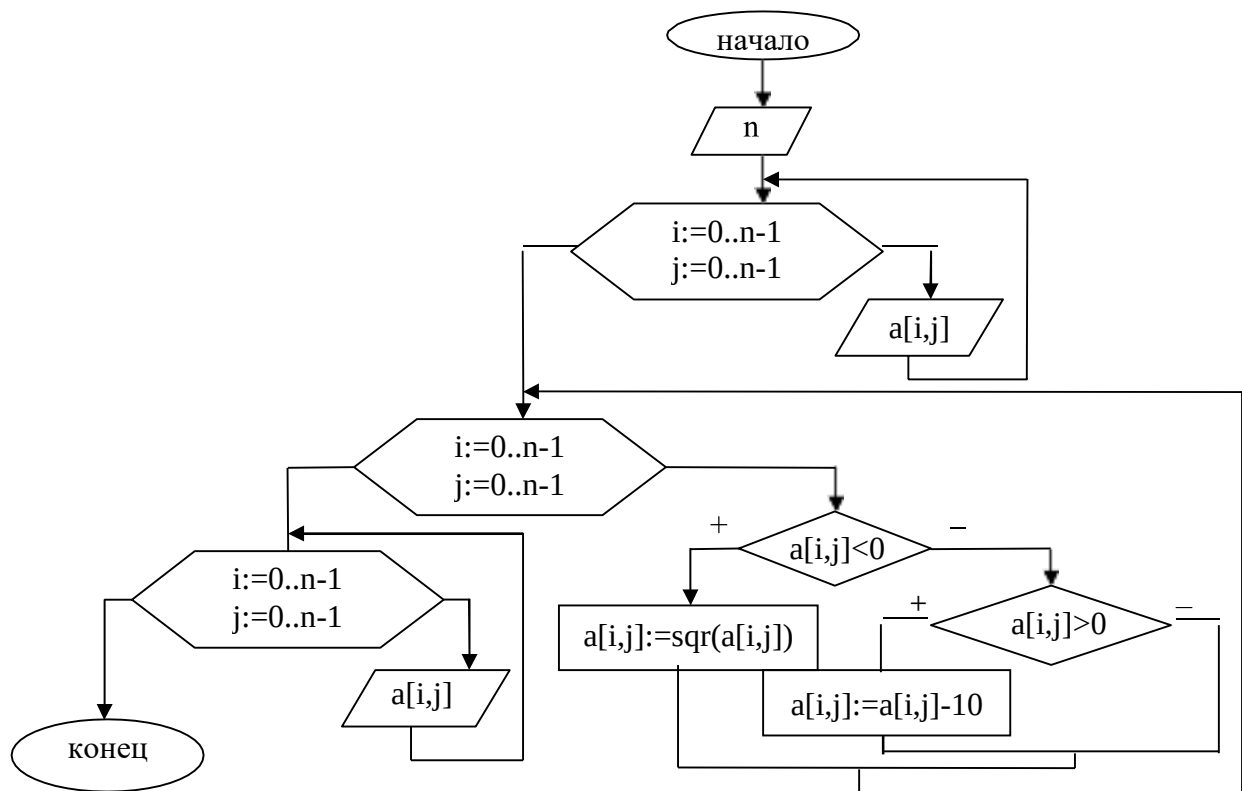


Рис. 48.

Задания для самостоятельного выполнения

Вариант №1	1. Дана матрица $B(n, n)$. Вычислить сумму и число элементов матрицы, находящихся под главной диагональю и на ней. 2. Дана матрица $D(n, n)$. Заменить нулевые элементы матрицы на 1, а все остальные на 10.
Вариант №2	1. Дана матрица $C(n, n)$. Вычислить сумму и число положительных элементов матрицы, находящихся над главной диагональю и на ней. 2. Дана матрица $B(n, n)$. Заменить элементы матрицы, кратные одновременно 3 и 5, на 100, остальные элементы заменить на 50.
Вариант №3	1. Дана матрица $C(n, n)$. Вычислить сумму и произведение элементов матрицы, квадраты которых меньше 225. 2. Дана матрица $D(n, n)$. Записать на место отрицательных элементов матрицы нули, а на место положительных элементов – единицы.
Вариант №4	1. Дана матрица $B(n, n)$ и число k. Найти количество элементов матрицы, равных заданному числу k. 2. Дана матрица $B(n, n)$. Заменить элементы, находящиеся выше главной диагонали и на ней по следующему правилу: отрицательные – на 5, положительные – на 10.
Вариант №5	1. Дана матрица $C(n, n)$. Вычислить количество и произведение элементов, кратных 2. 2. Дана матрица $B(n, n)$. Заменить элементы матрицы, большие 10, на их модули, остальные элементы заменить на их противоположные.
Вариант №6	1. Дана матрица $B(n, n)$. Найти количество и сумму элементов матрицы, кратных 2. 2. Дана матрица $C(n, n)$. Заменить четные элементы матрицы на их квадраты, остальные элементы заменить на их абсолютные значения.
Вариант №7	1. Дана матрица $B(n, n)$. Найти произведение и сумму нечетных элементов матрицы. 2. Дана матрица $D(n, n)$. Заменить отрицательные элементы главной диагонали на 2, а положительные – на 4.
Вариант №8	1. Дана матрица $B(n, n)$. Вычислить сумму и произведение элементов матрицы, находящихся на главной диагонали. 2. Дана матрица $A(n, n)$. Заменить на (-5) все элементы матрицы, кратные 5.

Лабораторная работа 8

Тема: Работа с файлами. Записи.

Основные понятия

Объявление файла

Файл – это совокупность данных, размещенных на внешнем носителе. Данные в файле называются компонентами. Количество данных в файле, в отличие от массива, в описании файла не указывают. Компонент файла не имеет индекса. Тип компонентов может быть как простым, так и сложным, но не может быть файлом. Файловый тип можно задать тремя способами. В разделе описания типов:

```
Type <имя типа>=file of <базовый тип>;  
    < имя типа >=TextFile;  
    < имя типа >=File;
```

В разделе объявления переменных:

```
Var < имя типа >: file of <базовый тип>;  
    < имя типа >: TextFile;  
    < имя типа >: File;
```

Здесь <имя типа> - имя файлового типа (правильный идентификатор); **file of** – зарезервированные слова (файл из); **TextFile** – имя стандартного типа текстовых файлов; <базовый тип> - любой тип Lazarus, кроме файлов.

Например:

```
Type comp=record;  
marka: string;  
hdd, ram: word;  
speed: integer;  
end;  
Text80= file of string[80];  
Var  
F1: file of char;  
F2: TextFile;  
F3: file;  
F4: Text80;  
F5: file of comp;
```

В зависимости от способа объявления можно выделить три вида файлов:

- типизированные файлы (задаются предложением **file of...**);

- текстовые файлы (определяются типом TextFile);
- нетипизированные файлы (определяются типом file);

В приведенном примере F1, F4 и F5 – типизированные файлы, F2 – текстовый файл, F3 – нетипизированный файл.

Доступ к файлам

Объявление файловой переменной задает только тип компонентов файла. Для того чтобы программа могла выводить данные в файл или считывать данные из файла, необходимо указать конкретный файл, т. е. связать файловую переменную с конкретным файлом (задать имя файла).

Файловая переменная связывается с именем файла в результате обращения к стандартной процедуре AssignFile:

AssignFile (<ф.п.>, <имя файла>);

Здесь <ф.п.> - файловая переменная (правильный идентификатор, объявленный в программе как переменная файлового типа); <имя файла > - текстовое выражение, содержащее имя файла и, если это необходимо, маршрут доступа к нему. Например:

AssignFile(f, 'a:\result.txt');

AssignFile(f, 'students\ivanov\korni.txt');

fname:=('otchet.txt'); AssignFile(f, fname);

Инициализировать файл означает указать для этого файла направление передачи данных. В Lazarus можно открыть файл для *чтения*, для *записи* информации, а также для чтения и записи одновременно.

Для *чтения* файл инициализируется с помощью стандартной процедуры Reset:

Reset (<ф.п.>);

Здесь <ф. п.> - файловая переменная, связанная ранее процедурой AssignFile с уже существующим файлом.

При выполнении этой процедуры дисковый файл подготавливается к чтению информации. В результате специальная переменная-указатель, связанная с этим файлом, будет указывать на начало файла, т. е. на компонент с порядковым номером 0.

В Lazarus разрешается обращаться к типизированным файлам, открытым процедурой Reset (т. е. для чтения информации), с помощью процедуры Write (т. е. для записи информации). Такая возможность позволяет легко обновлять ранее созданные типизированные файлы и при необходимости расширять их. Для текстовых файлов, открытых процедурой Reset, нельзя использовать процедуру Write или Writeln.

Стандартная процедура

Rewrite (<ф.п.>);

инициирует *запись* информации в файл, связанный с файловой переменной <ф.п.>. Процедурой *Rewrite* нельзя инициировать запись информации в ранее существовавший дисковый файл: при выполнении этой процедуры старый файл (если он был) уничтожается и никаких сообщений об этом в программу не передается. Новый файл подготавливается к приему информации, и его указатель принимает значение 0.

Стандартная процедура

Append (<ф.п.>);

инициирует *запись* в ранее существовавший текстовый файл для его расширения, при этом указатель файла устанавливается в его конец. Процедура *Append* применима только к текстовым файлам, т. е. их файловая переменная должна иметь тип *TextFile*. Процедурой *Append* нельзя инициировать запись в типизированный или нетипизированный файл. Если текстовый файл ранее уже был открыт с помощью *Reset* или *Rewrite*, использование процедуры *Append* приведет к закрытию этого файла и открытию его вновь, но уже для добавления записей.

Стандартная процедура

CloseFile (<ф.п.>);

закрывает файл, однако связь файловой переменной с именем файла, установленная ранее процедурой *AssignFile*, сохраняется. При создании нового или расширении старого файла процедура обеспечивает сохранение в файле всех новых записей и регистрацию файла в каталоге. Функции процедуры *CloseFile* выполняются автоматически по отношению ко всем открытым файлам при нормальном завершении программы. Поскольку связь файла с файловой переменной сохраняется, файл можно повторно открыть без дополнительного использования процедуры *AssignFile*

Текстовые файлы

Текстовый файл представляет собой последовательность строк. В конце каждой строки расположен символ завершения строки *Eoln* (End Of LiNe - конец строки) - символ #13, за которым может идти символ перевода строки *Lf* (linefeed) – символ #10. Заканчивается текстовый файл символом окончания файла *Eof* (End Of File - конец файла) – символ #26.

Для ввода данных из любого текстового файла можно воспользоваться функциями *read* и *readln*. Только, первым аргументом в этих функциях должна идти файловая переменная.

Например:

read(f, data); - читает строку из файла *f* в переменную *data*.

readln(f, data); - читает строку из файла *f* в переменную *data* и после этого переходит к следующей строке.

Процедура *write* и *writeln* осуществляют вывод в текстовый файл.

Например:

write(f, data); - данные из *data* записываются в текстовый файл.

writeln(f, data); - данные из *data* записываются в текстовый файл и производит переход на следующую строку.

Функции для работы с текстовыми файлами:

function Eof (var F: FileText):Boolean; - функция возвращает *true*, если достигнут конец файла.

function SeekEof(var F: FileText):Boolean; - аналогична *Eof*, но пробелы, табуляция и маркеры конца строки пропускаются.

function Eoln(var F: TextFile): Boolean; - тестирует маркер конца строки и возвращает *true*, если конец строки достигнут.

function SeekEoln [(var F: FileText)]:Boolean; - аналогична *Eoln*, но пробелы, табуляция и маркеры конца строки пропускаются.

Типизированные файлы

Типизированные файлы предназначены для хранения информации одинакового типа. Длина любого компонента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому из них (т. е. доступ к компоненту по его порядковому номеру).

Ввод-вывод информации в них также осуществляется с помощью процедур *read*, *readln*, *write* и *writeln*. Перед первым обращением к процедурам ввода-вывода указатель файла стоит в его начале и указывает на первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные в списках ввода-вывода должны иметь тот же тип, что и компоненты файла. Если этих переменных в списке несколько, указатель будет смещаться после каждой операции обмена данными между переменными и дисковым файлом.

Для типизированных файлов предусмотрены дополнительные процедуры и функции:

procedure Seek(var F; N: Longint); - перемещает указатель на элемент с номером *N*.

function FilePos(var F): Longint; - возвращает номер текущего элемента.

function FileSize(var F): Integer; - возвращает количество элементов в файле.

Нетипизированные файлы

Нетипизированный файл – это последовательность байтов, без определённой структуры, записанная в файл.

В отличие от других типов файлов в процедурах открытия и записи добавляется ещё один параметр – размер записи. Если этот параметр не будет указан, он принимается равным 128 байтам.

```
procedure Reset(var F: File [, RecSize: Word]);
```

```
procedure Rewrite(var F: File [, RecSize: Word]);
```

Максимальный размер записи – до 2 Гб. Наиболее оптимальный размер записи с точки зрения скорости – это физический размер сектора на накопительном устройстве. Чаще всего, он составляет 512 байт. Следует отметить, что скорость работы с нетипизированными файлами значительно выше, чем с другими типами файлов.

Вместо процедур `read` и `write` для нетипизированных файлов используются `BlockRead` и `BlockWrite`:

```
procedure BlockRead(var F: File; var Buf; Count: Integer [, var AmtTransferred: integer]);
```

```
procedure BlockWrite(var F: File; var Buf; Count: Integer [, var AmtTransferred: integer]);
```

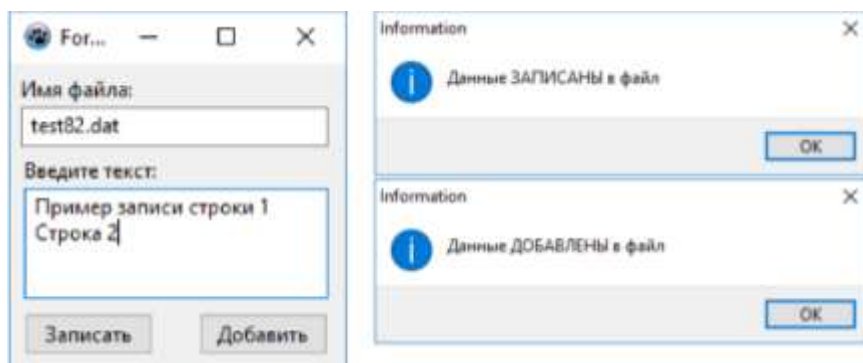
где `F` – файловая переменная;

`Buf` – переменная для сохранения/чтения данных в/из файла. Размер данной переменной должен быть: `Count*RecSize` байтов;

`Count` – указывает какое количество записей необходимо записать/прочитать за одно обращение к файлу;

`AmtTransferred` – содержит реальное количество прочитанных/записанных файлов.

Задание 1: Написать программу, которая выполняет запись или добавление строки из поля Мемо в текстовый файл. Окно программы показано ниже.



В приложении объявлены переменные:

f: TextFile; // файл

fName: String[80]; // имя файла

Сценарий работы приложения: после ввода текста в поле Мемо щелкаем на кнопке «Записать», данные будут записаны в текстовый файл (файл открывается процедурой **Rewrite**). Затем вводим новый текст и щелкаем на кнопке «Добавить» - файл открывается процедурой Append, и данные будут дописаны в файл.

Исходный текст обработчика события –щелчок на кнопке Записать – приведен в листинге 1. Процедура открывает файл в режиме создания нового или замещения существующего файла и записывает текст, находящийся в поле компонента Memo1. Имя файла нужно ввести во время работы в поле Edit1. Можно задать предопределенное имя файла во время разработки формы приложения. Для этого надо присвоить значение, например **test.txt**, свойству Edit1.Text.

Листинг 1. Создание нового или замещение существующего файла

```
procedure TForm1.Button1Click(Sender: TObject);
var
f: TextFile; // файл
fName: String[80]; // имя файла
i: integer;
begin
fName := Edit1.Text;
AssignFile(f, fName);
Rewrite(f); // открыть для перезаписи
// запись в файл (строки нумеруются с нуля)
for i:=0 to Memo1.Lines.Count do
writeln(f, Memo1.Lines[i]);
CloseFile(f); // закрыть файл
MessageDlg('Данные ЗАПИСАНЫ в файл', mtInformation,
[mbOk],0);
end;
```

В листинге 2 приведена процедура, которая запускается нажатием командной кнопки **Добавить**. Она открывает файл, имя которого указано в поле Edit1, и добавляет в него содержимое поля Memo1.

Листинг 2. Добавление в существующий файл

```
procedure TForm1.Button2Click(Sender: TObject);
var
f: TextFile; // файл
```

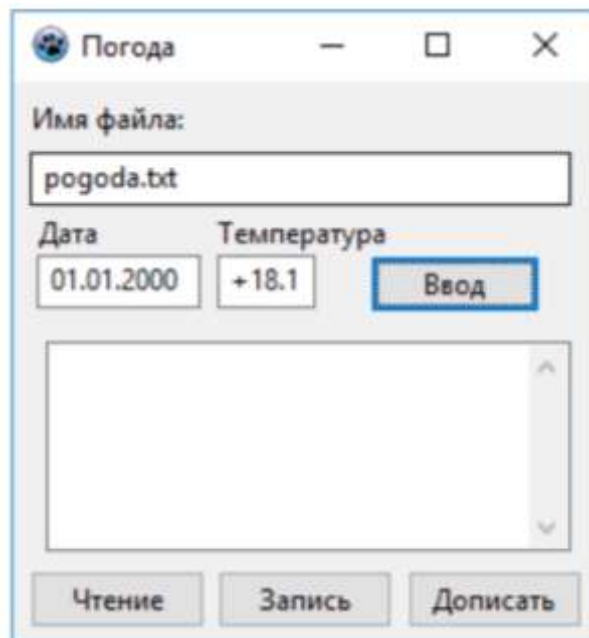
```

fName: String[80]; // имя файла
i: integer;
begin
fName := Edit1.Text;
AssignFile(f, fName);
Append(f); // открыть для добавления
// запись в файл строки нумеруются с нуля
for i:=0 to Memo1.Lines.Count do
writeln(f, Memo1.Lines[i]);
CloseFile(f); // закрыть файл
MessageDlg('Данные ДОБАВЛЕНЫ в файл ', mtInformation,
[mbOk],0);
end;

```

Задание 2: Следующая программа ведет простую базу данных. В поля «Дата» и «Температура» пользователь вводит дату и среднесуточную температуру. После щелчка на кнопке «Ввод» данные записываются в поле Мемо.

При щелчке на кнопке «Чтение» в поле Мемо отображается содержимое файла. При щелчке на кнопке «Запись» все содержимое Мемо записывается в файл, а при щелчке на кнопке «Дописать» – добавляется в файл.



Листинг 3 Простая база данных «Погода»

implementation

{ \$R *.lfm }

```

var
date: string[10]; temp: string[5];
f: TextFile; // файл
fName: String[80]; // имя файла
{ TForm1 }
procedure TForm1.Button2Click(Sender: TObject);
var i: integer;
begin
fName := Edit1.Text; AssignFile(f, fName);
Rewrite(f); // открыть для перезаписи
// запись в файл (строки нумеруются с нуля)
for i:=0 to Memo1.Lines.Count do
writeln(f, Memo1.Lines[i]);
CloseFile(f); // закрыть файл
MessageDlg('Данные ЗАПИСАНЫ в файл', mtInformation,
[mbOk],0);
end;
procedure TForm1.Button1Click(Sender: TObject);
var buf: string;
begin
fName := Edit1.Text; AssignFile(f, fName);
{$I-} // отмена проверки результатов ввода-вывода
Reset(f);
{$I+} // включение проверки результатов ввода-вывода
if IOResult<>0 then begin
MessageDlg('Ошибка открытия файла#13'+Edit1.Text,
mtInformation, [mbOk],0);
exit;
end;
while not EOF(f) do begin
readln(f, buf); // прочитать строку из файла
if Length(buf)>0 then
Memo1.Lines.Add(buf); // добавить строку в поле Memo1
end;
CloseFile(f);
end;
procedure TForm1.Button3Click(Sender: TObject);

```

```

begin
date:=Edit2.Text; temp:=Edit3.Text;
Memo1.Lines.Add(date+' '+temp);
end;
procedure TForm1.Button4Click(Sender: TObject);
var i: integer;
begin
fName := Edit1.Text; AssignFile(f, fName);
{$I-}
Append(f);
{$I+}
if IOResult<>0 then begin
MessageDlg('Ошибка открытия файла#13'+Edit1.Text,
mtInformation, [mbOk],0);
exit;
end;
for i:=0 to Memo1.Lines.Count-1 do
writeln(f, Memo1.Lines[i]);
CloseFile(f);
end;
end.

```

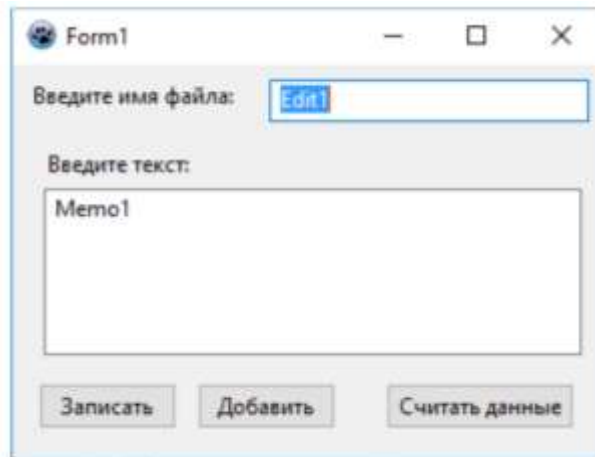
После нескольких запусков программы файл `pogoda.txt` может быть, например, таким:

```

22.12.2013 +5
03.03.2014 +12
04.03.2014 +15.4

```

Задание 3: Нужно написать программу, которая выполняет считывание строки из файла, имя которого ввел пользователь во время работы программы, и выводит эти строки в поле Мемо. Окно программы имеет вид:



Листинг 4. Чтение и изменение файла

```

procedure TForm1.Button1Click(Sender: TObject);
var
  f: TextFile; fName: String[80]; // имя файла
  i: integer;
begin
  fName := Edit1.Text; AssignFile(f, fName);
  Rewrite(f); // открыть для перезаписи
  // запись в файл (строки нумеруются с нуля)
  for i:=0 to Memo1.Lines.Count-1 do
    writeln(f, Memo1.Lines[i]);
  CloseFile(f); // закрыть файл
  MessageDlg('Данные ЗАПИСАНЫ в файл', mtInformation,
    [mbOk],0);
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  f: TextFile; fName: String[80]; // имя файла
  i: integer;
begin
  fName := Edit1.Text; AssignFile(f, fName);
  Append(f); // открыть для добавления
  // запись в файл строки нумеруются с нуля
  for i:=0 to Memo1.Lines.Count-1 do
    writeln(f, Memo1.Lines[i]);
  CloseFile(f); // закрыть файл
  MessageDlg('Данные ДОБАВЛЕНЫ в файл ', mtInformation,

```

```

[mbOk],0);
end;
procedure TForm1.Button3Click(Sender: TObject);
var
f: TextFile; fName: String[80];// имя файла
s:string;
begin
fName := Edit1.Text; AssignFile(f, fName);
Reset(f); // открыть для чтения
Memo1.Clear;
Memo1.Lines.Add('файл '+fName+' открыт');
// запись в файл строки нумеруются с нуля
while not Eof(f) do begin
readln(f, s); Memo1.Lines.Add(s);
end;
CloseFile(f);
MessageDlg('Данные СЧИТАНЫ ', mtInformation,
[mbOk],0);
end;

```

Для организации обработки файла использована инструкция цикла **while**, которая обеспечивает проверку значения функции **EOF** перед каждым чтением, в том числе и перед первым.

Наличие кнопки Записать и соответствующей ей процедуры позволяет сохранить содержимое поля Мемо в файле, т. е. программа чтение из файла представляет собой примитивный редактор текста.

Добавление очередной прочитанной из файла строки в поле Мемо выполняется применением метода **Add** к свойству Lines.

Задание 4¹. Создать Windows-приложение для записи в файл и чтения из файла ведомости об успеваемости учащихся в количестве 10 человек. Каждая запись файла должна содержать фамилию с инициалами, а также оценки по физике, математике и сочинению. Вывести список учащихся, отсортированный в алфавитном порядке и

¹ Задание повышенной сложности.

записать эту информацию в текстовый файл.

Один из возможных вариантов панели интерфейса создаваемого приложения показан на рис.

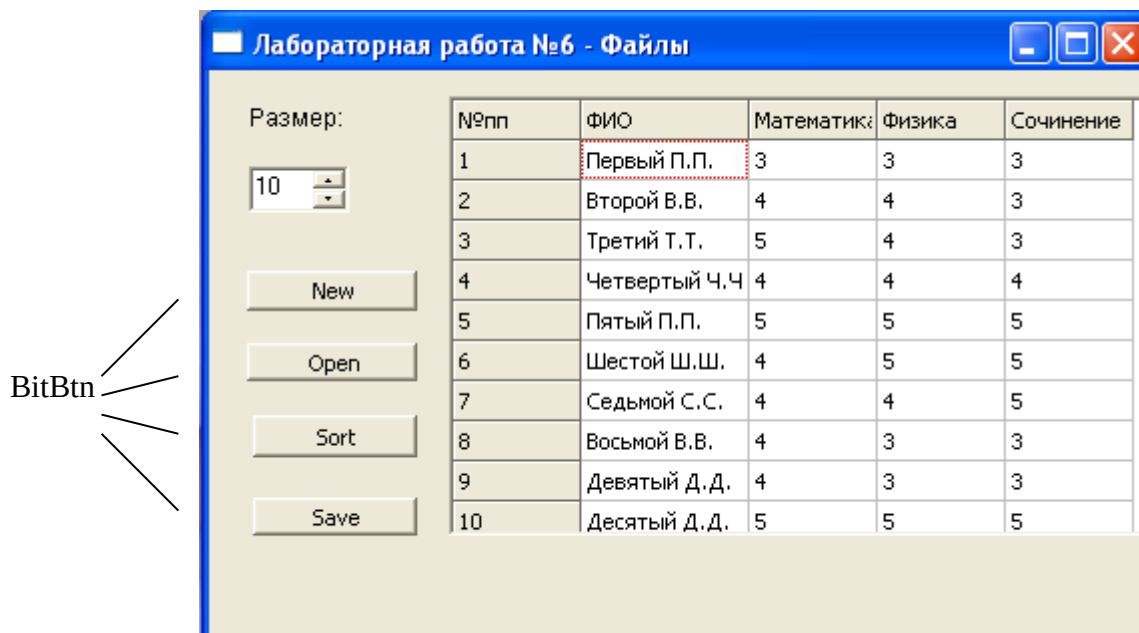


Рис. Окно учебной программы

При работе с файлами чтение и запись информации удобно организовывать с помощью компонентов OpenFileDialog и SaveDialog.

Компоненты OpenFileDialog и SaveDialog находятся на странице Dialogs. Все компоненты этой страницы являются невизуальными, т.е. не видны в момент выполнения приложения. Поэтому их можно разместить в любом удобном месте Формы. Оба рассматриваемых компонента имеют идентичные свойства и отличаются только внешним видом.

Для установки компонентов OpenFileDialog и SaveDialog на Форму необходимо на странице Dialogs *Палитры Компонентов* щелкнуть “мышью” соответственно по

пиктограмме  или  и разместить ее в любом свободном месте Формы. При выполнении приложения в момент вызова компонента появляется диалоговое окно, с помощью которого пользователь выбирает имя файла и маршрут к нему. В случае успешного завершения диалога имя выбранного файла и маршрут поиска содержится в свойстве FileName.

Пользователь имеет возможность настроить параметры окна диалога по своему усмотрению. В частности, изменить заголовок окна можно с помощью свойства Title. В свойстве DefaultExt можно указать расширение файла, если оно не задано пользователем. Свойство Filter используется для поиска (фильтрации) файлов,

отображаемых в окне. Установка фильтра производится следующим образом. Выделив соответствующий компонент, необходимо дважды щелкнуть по правой (белой) части свойства Filter *Инспектора Объектов*. В появившемся окне редактора фильтра - Filter Editor необходимо в колонке Filter Name набрать текст, характеризующий соответствующий фильтр, а в колонке Filter –маску. Для компонента OpenFileDialog1 установите значения масок как показано на рис. 2.

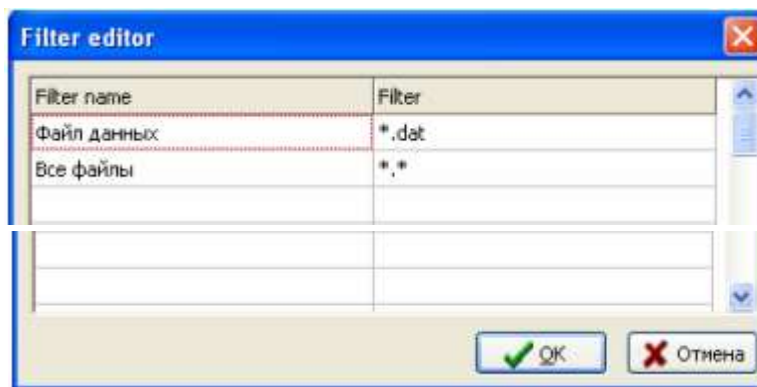


Рис. 2. Окно редактора фильтра

Маска *.dat означает что, будут видны файлы данных с любым именем и с расширением dat, а маска *.* - что будут видны все файлы (с любым именем и с любым расширением).

Для того, чтобы файл автоматически записывался с расширением dat в свойстве DefaultExt запишите требуемое расширение - dat.

Аналогичным образом настройте компонент SaveDialog1 для текстового файла (расширение txt).

Для удобства работы с несколькими различными процедурами обработки событий в свойстве Name каждого компонента BitBtn замените программные имена кнопок: BitBtn1 – на BitBtnNew, BitBtn2 – на BitBtnOpen, BitBtn3 – на BitBtnSort, BitBtn4 – на BitBtnSave. Двойным нажатием клавиши “мыши” на кнопках BitBtn создайте соответствующие процедуры обработки событий. Пользуясь текстом модуля unfile, внимательно наберите операторы этих процедур.

Листинг 5. Текст модуля unfile

```
unit unfile;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, Grids,
  StdCtrls, Spin, Buttons;
```

```

type
  { TForm1 }
TForm1 = class(TForm)
  BitBtnNew: TBitBtn;
  BitBtnOpen: TBitBtn;
  BitBtnSort: TBitBtn;
  BitBtnSave: TBitBtn;
  Label1: TLabel;
  OpenFileDialog1: TOpenDialog;
  SaveDialog1: TSaveDialog;
  SpinEdit1: TSpinEdit;
  StringGrid1: TStringGrid;
  procedure BitBtnNewClick(Sender: TObject);
  procedure BitBtnOpenClick(Sender: TObject);
  procedure BitBtnSaveClick(Sender: TObject);
  procedure BitBtnSortClick(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure SpinEdit1Change(Sender: TObject);
private
  { private declarations }
public
  { public declarations }
end;
var
  Form1: TForm1;
implementation
{ TForm1 }
type
  zap=record           // объявление записи
    fio: string[20];
    mat,fiz,soch: integer;
  end;
var
  MZap: array[1..25] of zap;    // объявление массива записей
  FileZap: file of zap; // объявление файла записей
  FileText: TextFile;    // объявление текстового файла
  FileNameZap,FileNameText: string; // имена файла записей и текстового файла
  n: integer;    // текущее кол-во элементов массива записей
{ Обработчик события создания Формы }
procedure TForm1.FormCreate(Sender: TObject);
begin
  with StringGrid1 do
    begin    // занесение информации в ячейки StringGrid1
      Cells[0,0]:='№пп';

```

```

Cells[1,0]:='ФИО';
Cells[2,0]:='Математика';
Cells[3,0]:='Физика';
Cells[4,0]:='Сочинение';
end;
BitBtnSort.Hide;           // спрятать кнопку "Sort"
BitBtnSave.Hide;          // спрятать кнопку "Save"
end;
{Обработчик кнопки изменения размера}
procedure TForm1.SpinEdit1Change(Sender: TObject);
var i,m: integer;
begin
m:=StrToInt(SpinEdit1.Text); // присвоить новое значение размера
with StringGrid1 do
begin
RowCount:=m+1; // пересчитать количество строк
if m>n then // если строки добавлены то
for i:=n+1 to m do // инициализировать новые ячейки
begin
Cells[0,i]:=IntToStr(i);
Cells[1,i]:="";
Cells[2,i]:="";
Cells[3,i]:="";
Cells[4,i]:="";
end;
end;
n:=m; // запомнить новое значение размера
end;
{ Обработчик нажатия кнопки Sort }
procedure TForm1.BitBtnSortClick(Sender: TObject);
var
i,j :integer;
vper:zap;
begin
for i:=1 to n do
with StringGrid1,MZap[i] do
begin
fio:=Cells[1,i];
mat:=StrToInt(Cells[2,i]);
fiz:=StrToInt(Cells[3,i]);
soch:=StrToInt(Cells[4,i]);
end;
{ сортировка методом отбора}
for i:=1 to n-1 do

```

```

for j:=i+1 to n do
  if MZap[i].fio>MZap[j].fio then
    begin
      vper:=MZap[i];
      MZap[i]:=MZap[j];
      MZap[j]:=vper;
    end;
for i:=1 to n do // заполнение ячеек StringGrid1 полями массива записей
  with StringGrid1,MZap[i] do
    begin
      Cells[0,i]:=IntToStr(i);
      Cells[1,i]:=fio;
      Cells[2,i]:=IntToStr(mat);
      Cells[3,i]:=IntToStr(fiz);
      Cells[4,i]:=IntToStr(soch);
    end;
end;
{ Создание нового файла записей}
procedure TForm1.BitBtnNewClick(Sender: TObject);
var
  i:integer;
begin // вывод на экран окна с предупреждающим сообщением
  if MessageDlg('Содержимое существующего файла будет уничтожено. Вы
уверены?',mtConfirmation, mbYesNoCancel, 0)=mrYes then
    begin
      for i:=1 to n do
        with StringGrid1,MZap[i] do
          begin // формирование полей массива записей
            fio:=Cells[1,i];
            mat:=StrToInt(Cells[2,i]);
            fiz:=StrToInt(Cells[3,i]);
            soch:=StrToInt(Cells[4,i]);
          end;
        with OpenFileDialog1 do
          begin
            Title:='Создание файла'; // заголовок окна диалога
          if Execute then //выполнение стандартного диалога выбора имени файла
            begin
              FileNameZap:=FileName; // присваивание имени файла
              // назначить файлу FileZap имя FileNameZap
              AssignFile(FileZap,FileNameZap);
              ReWrite(FileZap); // открыть файл на запись
            for i:=1 to n do
              write(FileZap,MZap[i]); // запись в файл массива записей

```

```

    CloseFile(FileZap);    // закрытие файла записей
end;
end;
end;
end;
{ обработчик нажатия кнопки Open }
procedure TForm1.BitBtnOpenClick(Sender: TObject);
var
    i:integer;
begin
    with OpenDialog1 do
        begin
            Title:='Открытие файла'; // заголовок окна диалога
            if Execute then // выполнение стандартного диалога выбора имени файла
                begin
                    FileNameZap:=FileName; // присваивание имени файла
                    // назначить файлу FileZap имя FileNameZap
                    AssignFile(FileZap,FileNameZap);
                    ReSet(FileZap);    // открыть файл на чтение
                    n:=0; // инициализация счетчика кол-ва прочитанных из файла элементов
                    while not EoF(FileZap) do
                        begin
                            n:=n+1; // увеличение счетчика
                            // чтение из файла n-го элемента массива записей
                            read(FileZap,MZap[n]);
                        end;
                        // занести кол-во прочитанных записей в SpinEdit
                        SpinEdit1.Text:=IntToStr(n);
                        StringGrid1.RowCount:=n+1; // присвоить кол-во строк
                        for i:=1 to n do
                            with StringGrid1,MZap[i] do
                                begin // заполнение ячеек StringGrid1 полями массива записей
                                    Cells[0,i]:=IntToStr(i);
                                    Cells[1,i]:=fio;
                                    Cells[2,i]:=IntToStr(mat);
                                    Cells[3,i]:=IntToStr(fiz);
                                    Cells[4,i]:=IntToStr(soch);
                                end;
                                CloseFile(FileZap); // закрытие файла записей
                            end;
                        end;
                        BitBtnSort.Show;    // показать кнопку "Sort"
                        BitBtnSave.Show;    // показать кнопку "Save"
                    end;

```

```

{ Сохранение данных в текстовом файле}
procedure TForm1.BitBtnSaveClick(Sender: TObject);
var
  i: integer;
begin
  with SaveDialog1 do
    if Execute then // выполнение стандартного диалога выбора имени файла
      begin
        FileNameText:=FileName; // присваивание имени файла
        // назначить файлу FileText имя FileNameText
        AssignFile(FileText,FileNameText);
        Rewrite(FileText); // открыть текстовый файл на запись
        for i:=1 to n do
          with MZap[i] do // запись в текстовый файл
            writeln(FileText,i:3,fio:20,mat:5,fiz:5,soch:5);
        CloseFile(FileText); // закрытие текстового файла по окончании записи
      end;
  BitBtnSort.Hide; // спрятать кнопку "Sort"
end;
initialization
  {$I unfile.lrs}
end.

```

Запустите созданное приложение. Занесите в соответствующие поля панели интерфейса информацию об успеваемости учащихся. Кнопкой “New” сохраните данные в файле. Завершите выполнение приложения.

Вновь запустите приложение и кнопкой “Open” откройте только что созданный файл. Убедитесь, что информация не содержит ошибок. При необходимости обнаруженные ошибки можно исправить, а также дополнить ведомость новой информацией. Для сортировки ведомости в алфавитном порядке воспользуйтесь кнопкой “Sort” и сохраните отсортированную информацию кнопкой “New”.

Еще раз завершите и вновь запустите приложение.

Кнопкой “Open” откройте файл и убедитесь, что в нем теперь содержится ведомость, отсортированная в алфавитном порядке. Кнопкой “Save” сохраните информацию в текстовом файле. Для просмотра содержимого текстового файла воспользуйтесь, например, приложением “Microsoft Word”.

Используя все управляющие компоненты панели интерфейса, убедитесь в правильном функционировании приложения во всех предусмотренных режимах

работы.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ ВЫПОЛНЕНИЯ

1. Ведомость абитуриентов, сдавших вступительные экзамены в университет находится в файле на диске и содержит: ФИО абитуриента, оценки. Определить средний балл по университету и вывести список абитуриентов, средний балл которых выше среднего балла по университету. Первыми в списке должны идти студенты, сдавшие все экзамены на 5.

2. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизор, радиоприемник и т. п.), марка изделия, дата приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о состоянии заказов на текущие сутки по группам изделий.

3. У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования на ближайшую неделю в следующем виде: дата выезда, пункт назначения, время отправления, число свободных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать m мест до города N на k -й день недели с временем отправления поезда не позднее t часов вечера. Вывести время отправления или сообщение о невозможности выполнить заказ в полном объеме.

4. Разработать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать: номер группы, ФИО студента, оценки за последнюю сессию. Вывести списки студентов по группам. В каждой группе ФИО студентов должны быть расположены в порядке убывания среднего балла.

5. В текстовом файле хранится телефонный справочник предприятия. Для каждого номера телефона указан номер помещения и ФИО служащих, сидящих в данном помещении. Составить программу, которая по номеру телефона выдает номер помещения и список сидящих в нем людей.

Номер телефона — двузначный. В одном помещении может находиться от одного до четырех служащих.

6. В текстовом файле хранится список служащих. Для каждого служащего указаны фамилия и инициалы, название занимаемой должности, год поступления на ра-боту

и оклад. Написать программу, выполняющую вывод на экран информации о служащем с максимальным окладом.

7. В текстовом файле хранится список товаров. Для каждого товара указаны его название, название магазина, в котором продается товар и его стоимость. Написать программу, выполняющую вывод на экран информации о товаре, название которого введено с клавиатуры;

8. Дан список студентов, содержащий следующие поля: фамилия, имя; номер телефона; день рождения (массив из трех чисел). Написать программу, выполняющую вывод на экран информации о студенте, номер телефона которого введен с клавиатуры; если такого нет, вывести на экран соответствующее сообщение.

Лабораторная работа 9

Графические методы и процедуры

В среде программирования Lazarus также есть возможность рисовать на разных объектах. В распоряжении программиста находится полотно (холст) — свойство Canvas, карандаш — свойство Pen, и кисть — свойство Brush.

Свойством Canvas обладают некоторые компоненты, например:

- форма (класс Tform);
- таблица (класс TstringGrid);
- растровая картинка (класс Timage);
- принтер (класс Tprinter);
- область для рисования (класс TPaintbox).

При рисовании компонента, обладающего свойством Canvas, сам компонент рассматривается как прямоугольная сетка, состоящая из отдельных точек, называемых пикселями.

Положение пикселя характеризуется его вертикальной (X) и горизонтальной (Y) координатами. Левый верхний пиксель имеет координаты (0,0). Вертикальная координата возрастает сверху вниз, горизонтальная — слева направо. Общее количество пикселей по вертикали определяется свойством Height, а по горизонтали — свойством

Width. Каждый пиксель может иметь свой цвет. Для доступа к любой точке полотна используется свойство Pixels[X,Y]:Tcolor. Это свойство определяет цвет пикселя с координатами X(integer), Y(integer). Изменить цвета любого пикселя полотна можно с помощью следующего оператора присваивания

Компонент.Canvas.Pixels[X,Y]:=Color; где Color - переменная или константа типа Tcolor. Определены следующие константы цветов таблица:

Таблица 1. Значение свойств Color

Константа	Цвет	Константа	Цвет
clBlack	Черный	clSilve	Серебристый
clMaroon	Каштановый	clRed	Красный
clGreen	Зеленый	clLime	Салатовый
clOlive	Оливковый	clBlue	Синий
clNavy	Темно-синий	clFuchsia	Ярко-розовый
clPurple	Розовый	clAqua	Бирюзовый
clTeal	Лазурный	clGray	Серый

Цвет любого пикселя можно получить с помощью следующего оператора присваивания:

Color:=Компонент.Canvas. Pixels[X,Y];

где Color- переменная типа Tcolor.

Класс цвета точки Tcolor определяется как длинное целое longint. Переменные этого типа занимают в памяти четыре байта. Четыре байта переменных этого типа содержат информацию о долях синего (B), зеленого (G) и красного (R) цветов и устроены следующим образом: \$00BBGGRR.

Для рисования используются методы класса Tcanvas, позволяющие изобразить фигуру (линию, прямоугольник и т.д.) или вывести текст в графическом режиме, и три класса, определяющие инструменты вывода фигур и текстов:

- TFont (шрифты);
- TPen (карандаш, перо);
- TBrush (кисть).

Класс TFont

- Можно выделить следующие свойства объекта Canvas.Tfont:
- Name (тип string) — имя используемого шрифта.
- Size (тип integer) — размер шрифта в пунктах (points).

Пункт – это единица измерения шрифта, равная 0,353 мм, или 1/72 дюйма.

- Style — стиль начертания символов, который может быть обычным, полужирным (fsBold), курсивным (fsItalic), подчеркнутым (fsUnderline) и перечеркнутым (fsStrikeOut). В программе можно комбинировать необходимые стили, например, чтобы установить стиль «полужирный курсив», необходимо написать следующий оператор:

Объект.Canvas. Font. Style:=[fsItalic,fsBold]

- Color (тип Tcolor) — цвет символов.
- Charset (тип 0..255) — набор символов шрифта. Каждый вид шрифта, определяемый его именем, поддерживает один или более наборов символов. В таблице приведены некоторые значения Charset.

Таблица 2. Значения свойства Charset

Константа	Значение	Описание
ANSI_CHARSET	0	Символы ANSI
DEFAULT_CHARSET	1	Задается по умолчанию. Шрифт выбирается только по его имени Name и размеру Size. Если описанный шрифт недоступен в системе, будет заменен другим
SYMBOL_CHARSET	2	Стандартный набор символов
MAC_CHARSET	77	Символы Macintosh
RUSSIAN_CHARSET	204	Символы кириллицы

Класс TPen

Карандаш (перо) используется как инструмент для рисования точек, линий, контуров геометрических фигур. Основные свойства объекта Canvas.Tpen:

- Color (тип Tcolor) – определяет цвет линии;
- Width (тип Integer) – задает толщину линии в пикселях;
- Style – дает возможность выбрать вид линии. Это свойство может принимать значение, указанное в таблице.

Таблица 3. Виды линий

Значение	Описание
psSolid	Сплошная линия
psDash	Штриховая линия
psDot	Пунктирная линия
psDashDot	Штрих-пунктирная линия
psDashDodDot	Линия, чередующая штрих и два пунктира
psClear	Нет линии

Mode – определяет, каким образом взаимодействуют цвета пера и полотна. Выбор значения этого свойства позволяет получать различные эффекты, возможные значения Mode приведены в таблице. По умолчанию вся линия вычерчивается цветом, определяемым значением Pen.Color, но можно определять инверсный цвет линии по отношению к цвету фона. В этом случае независимо от цвета фона, даже если цвет линии и фона одинаков, линия будет видна.

Таблица 4. Возможные значения свойства Mode

Режим	Операция	Цвет пикселя
pmBlack	Black	Всегда черный
pmWhite	White	Всегда белый
pmNop	-	Неизменный
pmNot	Not Screen	Инверсный цвет по отношению к цвету

		фона
pmCopy	Pen	Цвет, указанный в свойствах Color пера Pen (это значение принято по умолчанию)
pmNotCopy	Not Pen	Инверсия цвета пера

Класс TBRUSH

Кисть (Canvas.Brush) используется методами, обеспечивающими вычерчивание замкнутых фигур для заливки. Кисть обладает двумя основными свойствами:

- Color (тип Tcolor) – цвет закрашивания замкнутой области;
- Style – стиль заполнения области.

Класс TCANVAS

Это класс является основным инструментом для рисования графики. Рассмотрим наиболее часто используемые методы этого класса.

Procedure MoveTo(X, Y : Integer);

Метод MoveTo изменяет текущую позицию пера на позицию, заданную точкой (X, Y). Текущая позиция хранится в переменной PenPos типа Tpoint. Определение типа Tpoint следующее:

type Tpoint =record

X: Longint;

Y: Longint;

end;

Текущую позицию пера можно считывать с помощью свойства PenPos следующим образом:

X:=PenPos.X;

Y:=PenPos.Y;

Procedure LineTo(X, Y :Integer);

Метод LineTo соединяет прямой линией текущую позицию пера и точку с координатами (X, Y). При этом текущая позиция пера перемещается в точку с координатами (X, Y).

Задание 1. Нарисовать на форме прямые линии

Для решения задачи расположим на форме кнопку и пропишем процедуру обработки события этой кнопки TForm1.Button1Click:

Procedure TForm1.Button1Click(Sender: TObject)

begin

Form1.Canvas.LineTo(30,50);

end;

В результате щелчка по кнопке на форме возникнет прямая линия, соединяющая точку с координатами (0,0) и точку с координатами (30,50). Теперь перепишем процедуру обработки события следующим образом:

Procedure *Tform1.Button1Click(Sender: TObject)*

begin

Form1.Canvas.LineTo(Canvas.PenPos.x+30,

Canvas.PenPos.y+50);

end;

При первом щелчке по кнопке на экране прорисовывается аналогичная линия. Но при повторном щелчке первая процедура продолжает рисовать линию, соединяющую точки (0,0) и (30,50). Вторая процедура рисует линию, которая соединяет текущую точку с точкой, получившейся из текущей добавлением к координате X числа 30, а к координате Y – числа 50. Т.е. при повторном щелчке по кнопке процедура соединяет прямой линией точки (30,50) и (60,100). При третьем щелчке по кнопке будут соединяться прямой линией точки (60,100) и (90,150) и т.д.

Procedure *PolyLine(const Points array of Tpoint);*

Метод *PolyLine* рисует ломаную линию, координаты вершин которой определяются массивом *Points*.

Задание 2. Рассмотрим работу процедуры на примере. Расположим на форме кнопки Рисовать (*Button1*) и Выход (*Button2*) и запишем следующие операторы процедур обработки события:

procedure *Tform1.Button1Click(Sender: TObject);*

var *temp :array [1..25] of Tpoint;*

i : byte;

j: integer;

begin

j:=1;

for i:=1 to 25 do

begin

//вычисление координат вершин ломаной линии

*temp[i].x:=25+(i-1)*10;*

temp[i].y:=150-j(i-1)*5;*

j:=-j;

end;

Form1.Canvas.Polyline (temp);

end;

procedure *Tform1.Button2Click(Sender: TObject);*

begin

Form1.Close;

end;

После запуска программы и щелчка по кнопке «Рисовать» окно формы будет выглядеть, как на рисунке 1.

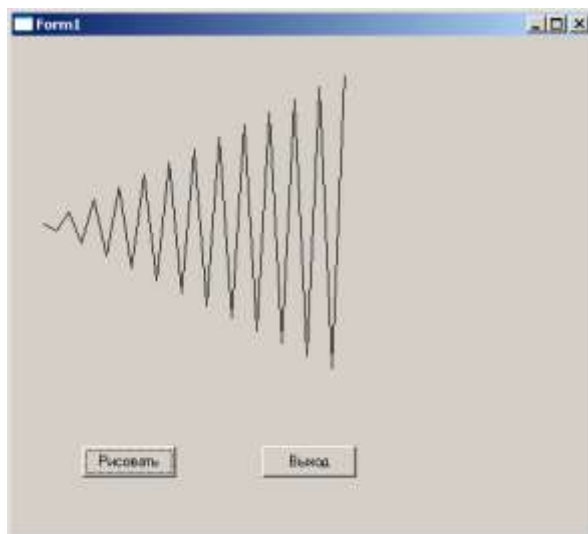


Рисунок 1. Пример использования процедуры PolyLine

Давайте рассмотрим основные методы класса TCanvas

Procedure Ellipse (X1, Y1, X2, Y2 :Integer);

Метод Ellipse вычерчивает на холсте эллипс или окружность. X1, Y1, X2, Y2 — это координаты прямоугольника, внутри которого вычерчивается эллипс. Если прямоугольник является квадратом, то вычерчивается окружность.

Procedure Arc (X1,Y1,X2,Y2,X3,Y3,X4,Y4:Integer);

вычерчивает дугу эллипса. X1, Y1, X2, Y2 — это координаты, определяющие эллипс, частью которого является дуга. X3, Y3 —

координаты, определяющие начальную точку дуги, X4, Y4- координаты, определяющие конечную точку дуги. Дуга рисуется против часовой стрелки.

Procedure Rectangle (X1, Y1, X2, Y2 :Integer);

рисует прямоугольник. X1, Y1, X2, Y2- координаты верхнего левого и нижнего правого углов прямоугольника.

Procedure RoundRect (X1,Y1,X2,Y2,X3,Y3:Integer);

вычерчивает прямоугольник со скругленными углами. X1, Y1, X2,Y2- координаты верхнего левого и нижнего правого углов прямоугольника, а X3, Y3 — размер эллипса, одна четверть которого используется для вычерчивания скругленного угла.

Procedure Polygon(const Points array of Tpoint);

рисует замкнутую фигуру (многоугольник) по множеству угловых точек, заданному массивом Points. При этом первая точка соединяется прямой линией с последней. Этим метод Polygon отличается от метода Poliline, который не замыкает конечные точки. Рисование осуществляется текущим пером Pen, а внутренняя область фигуры закрашивается текущей кистью Brush.

Procedure Pie (X1,Y1,X2,Y2,X3,Y3,X4,Y4 :Integer);

рисует замкнутую фигуру — сектор окружности или эллипса с помощью текущих параметров пера Pen, внутренняя область закрашивается текущей кистью Brush.

Точки (X1, Y1) и (X2, Y2) задают прямоугольник, описывающий эллипс. Начальная точка дуги определяется пересечением эллипса с прямой, проходящей через его центр и точку (X3, Y3). Конечная точка дуги определяется пересечением эллипса с прямой, проходящей через его центр и точку (X4, Y4). Дуга рисуется против часовой стрелки от начальной до конечной точки. Рисуются прямые, ограничивающие сегмент и проходящие через центр эллипса и точки (X3, Y3) и (X4, Y4).

procedure TextOut(X,Y:Integer;const Text:String);

пишет строку текста Text, начиная с позиции с координатами (X,Y). Текущая позиция PenPos пера Pen перемещается на конец выведенного текста. Надпись выводится в соответствии с текущими установками шрифта Font, фон надписи определяется установками текущей кисти. Для выравнивания позиции текста на канве можно использовать методы, позволяющие определить высоту и длину текста в пикселях — TextExtent, TextHeight и TextWidth.

Рассмотрим эти функции.

function TextExtent (const Text : String): Tsize;

возвращает структуру типа Tsize, содержащую длину и высоту в пикселях текста Text, который предполагается написать на канве текущим шрифтом.

Type Tsize = record

cx:Longint; cy:Longint;

end;

function TextHeight(const Text:String):Integer;

возвращает высоту в пикселях текста Text, который предполагается написать на канве текущим шрифтом.

function TextWidth(const Text:String):Integer;

возвращает длину в пикселях текста Text, который предполагается написать на канве текущим шрифтом. Это позволяет перед выводом текста на канву определить размер надписи и расположить ее и другие элементы изображения наилучшим образом. Если цвет кисти в момент вывода текста отличается от того, которым закрашена канва, то текст будет выведен в цветной прямоугольной рамке, но ее размеры будут точно равны размерам надписи.

Мы рассмотрели основные функции рисования. Прежде чем перейти непосредственно к рисованию, обратите внимание на то, что если вы свернете окно с графикой, а затем его восстановите, то картинка на форме исчезнет. Изменение размеров окна также может испортить графическое изображение в окне. Для решения этой проблемы существуют процедуры обработки событий

Объект.FormPaint

и Объект.FormResize.

Процедура Объект.FormPaint выполняется после появления формы на экране.

Процедура Объект.FormResize выполняется после изменения размера формы.

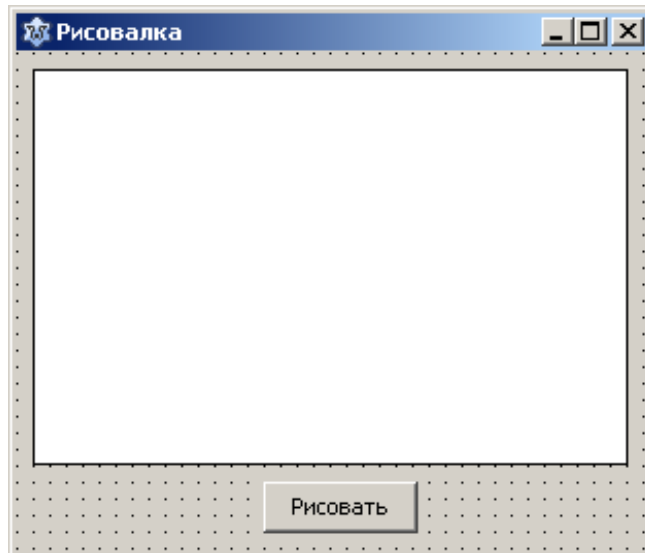
Следовательно, для того чтобы не пропадала картинка, все операторы рисования нужно помещать внутрь Объект.FormPaint и дублировать в процедуре Объект.FormResize.

Задание 3. Нарисовать на объекте Paintbox Дом.

Разместите на форме компонент Paintbox. Найти его можно на вкладке Additional палитры компонентов. Этот компонент размещается на форме в виде прозрачного пунктирного квадрата, и в его пределах можно рисовать.

Разместите также на форме кнопку и измените её свойства:

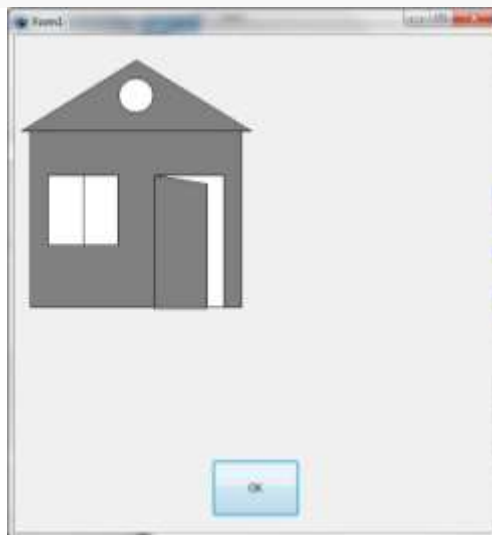
Caption	Рисовать
---------	----------



Теперь дважды щёлкните по кнопке Рисовать и напишите код программы, которая рисует дом.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  with Paintbox1.Canvas do begin  
    Brush.Color := clGray;  
    Rectangle(10,100,250,300); // корпус  
    Polygon([Point(0,100),Point(130,20),Point(260,100)]); // крыша  
    Brush.Color := clWhite;  
    Ellipse(110,40,150,80); // чердак  
    Rectangle(30,150,110,230); // окно  
    MoveTo(70,150);  
    LineTo(70,230);  
    Rectangle(150,300,230,150); // дверь  
    Brush.Color := clGray;
```

```
Polygon([Point(150,300),Point(150,150),Point(210,160),Point(210,300)]);  
end;  
end;
```



Задание. Придумайте свой рисунок и напишите к нему программу, максимально используя графические возможности Lazarus.

При создании рисунка следует учитывать размеры и особенности компонента (ось ОХ направлена слева направо, ее координаты лежат в пределах от 0 до Width; ось ОУ направлена вниз, ее координаты находятся в пределах от 0 до Height). Значения координат X и Y должны быть целыми.

Рекомендуемая литература:

Основная литература:

1. Грузина, Э.Э. Практикум по программированию / Э.Э. Грузина, Н.Л. Черноусова ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Кемеровский государственный университет». - Кемерово : Кемеровский государственный университет, 2013. - Ч. 1. - 100 с.: ил. - Библиогр. в кн. - ISBN 978-5-8353-1604-5. - ISBN 978-5-8353-1605-2 (Ч. 1); То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=278837>
2. Павлова О.А. Решение задач на ЭВМ: MathCAD [Электронный ресурс] : практикум / О.А. Павлова. — Электрон. текстовые данные. — Саратов: Вузовское образование, 2018. — 53 с. — 978-5-4487-0240-2. — Режим доступа: <http://www.iprbookshop.ru/75275.html>
3. Задохина Н.В. Информатика и математика. Решение логико-познавательных задач [Электронный ресурс]: учебное пособие для студентов вузов / Н.В. Задохина. — Электрон. текстовые данные. — М.: ЮНИТИ-ДАНА, 2015. —

127 с. — 978-5-238-02661-9. — Режим доступа:
<http://www.iprbookshop.ru/34474.html>

Дополнительная литература:

1. Аляев, Ю. А. Алгоритмизация и языки программирования Pascal, C++, Visual Basic : учебно-справочное пособие / Ю. А. Аляев, О. А. Козлов. - М. : Финансы и статистика, 2007. - 320 с. : ил. - Библиогр.: с. 318-319. - ISBN 978-5-279-02294-6, экземпляров 30
2. Гавришина, О. Н. Технология программирования на Visual Basic for Application в MsOffice / О.Н. Гавришина. - Кемерово : Кемеровский государственный университет, 2010. - 98 с. - ISBN 978-5-8353-0993-1, экземпляров неограничено
3. Жарков, В. А. Visual Basic 2005, DirectX 9.0c и Microsoft Agent в компьютерной графике, мультимедиа и играх / В. А. Жарков. - М. : Жарков Пресс, 2008. - 822 с. : ил. - Библиогр.: с. 816-817. - Предм. указ.: с. 817-822. - ISBN 5-94212-011-0, экземпляров неограничено
4. Зеньковский, В. А. Программирование на Visual Basic 6.5 и Visual Basic.Net : учебное пособие / Зеньковский В. А. - Москва : СОЛОН-ПРЕСС, 2009. - 248 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-98003-260-6, экземпляров неограничено
5. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход / С.В. Зыков. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 189 с. - (Основы информационных технологий). - ISBN 5-9556-0009-4, экземпляров неограничено
6. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход / С.В. Зыков. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 189 с. - (Основы информационных технологий). - ISBN 5-9556-0009-4, экземпляров неограничено
7. Кауфман, В. Ш. Языки программирования : Концепции и принципы / В. Ш. Кауфман. - Москва : ДМК Пресс, 2011. - 464 с. : ил. - (Классика программирования). - ISBN 978-5-94074-622-5, экземпляров неограничено
8. Кукушкина, Е. В. Начальные сведения о языке программирования Visual Basic for Application / Е.В. Кукушкина. - Екатеринбург : Издательство Уральского университета, 2014. - 111 с. - ISBN 978-5-7996-1287-0, экземпляров неограничено
9. Лавлинский, В. В. Технология программирования на современных языках программирования / В.В. Лавлинский ; О.В. Коровина. - Воронеж : Воронежская государственная лесотехническая академия, 2012. - 118 с., экземпляров неограничено

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. alglib.sources.ru - Библиотека алгоритмов (численные методы и алгоритмы вычислительной математики)

2. algolist.manual.ru - Алгоритмы, методы, исходники
3. <http://book.kbsu.ru/> - Л. З. Шауцукова «Информатика» (теория, практика)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для обучающихся по организации и проведению самостоятельной работы
по дисциплине

«ПРАКТИКУМ ПО РЕШЕНИЮ ЗАДАЧ НА ЭВМ»

44.03.05 Педагогическое образование (с двумя профилями подготовки)
направленность (профиль):
«Информатика и математика»

Ставрополь, 2026

СОДЕРЖАНИЕ

1. Общие положения.....	97
2. Цель и задачи самостоятельной работы	97
3. Порядок выполнения самостоятельной работы студентом.....	98
3.1. Методические рекомендации по работе с учебной литературой	98
3.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям	100
3.3. Методические рекомендации по самопроверке знаний	100
3.4. Методические рекомендации по написанию научных текстов (докладов, рефератов, эссе, научных статей и т.д.)	100
3.5. Методические рекомендации по выполнению исследовательских проектов.....	104
3.6. Методические рекомендации по подготовке к экзаменам и зачетам	106
4. Контроль самостоятельной работы студентов	108
5. Список литературы для выполнения СРС	108

1. Общие положения

Самостоятельная работа - планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

- формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);
- написание докладов;
- подготовка к семинарам, практическим и лабораторным работам, их оформление;
- составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);
- выполнение учебно-исследовательских работ, проектная деятельность;
- подготовка практических разработок и рекомендаций по решению проблемной ситуации;
- выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;
- компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;
- выполнение курсовых работ (проектов) в рамках дисциплин;
- выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине «Практикум по решению задач на ЭВМ» направлена на формирование следующих **компетенций**:

2. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование набора общенаучных, профессиональных и специальных компетенций будущего бакалавра по направлению подготовки «Педагогическое образование».

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на практических и лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

3. Порядок выполнения самостоятельной работы студентом

3.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист

помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того, насколько осознанно читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют **четыре основные установки в чтении научного текста:**

информационно-поисковый (задача – найти, выделить искомую информацию)

усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)

аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

Основные виды систематизированной записи прочитанного:

Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова.

При записи не забудьте вынести справочные данные на поля конспекта;

2. Выделите главное, составьте план;

3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;

4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.

5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

3.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям

Для того чтобы практические и лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

3.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

3.4. Методические рекомендации по написанию научных текстов (докладов, рефератов, эссе, научных статей и т.д.)

Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время.

Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время, ведь, как известно, свободное (от всяких глупостей) время – важнейшее условие настоящего творчества, для него наконец-то появляется время. Иногда именно на организацию такого времени уходит немалая часть сил и талантов.

Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст. Каждый раз надо представлять, что ваш текст будет кто-то читать и ему захочется сориентироваться в нем, быстро находить ответы на интересующие вопросы (заодно представьте себя на месте такого человека). Понятно, что работа, написанная «сплошным текстом» (без заголовков, без выделения крупным шрифтом наиболее важных мест и т. п.), у культурного читателя должна вызывать брезгливость и даже жалость к автору (исключения составляют некоторые древние тексты, когда и жанр был иной и к текстам относились иначе, да и самих текстов было гораздо меньше – не то, что в эпоху «информационного взрыва» и соответствующего «информационного мусора»).

Объем текста и различные оформительские требования во многом зависят от принятых в конкретном учебном заведении порядков.

Реферат (доклад) – это самостоятельное исследование студентом определенной проблемы, комплекса взаимосвязанных вопросов.

Реферат не должна составляться из фрагментов статей, монографий, пособий. Кроме простого изложения фактов и цитат, в реферате должно проявляться авторское видение проблемы и ее решения.

Рассмотрим основные этапы подготовки реферата студентом.

Выполнение реферата начинается с выбора темы.

Затем студент приходит на первую консультацию к руководителю, которая предусматривает:

- обсуждение цели и задач работы, основных моментов избранной темы;
- консультирование по вопросам подбора литературы;
- составление предварительного плана.

Следующим этапом является работа с литературой. Необходимая литература подбирается студентом самостоятельно.

После подбора литературы целесообразно сделать рабочий вариант плана работы. В нем нужно выделить основные вопросы темы и параграфы, раскрывающие их содержание.

Составленный список литературы и предварительный вариант плана уточняются, согласуются на очередной консультации с руководителем.

Затем начинается следующий этап работы – изучение литературы. Только внимательно читая и конспектируя литературу, можно разобраться в основных вопросах темы и подготовиться к самостоятельному (авторскому) изложению содержания реферата. Конспектируя первоисточники, необходимо отразить основную идею автора и его позицию по исследуемому вопросу, выявить проблемы и наметить задачи для дальнейшего изучения данных проблем.

Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.

Рабочий вариант текста реферата предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление. После доработки реферат сдается на кафедру для его оценивания руководителем.

Требования к написанию реферата

Написание 1 реферата является обязательным условием выполнения плана СРС по любой дисциплине профессионального цикла.

Тема реферата может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласовать с преподавателем.

Реферат должен быть написан научным языком.

Объем реферата должен составлять 20-25 стр.

Структура реферата:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.

- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.

- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса

- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.

- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- реферат должен быть представлен в сброшюрованном виде.

Порядок защиты реферата:

Защита реферата проводится на практических занятиях, после окончания работы студента над ним и исправления всех недочетов, выявленных преподавателем в ходе консультаций. На защиту реферата отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите реферата приветствуется использование мультимедиа-презентации.

Оценка реферата

Реферат оценивается по следующим критериям:

- соблюдение требований к его оформлению;
- необходимость и достаточность для раскрытия темы приведенной в тексте реферата информации;
- умение студента свободно излагать основные идеи, отраженные в реферате;
- способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.

Описание шкалы оценивания

Максимально возможный балл за весь текущий контроль устанавливается равным 55. Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	100
Хороший	80
Удовлетворительный	60
Неудовлетворительный	0

Темы рефератов

1. Алгоритмы определения алгебраических свойства чисел.
2. Поиск и замена элементов строки.
3. Алгоритм выделения подстроки.
4. Взаимное преобразование текстовой и числовой информации.
5. Построение последовательностей чисел.
6. Выделение подпоследовательностей.
7. Простые алгоритмы сортировки (методы “пузырька”, обмена и др.).
8. Сложная сортировка. Алгоритм бинарной сортировки. Дерево сортировки.
9. Алгоритмы поиска. Бинарный поиск.
10. Рекурсия и ее свойства.
11. Алгоритмы использующие рекурсию.
12. Файлы последовательного и прямого доступа.
13. Работа с текстовым файлом.
14. Работа с типизированным файлом.
15. Работа с базами данных.
16. Выборка и фильтрация данных файла.
17. Алгоритмы графического построения геометрических фигур.
18. Алгоритм построения графика функции.
19. Преобразование координат.
20. Алгоритмы построения 3-х мерного изображения.
21. Статические и динамические структуры данных.
22. Указатели и их использование.
23. Стек и очередь, организация и использование.
24. Построение односвязного списка.
25. Построение двусвязного списка.
26. Алгоритмы обработки списков.
27. Иерархия классов. Проектирование иерархии классов.

28. Представление графа в ЭВМ.
29. Задача Прима-Краскала и ее решение.
30. Задача Дейкстры и ее решение.

3.5. Методические рекомендации по выполнению исследовательских проектов

Исследовательская проектная работа – это групповая работа, для выполнения которой необходим выбор и приложение научной методики к поставленной задаче, получение собственного теоретического или экспериментального материала, на основании которого необходимо провести анализ и сделать выводы об исследуемом явлении. Выполнение проекта – это всегда коллективная, творческая практическая работа, предназначенная для получения определенного продукта или научно-технического результата. Такая работа подразумевает четкое, однозначное формирование поставленной задачи, определение сроков выполнения намеченного, определение требований к разрабатываемому объекту.

Выполнение 1 группового проекта является обязательным условием выполнения самостоятельной работы по любой дисциплине профессионального цикла. Тема проектного задания может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласовать с преподавателем.

Требования по выполнению и оформлению проекта

При выполнении проекта приветствуется работа в группе (2-3 человека). Проект – это исследовательская работа, в ходе которой студенты должны продемонстрировать владение навыками научного исследования, умения проводить анализ, обобщать информацию, делать выводы, предлагать свои решения проблемы, рассматриваемой в проекте.

При подготовке материалов проекта студенты должны продемонстрировать владение современными методами компьютерной обработки данных.

Критерии оценки работы участника проекта.

Для каждого из участников проекта оцениваются:

- профессиональные теоретические знания в соответствующей области;
- умение работать со справочной и научной литературой, осуществлять поиск необходимой информации в Интернет;
- умение работать с техническими средствами;
- умение пользоваться соответствующими выполняемому проекту информационными технологиями;
- умение готовить материалы проекта для презентации: составлять и редактировать тексты, формировать презентацию проекта;
- умение работать в команде;
- умение публично представлять результаты собственной деятельности;
- коммуникабельность, инициативность, творческие способности.

Критерии выставления оценки участникам проекта

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
«Отлично»	Работа выполнена на высоком профессиональном уровне. Представленный материал в основном фактически верен, допускаются негрубые фактические неточности. Студент свободно отвечает на вопросы, связанные с проектом.	Технические средства и информационные технологии освоены и использованы для реализации проекта полностью	Студент проявил инициативу, творческий подход, способность к выполнению сложных заданий, навыки работы в коллективе, организационные способности.	Проект представлен полностью и в срок.
«Хорошо»	Работа выполнена на достаточно высоком профессиональном уровне. Допущено до 4–5 фактических ошибок. Студент отвечает на вопросы, связанные с проектом, но недостаточно полно.	Обнаруживаются некоторые ошибки в использовании соответствующих технических средств и информационных технологий	Студент достаточно полно, но без инициативы и творческих находок выполнил возложенные на него задачи.	Проект представлен достаточно полно и в срок, но с некоторыми недоработками.
«Удовлетворительно»	Уровень недостаточно высок. Допущено до 8 фактических ошибок. Студент может ответить лишь на некоторые из заданных вопросов, связанных с проектом.	Обнаруживает недостаточное владение навыками работы с техническими средствами и соответствующим и информационным и технологиями	Студент выполнил большую часть возложенной на него работы.	Проект сдан со значительным опозданием (более недели) и не полностью
«Неудовлетворительно»	Работа не выполнена или выполнена на низком уровне. Допущено более 8 фактических	Навыков работы с техническими средствами нет, информационные	Студент практически не работал, не выполнил свои	Проект не сдан.

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
	ошибок. Ответы на связанные с проектом вопросы обнаруживают непонимание предмета и отсутствие ориентации в материале проекта.	технологии не освоены	задачи или выполнил лишь отдельные не существенные поручения в групповом проекте.	

Студенты должны: защитить проект в режиме презентации, предъявить файлы выполненного проекта, уметь рассказать о технологиях, использованных ими при выполнении проекта, дать оценку работы каждого члена группы (*если проект групповой*).

Максимально возможный балл за весь текущий контроль устанавливается равным **55**. Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	100
Хороший	80
Удовлетворительный	60
Неудовлетворительный	0

Темы индивидуальных и групповых проектов

1. Разработка программы «Клавиатурный тренажер»
2. Разработка приложения «Текстовый редактор»

3.6. Методические рекомендации по подготовке к экзаменам и зачетам

Изучение многих общепрофессиональных и специальных дисциплин завершается экзаменом. Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене студент демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Экзаменационная сессия - это серия экзаменов, установленных учебным планом. Между экзаменами интервал 3-4 дня. Не следует думать, что 3-4 дня достаточно для успешной подготовки к экзаменам.

В эти 3-4 дня нужно систематизировать уже имеющиеся знания. На консультации перед экзаменом студентов знакомят с основными требованиями, ответят на возникшие у них вопросы. Поэтому посещение консультаций обязательно.

Требования к организации подготовки к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение режима дня; сон не менее 8 часов в сутки, занятия заканчиваются не позднее, чем за 2-3 часа до сна. Оптимальное время занятий, особенно по математике - утренние и дневные часы. В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неустойчивые занятия спортом. Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая-либо лекция, необходимо во время ее восстановить (переписать ее на кафедре), обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным. В-третьих, при подготовке к экзаменам у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра. Здесь можно эффективно использовать листы опорных сигналов.

Вначале следует просмотреть весь материал по сдаваемой дисциплине, отметить для себя трудные вопросы. Обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения, используя при этом листы опорных сигналов.

Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

Правила подготовки к зачетам и экзаменам:

- Лучше сразу сориентироваться во всем материале и обязательно расположить весь материал согласно экзаменационным вопросам (или вопросам, обсуждаемым на семинарах), эта работа может занять много времени, но все остальное – это уже технические детали (главное – это ориентировка в материале!).

- Сама подготовка связана не только с «запоминанием». Подготовка также предполагает и переосмысление материала, и даже рассмотрение альтернативных идей.

- Готовить «шпаргалки» полезно, но пользоваться ими рискованно. Главный смысл подготовки «шпаргалок» – это систематизация и оптимизация знаний по данному предмету, что само по себе прекрасно – это очень сложная и важная для студента работа, более сложная и важная, чем простое поглощение массы учебной информации. Если студент самостоятельно подготовил такие «шпаргалки», то, скорее всего, он и экзамены сдавать будет более уверенно, так как у него уже сформирована общая ориентировка в сложном материале.

- Как это ни парадоксально, но использование «шпаргалок» часто позволяет отвечающему студенту лучше демонстрировать свои познания (точнее – ориентировку в знаниях, что намного важнее знания «запомненного» и «тут же забытого» после сдачи экзамена).

- Сначала студент должен продемонстрировать, что он «усвоил» все, что требуется по программе обучения (или по программе данного преподавателя), и лишь после этого он вправе высказать иные, желательно аргументированные точки зрения.

Вопросы для собеседования

1. Алгоритмы определения алгебраических свойства чисел.
2. Поиск и замена элементов строки.
3. Алгоритм выделения подстроки.
4. Взаимное преобразование текстовой и числовой информации.
5. Построение последовательностей чисел.
6. Выделение подпоследовательностей.

7. Простые алгоритмы сортировки (методы “пузырька”, обмена и др.).
8. Сложная сортировка. Алгоритм бинарной сортировки. Дерево сортировки.
9. Алгоритмы поиска. Бинарный поиск.
10. Рекурсия и ее свойства.
11. Алгоритмы использующие рекурсию.
12. Файлы последовательного и прямого доступа.
13. Работа с текстовым файлом.
14. Работа с типизированным файлом.
15. Работа с базами данных.
16. Выборка и фильтрация данных файла.
17. Алгоритмы графического построения геометрических фигур.
18. Алгоритм построения графика функции.
19. Преобразование координат.
20. Алгоритмы построения 3-х мерного изображения.
21. Статические и динамические структуры данных.
22. Указатели и их использование.
23. Алгоритмы обработки списков.
24. Стек и очередь, организация и использование.
25. Построение односвязного списка.
26. Построение двусвязного списка.
27. Типы задач по информатике
28. Классификация задач по информатике по различным признакам
29. Качественные задачи по информатике
30. Количественные задачи по информатике
31. Основные этапы решения задач
32. Методика решения задач по теме: «Работа с массивами и матрицами в языке программирования».
33. Основные типы задач по теме: «Работа с массивами и матрицами в языке программирования».
34. Основные типы задач по теме: «Выполнение алгоритмов для исполнителя».
35. Методика решения задач по теме: «Выполнение алгоритмов для исполнителя».
36. Методика решения задач по теме: Анализ программы.
37. Основные типы задач по теме: «Рекурсивные алгоритмы».
38. Методика решения задач по теме: «Анализ программы, содержащей подпрограммы, циклы и ветвления».
39. Методика проведения анализа программы содержащей подпрограммы.
40. Обучение исправлению ошибок в простой программе с условными операторами.

4. Контроль самостоятельной работы студентов

Контроль самостоятельной работы проводится преподавателем в аудитории.

Предусмотрены следующие виды контроля: собеседование, оценка реферата, оценка презентации, оценка участия в круглом столе, оценка выполнения проекта.

Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

5. Список литературы для выполнения СРС

Основная литература:

4. Грузина, Э.Э. Практикум по программированию / Э.Э. Грузина, Н.Л. Черноусова ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Кемеровский государственный университет». - Кемерово : Кемеровский государственный университет, 2013. - Ч. 1. - 100 с.: ил. - Библиогр. в кн. - ISBN 978-5-8353-1604-5. - ISBN 978-5-8353-1605-2 (Ч. 1); То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=278837>
5. Павлова О.А. Решение задач на ЭВМ: MathCAD [Электронный ресурс] : практикум / О.А. Павлова. — Электрон. текстовые данные. — Саратов: Вузовское образование, 2018. — 53 с. — 978-5-4487-0240-2. — Режим доступа: <http://www.iprbookshop.ru/75275.html>
6. Задохина Н.В. Информатика и математика. Решение логико-познавательных задач [Электронный ресурс]: учебное пособие для студентов вузов / Н.В. Задохина. — Электрон. текстовые данные. — М.: ЮНИТИ-ДАНА, 2015. — 127 с. — 978-5-238-02661-9. — Режим доступа: <http://www.iprbookshop.ru/34474.html>

Дополнительная литература:

10. Аляев, Ю. А. Алгоритмизация и языки программирования Pascal, C++, Visual Basic : учебно-справочное пособие / Ю. А. Аляев, О. А. Козлов. - М. : Финансы и статистика, 2007. - 320 с. : ил. - Библиогр.: с. 318-319. - ISBN 978-5-279-02294-6, экземпляров 30
11. Гавришина, О. Н. Технология программирования на Visual Basic for Application в MsOffice / О.Н. Гавришина. - Кемерово : Кемеровский государственный университет, 2010. - 98 с. - ISBN 978-5-8353-0993-1, экземпляров неограничено
12. Жарков, В. А. Visual Basic 2005, DirectX 9.0c и Microsoft Agent в компьютерной графике, мультимедиа и играх / В. А. Жарков. - М. : Жарков Пресс, 2008. - 822 с. : ил. - Библиогр.: с. 816-817. - Предм. указ.: с. 817-822. - ISBN 5-94212-011-0, экземпляров неограничено
13. Зеньковский, В. А. Программирование на Visual Basic 6.5 и Visual Basic.Net : учебное пособие / Зеньковский В. А. - Москва : СОЛОН-ПРЕСС, 2009. - 248 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-98003-260-6, экземпляров неограничено
14. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход / С.В. Зыков. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 189 с. - (Основы информационных технологий). - ISBN 5-9556-0009-4, экземпляров неограничено
15. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход / С.В. Зыков. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 189 с. - (Основы информационных технологий). - ISBN 5-9556-0009-4, экземпляров неограничено
16. Кауфман, В. Ш. Языки программирования : Концепции и принципы / В. Ш. Кауфман. - Москва : ДМК Пресс, 2011. - 464 с. : ил. - (Классика программирования). - ISBN 978-5-94074-622-5, экземпляров неограничено
17. Кукушкина, Е. В. Начальные сведения о языке программирования Visual Basic for Application / Е.В. Кукушкина. - Екатеринбург : Издательство Уральского университета, 2014. - 111 с. - ISBN 978-5-7996-1287-0, экземпляров неограничено
18. Лавлинский, В. В. Технология программирования на современных языках программирования / В.В. Лавлинский ; О.В. Коровина. - Воронеж : Воронежская государственная лесотехническая академия, 2012. - 118 с., экземпляров неограничено

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

4. alglib.sources.ru - Библиотека алгоритмов (численные методы и алгоритмы вычислительной математики)
 5. algolist.manual.ru - Алгоритмы, методы, исходники
- <http://book.kbsu.ru/> - Л. З. Шауцукова «Информатика» (теория, практика)