МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»
НОВОТРОИЦКИЙ ФИЛИАЛ

КАФЕДРА ЭЛЕКТРОЭНЕРГЕТИКИ И ЭЛЕКТРОТЕХНИКИ

Р.Р. Абдулвелеева

Объектно-ориентированное программирование в среде Lazarus

Лабораторный практикум

для студентов направлений подготовки
09.03.03 Прикладная информатика,
13.03.02 Электроэнергетика и электротехника,
15.03.02 Технологические машины и оборудование,
13.03.01 Теплоэнергетика и теплотехника,
22.03.02 Металлургия,
18.03.01 Химическая технология,
очной и заочной форм обучения

Новотроицк, 2020 г.

УДК 004.4

ББК 32.97

А14

Рецензенты:

Доцент ФГАОУ ВО «Национальный исследовательский технологический университет «МИСиС», к.т.н. Лицин К.В.

Доцент ОГТИ (филиал) ФГБОУ ВО «Оренбургский государственный университет», к.ф.-м.н. Абрамов С.М.

Абдулвелеева Р.Р. Объектно-ориентированное программирование в среде Lazarus: лабораторный практикум. – Новотроицк: НФ НИТУ «МИСиС», 2020. – 70 с.

Лабораторный практикум предназначен для студентов направлений подготовки 09.03.03 Прикладная информатика, 13.03.02 Электроэнергетика и электротехника, 15.03.02 Технологические машины и оборудование и 13.03.01 Теплоэнергетика и теплотехника, 22.03.02 Металлургия, 18.03.01 Химическая технология, изучающих раздел «Объектно-ориентированное программирование в среде Lazarus» дисциплины «Информатика». В практикуме рассмотрено применение программы Lazarus для разработки приложений. Приведены лабораторные работы по темам: «Структура среды программирования Lazarus. Создание Windows-приложений», «Программирование задач линейной структуры», «Программирование алгоритмов разветвляющейся структуры», «Программирование алгоритмов разветвляющейся структуры. Оператор множественного выбора».

Рекомендовано Методическим советом НФ НИТУ «МИСиС»

ФГАОУ ВО «Национальный
исследовательский технологический
университет «МИСиС»
Новотроицкий филиал, 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В учебном пособии предлагается описание процесса использования принципов и методов объектно-ориентированного программирования на примере разработки форм, диалоговых окон и приложений. Предложены коды и скриншоты программных продуктов в качестве образцов для выполнения лабораторных работ, даны пошаговые инструкции по выполнению алгоритма программ с применением языка программирования Lazarus. Представлены формулировки заданий с кодами и скриншотами по реализации лабораторных работ. Все лабораторные работы выполняются с использованием Lazarus.

В первой лабораторной работе изучается структура среды программирования, назначение основных окон среды Lazarus, использование компонентов, описание процедур обработки событий.

Во второй лабораторной работе изучаются основные типы данных, операции, функции их обработки, операторы и процесс создание программ линейной структуры.

В третьей лабораторной работе изучается процесс использования алгоритмической структуры – ветвление, логические операторы, условный оператор (сокращенная и полная форма) в процессе разработки приложений.

В четвертой лабораторной работе рассматривается процесс использования оператора множественного выбора, компоненты RadioGroup, ListBox, Memo и свойства этих компонентов.

Вариант лабораторных работ выбирается в соответствии с номером компьютера либо номером студента в списке группы, что указано в заданиях для самостоятельной работы.

Защита лабораторных работ проводится устно, для этого студент должен иметь отчет о проведенной работе. Отчет должен содержать:

- название работы;
- цель работы;
- теоретические сведения;
- структурные блок-схемы разработанных приложений, выполненные в среде MSVisio;
- результаты выполнения практических упражнений в виде файлов проектов;
- демонстрацию результатов тестирования программы в соответствии с заданием;
- ответы на контрольные вопросы;
- библиографический список.

ЛАБОРАТОРНАЯ РАБОТА № 1
СТРУКТУРА СРЕДЫ ПРОГРАММИРОВАНИЯ LAZARUS. СОЗДАНИЕ WINDOWS-ПРИЛОЖЕНИЙ

Цель работы: изучить структуру среды программирования *Lazarus* и научиться создавать Windows-приложения.

1 Теоретическое введение

1.1 Концепции объектно-ориентированного программирования

Метод программирования, реализующий объекты в виде главных элементов программы, называется объектно-ориентированным программированием (ООП). *Объект* – основное понятие ООП. *Объект* – совокупность свойств (данных данного объекта), методов их обработки (подпрограмм) и событий (реакций объекта), приводящие к изменению свойств объекта.

Класс – фундаментальное понятие ООП. Класс является описанием поведения его представителя. Класс – это шаблон, программного объекта, в котором описаны свойства и методы, задающие поведение объектов этого класса. *Экземпляр* класса - *объект*, имеет структуру этого класса. Каждый объект является представителем (экземпляром) определенного класса.

Иногда под термином класс понимают тип структурированных данных, в этом случае объект будет являться структурированной переменной типа класс. В ней содержится информация о предмете или понятии, реализуемом в данной программе.

В описании класса нельзя определять конкретные значения полей (элементов), входящих в этот класс. Это вызовет мгновенную ошибку компилятора объектно – ориентированного языка. *Класс* определяет только общий тип объекта. Далее в программе необходимы выполнения действий над значениями в том числе и над полями описанного класса. Помимо всех методов, входящих кроме полей, функцию (*set* - функцию) инициализации (присвоения) значений класса должен выполнить специальный метод данного класса – конструктор. Эта функция создаёт конкретный экземпляр класса – объект. *Конструктор* – метод, имеющий имя класса, и запускается столько раз, сколько разных объектов одного и того же класса нам необходимо для работы программы. А если при этом конструктором был динамически выделен участок памяти под данный экземпляр класса и его необходимо очистить, применяют специальный метод – деструктор. *Деструктор* запускается только один раз. Объекты одного и того же класса имеют поля и методы схожие по своим свойствам и правилам поведения. Говорят, что объекты имеют одинаковый интерфейс. Интерфейс класса иногда называют особенностями класса.

Объект — это частный случай обобщённой структуры ООП - *класса* (участок памяти), который содержит данные и правила (методы с кодом алгоритма) определяющие:

− *поля* объекта, то есть атрибуты исходных данных, своего рода, переменная – посредник между самим обобщённым классом и пользователем, который определяет текущее свойство данного объекта через общение с этой самой переменной;

− *методы* объекта представляют собой определённое количество функций и процедур, которые в ответ на их вызов (переданное сообщение члену объекта) выполняют тот или иной алгоритм действий над этими членами экземпляра класса;

− *свойства* того или иного объекта объявляются для получения и передачи атрибутов данных описания индивидуальности объекта посредством специальных переменных (*property*).

Объявить класс означает ввести необходимые свойства объектов: поля, методы и свойства объектов с определением предка данного класса в том числе.

Известно, что у объекта поля и функции по интерфейсу определяются как *public*–открытые, и как *private*– закрытые или приватные.

Выполнение объектно-ориентированной программы представляется последовательным вызовом сообщений (методов) соответствующих объектов. Объект при этом начинает выполнение кода определённой функции (процедуры) с полученными значениями переменных. Послать сообщения объекту в коде программы означает записать имя метода по его имени и с заданными аргументами непосредственно после указания имени вызываемого объекта. В этом и заключается такое взаимодействие объектов как передача сообщений между этими объектами.

ООП предполагает, что любая функция (процедура) со структурированными переменными в программе успешно определяют объект некоторого класса как типа данных либо уже имеющегося в компиляторе, либо полученного из уже существующих, для определения любого задуманного физического предмета через его абстрагирование в виде одного класса и с использованием уже существующих классов (встроенных или пользовательских) – программирование "от класса к классу".

В основу ООП положены принципы программирования: *наследование, инкапсуляция, полиморфизм*.

Наследование – позволяет новому классу – потомку, созданному на базе уже существующего иметь (наследовать) все свойства класса-родителя.

Инкапсуляция – данные и свойственные им процедуры обработки, объединенные в одном объекте.

Полиморфизм (греч., *poly* – много, *morphos* – форма) – много форм. *Полиморфизм* – возможность различных реализаций одного и того же класса с одинаковой спецификацией (общими свойствами) в зависимости от типа обрабатываемых дан-

ных, т. е. полиморфизм определяет разнообразные формы реализации действий с одинаковыми именами.

Полиморфизм означает информированность объектов об используемых методах в зависимости от месторасположения.

Модульность – свойство программ, которое позволяет осуществлять свободное копирование и внедрение одного объекта в другие. Модуль включает в себя объекты с полным определением их характеристик, методов и свойств.

В основе всех современных сред программирования лежат концепции ООП. И все среды разработки являются средами визуального проектирования, в том числе и *Lazarus*. Основное преимущество сред визуального проектирования – освобождение разработчика программ от необходимости рутинной работы по созданию стандартизированных элементов интерфейса разрабатываемого проекта (кнопок, меню, строк ввода и т.д.). Эти элементы уже созданы разработчиком среды и их состав, свойства, принципы использования являются практически промышленным стандартом и одинаковы в любой среде разработки современных программ.

1.2 Структура среды программирования Lazarus

Среда программирования *Lazarus* состоит из пяти отдельно расположенных окон (рисунок 1.1.).

Окна могут перемещаться по экрану.

После запуска *Lazarus* появятся основные окна:

1. Главное окно (Project1).
2. Окно формы (*Form1*).
3. Окно Инспектора Объектов (*ObjectInspector*).
4. Окно кода программы или окно редактора кода (*Unit1.pas*).
5. Окно сообщений.

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

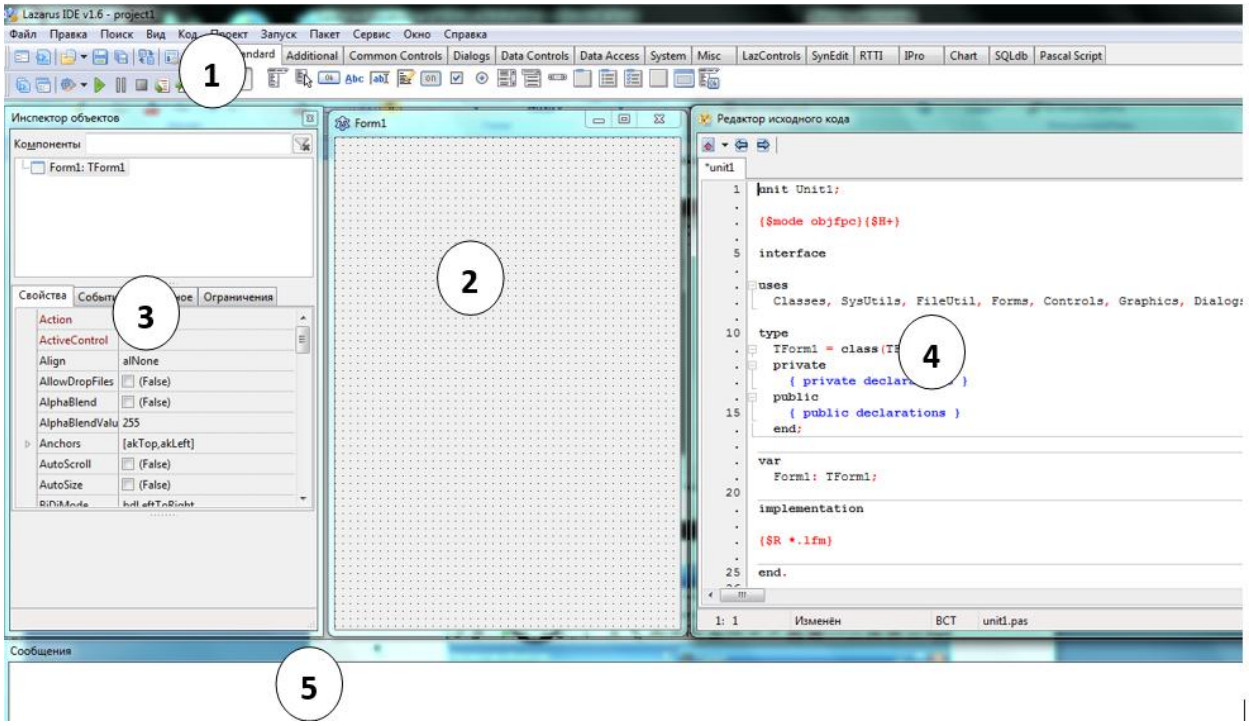


Рисунок 1.1 – Главное окно среды программирования *Lazarus*

Все объекты реализованы в Lazarus в виде палитры визуальных компонентов (рисунок 1.2).

Рисунок 1.2 – Палитра компонентов Lazarus

Компоненты сгруппированы в виде закладок. Щелчком мыши на закладке, происходит переход на соответствующую ей страницу. При подведении курсора мыши к компоненте (или нажатие правой кнопки мыши на компоненте) появляется всплывающая подсказка – название компоненты.

Окно формы (элемент 2 на рисунке 1.1) – окно разрабатываемого проекта (окно Windows-приложения): имеет заголовок, кнопку вызова системного меню, кнопку максимизации, минимизации и закрытия окна, рамку окна. На форме будут размещаться компоненты образуя интерфейс будущей программы.

Окно Инспектора Объектов (элемент 3 на рисунке 1.1) содержит страницы: Свойства (*Properties*) и События (*Events*), каждая из которых разделена на две части.

Страница Свойства служит для указания свойств активного (выделенного) объекта. В левой части страницы находится название свойства, а в правой – его значение. Значениями свойств могут быть слова, числа, значения из раскрывающегося списка. Щелчок на «▼» справа от поля значения свойства, раскрывает список значений. При нажатии на «...» вызывается специальное диалоговое окно. Если значе-

нием свойства является число или текст, то после его набора необходимо нажать Enter. При нажатии на Esc ввод отменяется.

Страница События используется для задания реакции на событие, состоит из двух частей. В первой – название события, а во второй – название процедур, обрабатывающих данное событие. Для создания новой реакции на событие необходимо дважды щелкнуть в правой половине напротив нужного события, появится название новой процедуры и откроется окно редактора кода.

Окно кода программы, фрагмент которого представлен на рисунке 1.3, предназначено для набора текста программы на языке программирования, и требует соблюдение совокупности правил записи текста. В системе программирования Lazarus используется язык программирования FreePascal, ObjectPascal.

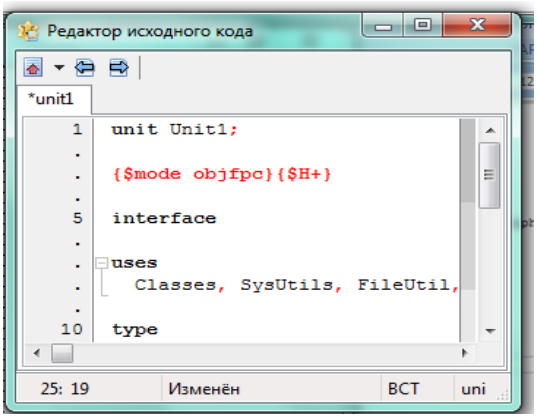


Рисунок 1.3 – Окно редактора кода Lazarus

Для разработки приложения, а значит написания программы необходимо:

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

Чтобы разместить компоненты на форме нужно щелкнуть мышью на компоненте, затем щелкнуть на форме.

Опишем некоторые компоненты. Все компоненты имеют свойство *Name* (имя). Имена создаются в *Lazarus* так: 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, при этом желательно использовать только английские буквы и цифры. Свойство *Name* задается первоначально, это имя переменной, с которой оперируют при написании программы.

Компонента *Form* (экранная форма) задает внешний вид окна приложения, и в свою очередь сама является полноценным компонентом с собственными свойствами

и событиями, но на палитре компонентов ее нет. Основные свойства и события компоненты Form приведены в таблицах 1.1 и 1.2.

Таблица 1.1 – Свойства компоненты Form

Свойство	Значение свойства
Caption	Задает заголовок окна формы
Color	Задает цвет формы
Font	Задает атрибуты шрифта формы

Значения свойств задается в окне свойств объекта или в коде программе.

Пример использования в программе:

Form1.Color:=clblue; {задание голубого цвета формы}

Таблица 1.2 – События компоненты Form

Событие	Значение события
OnCreate	Создание формы
OnClose	Закрытие формы

Компонент *Label* (надпись или метка) используют для ввода названия, текста, пояснения вводимых данных. Основные свойства компонента Label приведены в таблице 1.3.

Пример использования в программе:

Label1.Caption:='Результат';

Таблица 1.3 – Свойства компоненты Label

Свойство	Значение свойства
Caption	Задает заголовок надписи, выводимой на экран
Visible	Задает видимость надписи на экране. Имеет значение True - надпись видна или False – нет.
Font	Задает шрифт, используемый для отображения текста
Alignment	Расположение текста в метке: taRightJustify – текст прижат к правому краю; taCenter – текст размещен по центру; taLeftJustify – текст прижат к левому краю.

Компонент *Edit* (однострочное окно ввода/вывода) используется для ввода/вывода чисел и текста на форму. Основные свойства и методы компонента Edit приведены в таблицах 1.4 и 1.5.

Таблица 1.4 – Свойства компоненты Edit

Свойство	Значение свойства
AutoSize	Задает необходимость изменения размера компонента при изменении размера шрифта (если True)
Text	Задает содержимое строки редактирования (изначально задается пустым)

Таблица 1.5 – Основные методы компонента Edit

Метод	Значение метода
Clear	Удаляет весь текст
SetFocus	Устанавливает фокус ввода

Компонент *Panel* (контейнер общего назначения) используется для размещения на ней сгруппированных компонент. Основные свойства компонента Panel приведены в таблице 1.6.

Таблица 1.6 – Свойства компоненты Panel

Свойство	Значение свойства
AutoSize	Задает необходимость изменения размера компонента при изменении размера шрифта (если True)
BevelOuter	Внешняя рамка: bvRaised (выпуклая); bvLowered (вдавленная); bvNone (отсутствует)
BevelInner	Внутренняя рамка: bvRaised (выпуклая); bvLowered (вдавленная); bvNone (отсутствует)
BevelWidth	Ширина рамки

Компонент *Button* (командная кнопка) используется для задания реакции на событие. Основные свойства и события компонента Button приведены в таблицах 1.7 и 1.8.

Таблица 1.7 – Свойства компоненты Button

Свойство	Значение свойства
Caption	Задает название кнопки
Height	Задает высоту кнопки
Width	Задает ширину кнопки
Left	Задает расстояние от левой границы кнопки до левой границы формы
Top	Задает расстояние от верхней границы кнопки до верхней границы формы

Таблица 1.8 – Основные события компонента Button

События	Значение события
OnClick	Происходит, когда пользователь щелкает основной (левой) кнопкой мыши на объекте
OnDbClick	Происходит, когда пользователь дважды щелкает основной (левой) кнопкой мыши на объекте
OnClose	Закрытие окна формы

2 Выполнение работы

2.1 Загрузите программу *Lazarus*.

2.2 Используя свойство *Caption* изменить заголовок окна формы с *Form1* на заголовок «Режимы работы оборудования».

2.3 Используя свойство Color изменить цвет формы на *clAqua*,

2.4 Разместить в центре формы компоненту *Label*. Задать: надпись метки – РЕЖИМЫ РАБОТЫ ОБОРУДОВАНИЯ, цвет метки – серый. Изменить свойство *Font*: шрифт – *TimesNewRoman*, начертание – жирный, размер – 14.

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

2.6 Расположить на форме три командные кнопки *Button1*, *Button2*, *Button3*. Задать надписи на этих кнопках «Выкл.», «Авто», «Ручное управление». В результате должна получиться форма, показанная на рисунке 1.4.

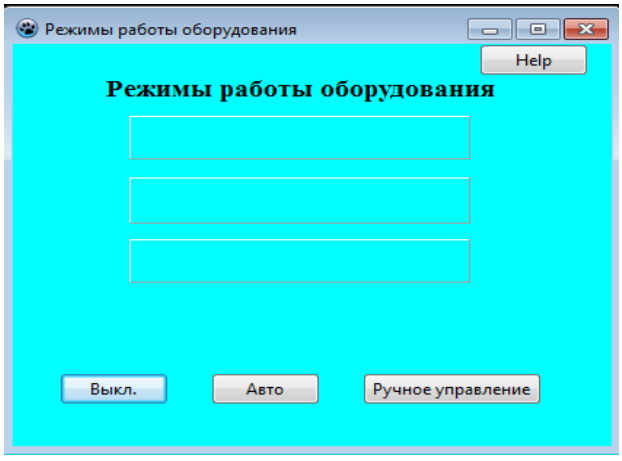


Рисунок 1.4 – Результат выполнения заданий 1-5

2.7 Зададим реакцию на события с использованием окна кода программы.

События устанавливаются в окне Инспектора объектов на странице События. Зададим в программе реакцию на событие, что при нажатии мышью (событие *OnClick*) на командные кнопки будет соответственно изменяться цвет панели.

Задать событие для первой командной кнопки *Button1*. Для этого выделить данный компонент и перейти в Инспекторе Объектов на страницу События. Затем

напротив события *OnClick* дважды щелкнуть левой кнопкой мыши (рисунок 1.5). После выбора события автоматически открывается окно кода программы.

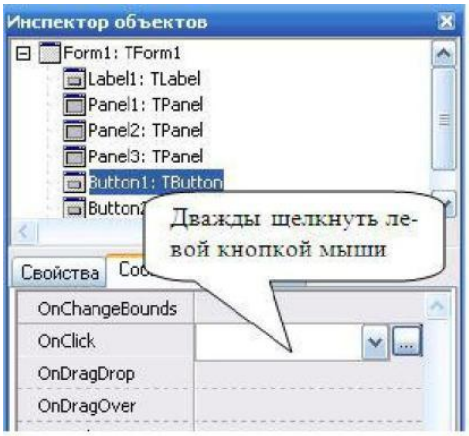


Рисунок 1.5 – Задание события *OnClick*

После задания события в коде программы автоматически создается процедура обработки события, внутри которой описывается реакция на событие. Заголовок процедуры формируется следующим образом:

ProcedureTForm1.Button1Click (Sender:TObject); {событие нажатие мыши на компонентеButton1, расположенной на форме Form1 }

Begin

….

End;

Между *Begin* и *End* указывается реакция на события – перечень совершаемых действий.

2.8 Создадим событие и реакцию на событие для командных кнопок:

а) при нажатии на командную кнопку *Button1* с надписью «Красный» цвет компоненты *Panel1* будет меняться на красный, а цвет компонент *Panel2* и *Panel3* будет меняться на белый. Записать в процедуре следующую последовательность действий:

ProcedureTForm1.Button1Click (Sender:TObject);

Begin

Panel1.Color:=clRed;

Panel2.Color:=clWhite;

Panel3.Color:=clWhite;

End;

б) для командной кнопки *Button2*: цвет компоненты *Panel2* будет меняться на желтый, а цвет компонент *Panel1* и *Panel3* будет меняться на белый. Название цвета, можно посмотреть в закладке *Color* в Инспекторе Объектов.

в) для командной кнопки *Button3*: цвет компоненты *Panel3* будет меняться на зеленый, а цвет компонент *Panel1* и *Panel2* будет меняться на белый.

2.9 Осуществим 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 проект *project1* (*.*lpi*), в котором сохраняется проект, работающим с программой, с расширением *.*lpi*.

Далее появляется диалоговое окно Сохранить *Unit1* (*.*pas*), в котором на первом этапе сохраняется программа с расширением *.*pas*.

На сетевом диске в своей папке сохраните программу.

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:

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

Если программа зависла необходимо нажать комбинацию клавиш Ctrl+F2 или выбрать команду Запуск → Останов.

2.10 Добавим событие при нажатии на командную кнопку *Button1* на компоненте *Panel1*появиться информация «ВЫКЛЮЧЕНО» белым цветом, жирным шрифтом, 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:='ВЫКЛЮЧЕНО';    {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;
```

2.11 Добавим событие при нажатии на командную кнопку *Button2* на компоненте *Panel2* появиться информация «АВТОМАТИЧЕСКОЕ УПРАВЛЕНИЕ» белым цветом, жирным шрифтом, размер шрифта – 12.

2.12 Добавим событие при нажатии на командную кнопку Button3 на компоненте Panel3 появиться информация «РУЧНОЕ УПРАВЛЕНИЕ» белым цветом, жирным шрифтом, размер шрифта – 12.

2.13 Сохранить изменения в программе и запустить ее на исполнение.

2.14 Выведем в окне программы строку сообщения. Для этого существует команда:

```
ShowMessage('текст сообщения');
```

2.15 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;
```

2.17 На форме добавить командную кнопку *Button5*. Задать для нее надпись «Help»или «Помощь». При нажатии на кнопку должно выводиться сообщение о назначении программы. Для реализации данного задания выполнить для нее реакцию на событие:

```
Procedure TForm1.Button5Click(Sender:TObject);
 begin
        ShowMessage ('Программа предназначена для управления оборудованием
        в ручном и автоматическом режимах');
end;
```

Задания для самостоятельного выполнения

Разработайте приложение, которое содержит следующие компоненты с соответствующими реакциями на событие:

- программа, при выборе соответствующей командной кнопки, изменяет содержание надписи «Компьютерные технологии» на «Информационные системы»;

- при выборе соответствующей командной кнопки, выводит сообщение «Программа будет закрыта» затем завершает работу программы;

- при выборе соответствующей командной кнопки, меняет цвет формы на голубой, зеленый, красный;

- при выборе соответствующей командной кнопки, выводится сообщение условия задачи или задание (математической, физической или занимательной), которое нужно выполнить пользователю этого приложения;

- при выборе соответствующей командной кнопки, меняет вид отображения рамки панели «выпуклая», «вдавленная», «отсутствует»;

- выводит сообщение о назначении компонентов при щелчке мыши на данной компоненте (для кнопки и панели);

- при выборе соответствующей командной кнопки, выводит сообщение об авторе программы.

Интерфейс программы должен быть красивым, эстетически комфортным для пользователя, шрифт должен быть читаемым (не мелким), программа должна быть протестирована.

3 Контрольные вопросы

3.1 Какие концепции лежат в основе ООП?

3.2 Что такое наследование, инкапсуляция и полиморфизм в ООП?

3.3 Перечислите элементы окна *Lazarus* и их назначение.

3.4 Как задать значение свойств компонент?

3.5 Как создать процедуру реакции на нажатие кнопки, помещенной на форму?

3.6 Как вывести текст на форме?

3.7 Как сохранить приложение? Как запустить созданное приложение?

3.8 С какими расширениями создаются файлы при сохранении приложения?

ЛАБОРАТОРНАЯ РАБОТА№ 2
ПРОГРАММИРОВАНИЕ ЗАДАЧ ЛИНЕЙНОЙ СТРУКТУРЫ

Цель работы: научиться программировать в *Lazarus* задачи линейной структуры.

1 Теоретическое введение

Программирование задач линейной структуры требует знание основных типов данных, операций и функции их обработки. К основным типам данных относятся: целые, вещественные и логические.

Целые типы данных используются для представления целых чисел. Различные целые типы имеют существенно различные диапазоны хранимых значений, затраты памяти растут с ростом допустимого диапазона значений. Наиболее распространенный целый тип данных - *Integer*.

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

Логический тип данных обозначается – *Boolean*. Переменные типа *Boolean* представляют собой логические значения *True* (истина) или *False* (ложь).

В программе каждая переменная перед использованием должна быть описана, так устанавливается факт существования переменной, задается ее тип, резервируется память компьютера для оперирования с ней.

Для описания переменных в программе 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;
   xl,x2: integer;
```

Для присваивания переменной вычисленного значения используется оператор *присваивания*. 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. Любой оператор языка завершается точкой с запятой. Переменные и полученный 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, затем умножение, деление, арифметические функции, а дальше сложение, вычитание. Числа записываются при помощи цифр, причем *целая часть* от *дробной* отделяется *точкой*.

Над числовыми данными допускается выполнять арифметические операции: сложение; вычитание; * - умножение; / - деление; *div* - целочисленное деление; *mod* - остаток от деления.

При записи алгебраических выражений можно использовать функции, представленные в таблице 2.1.

Если необходимо вычислить функцию, которая не входит в набор стандартных и представленных в таблице 2.1, то её можно выразить с помощью арифметических операций и стандартных функций.

Таблица 2.1 - Арифметические функции языка Pascal

Обозначение	Действие	Пример
abs(x)	модуль числа	abs(-5.3)=5.3
sqr(x)	квадрат числа	sqr(4)=16
sqrt(x)	квадратный корень	sqrt(4)=2
cos(x)	косинус	cos(π)=1
sin(x)	синус	sin(π)=0
arctan(x)	арктангенс	arctan(3.14)=1.3
exp(x)	экспонента e^x	exp(5)=148.4
ln(x)	натуральный логарифм	ln(4)=1.38
pi	число π	
int(x)	целая часть числа	int(5.4)=5
frac(x)	дробная часть числа	frac(5.6)=0.6
round(x)	округление числа	round(4.5)=5
trunc(x)	отсекание дробной части числа	trunc(4.5)=4
random(n)	случайное число от 0 до n	random(7)=4
exp(n*ln(x))	возведение числа x в степень n (x^n)	exp(1/3*ln(x))=$x^{1/3}$
ln(x)/ln(a)	логарифм числа x по основанию a ($\log_a x$)	ln(10)/ln(2)=3.3

2 Выполнение работы

2.1 Вычислить значение выражения $\frac{}{}$ для любого x.

Алгоритм для вычисления значения функции представлен на рисунке 2.1.

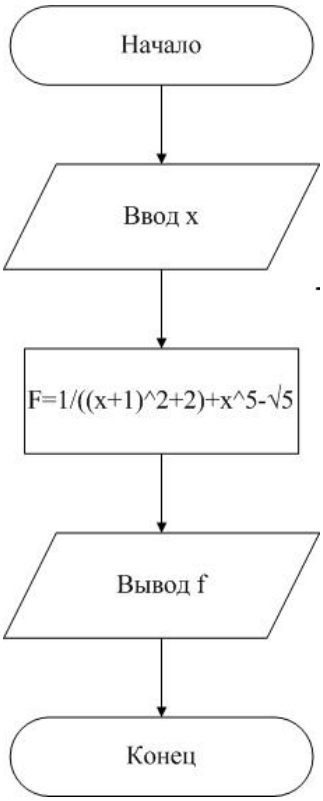


Рисунок 2.1 – Алгоритм решения задания 2.1

Решение любой задачи начинается с определения типа входных и выходных переменных: *входные переменные: x: integer; выходные переменные: f : real.*

Разместите компоненты на форме и задайте их свойства. Расположите компоненты на форме в соответствии с рисунком 2.2 и задайте свойства компонент по таблице 2.3.

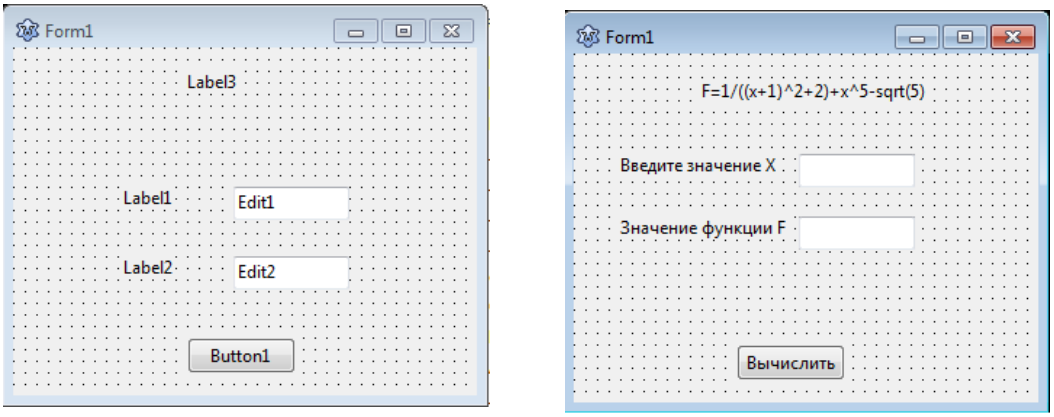


Рисунок 2.2 – Вид компонентов для задания 2.1

Вариант реализации 2.1 в приложении показан на рисунке 2.1'

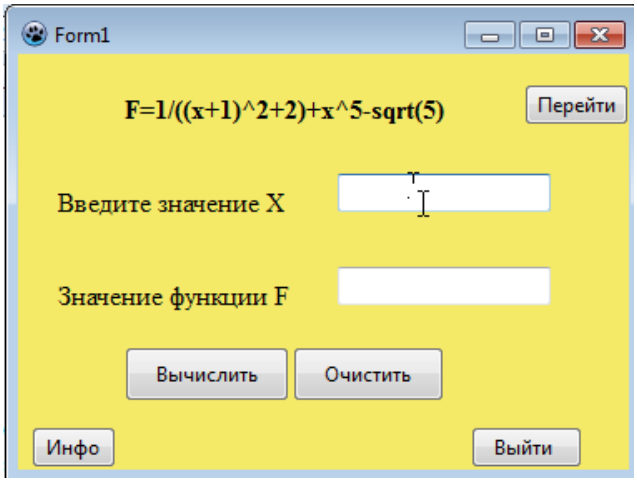


Рисунок 2.1' – Пример реализации задания 2.1

Таблица 2.3 – Свойства компонент для задания 2.1

Компонент	Свойство	Значение
Form1	*Caption*	Значение функции
Label1	*Caption*	Введите значение x
Label2	*Caption*	Значение функции f
Button1	*Caption*	Вычислить
Edit1	*Text*	пусто
Edit2	*Text*	пусто

Важно: при описании процедур 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 для вычисления значения функции будет иметь вид:

```
procedure TForm1.Button1Click(Sender: TObject);
var x:integer;f:real;
begin
x:=strtoint(edit1.text);
f:=1/(sqr(x+1)+2)+ x*x*x*x*x-sqrt(x);
edit2.text:=floattostr(f);
end;
```

2.2 Добавить на форму, представленную на рисунке 2.2, командную кнопку *Button2*. Задать надпись *Очистить*. Задать для неё реакцию на событие – при нажатии на кнопку должны очищаться компоненты *Edit1* и *Edit2* и курсор помещаться в компоненту *Edit1*.

Данная процедура будет иметь вид:

```
procedure TForm1.Button2Click(Sender:TObject);
begin
edit1.text:=''; //очистка поля ввода
edit2.text:=''; //очистка поля ввода
edit1.SetFocus;// перевод (курсора) в первую строку ввода
end;
```

2.3 Измените размер шрифта для названия компонентов на более крупный комфортный для пользователя. Измените цвет формы и компонентов. Измените заголовок формы. Добавьте кнопку для завершения работы программы. Разместите компонент с подсказкой для проверки области допустимого значения математического выражения методом тестирования программы.

2.4 Задан треугольник со сторонами *a, b* и *c*. Вычислить площадь S, периметр P и величины углов α, β и γ (в градусах) треугольника, рисунок 2.3.

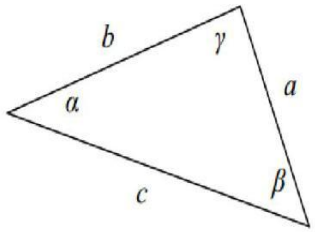


Рисунок 2.3 – Данные для задания 2

Перед началом написания программы необходимо составить математическую модель решения задачи и подобрать необходимые формулы. Для вычисления площади треугольника примените формулу Герона. Угол α можно найти по теореме косинусов, угол β - по теореме синусов. Углы будут вычислены в радианах. Переведите в градусы, тогда угол γ можно найти по теореме существования треугольника.

Определим в программе входные и выходные переменные и их тип:

− входные переменные: a, b, c (стороны),

− выходные переменные: alfa, betta, gamma (углы), S (площадь), r (полупериметр), P (периметр)

В редакторе кода запишем следующее:

a, b, c, P: integer; alfa, betta, gamma, S, r:real;

Построим алгоритм решения задачи.

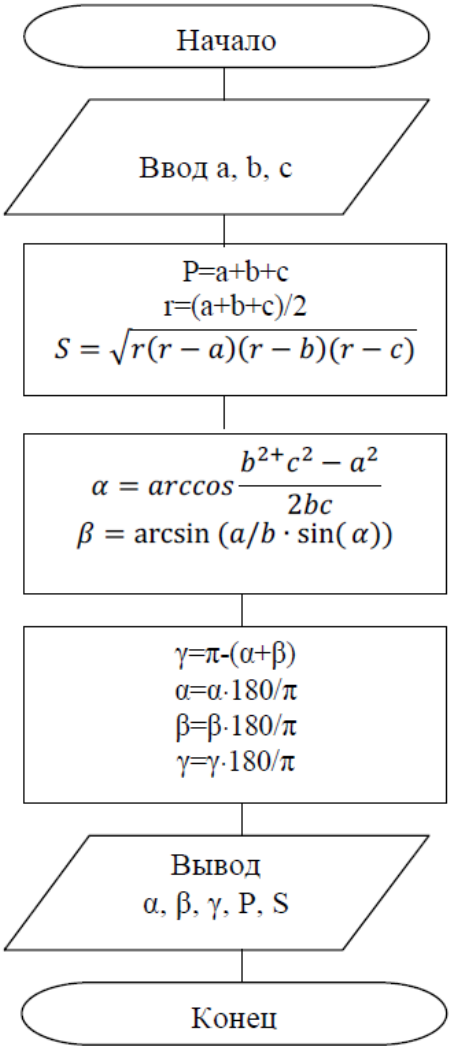


Рисунок 2.4 – Алгоритм решения задания 2.3

Размещаем компоненты на форме и задаем их свойства, в соответствии с рисунком 2.5.

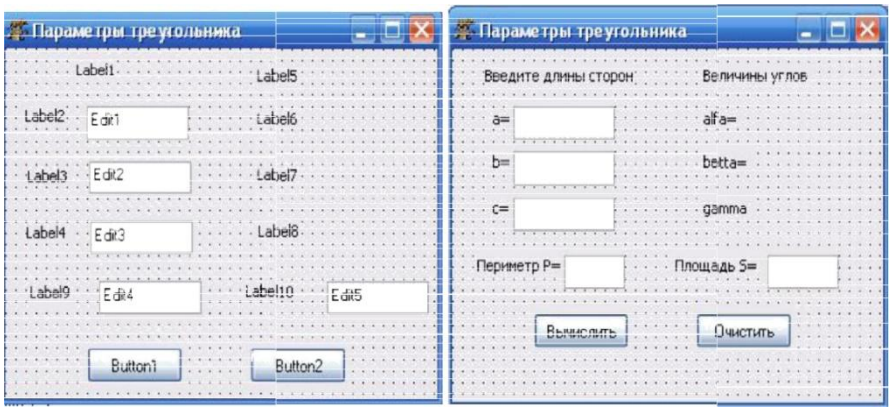


Рисунок 2.5 – Расположение компонент для задания 2.3

Процедура 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:=(a+b+c)/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);
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:='betta=';
        label8.caption:='gamma=';
end;
```

Задания для самостоятельного выполнения

Задание 1. Составить алгоритм и программу для решения задачи. В данном задании номер варианта определяется по номеру компьютера. Все приложения должны быть протестированы. Тестовые значения занесите в тетрадь.

1.1. Обменять значения двух переменных, используя третью переменную.

1.2. Составьте программу, которая определит размер сдачи после покупки в магазине некоторых товаров (наименование товара придумайте самостоятельно): товар 1 стоимостью a руб., товар 2 стоимостью b руб., товар 3 стоимостью c руб. Исходная сумма, выделенная на всю покупку d руб. В случае нехватки денег сдача получится отрицательной.

1.3. Даны стороны треугольника: a,b,c. Вычислить косинусы углов по теореме косинусов: $c^2=a^2+b^2-2a\cdot b\cdot cos(\alpha)$. Проверить условие существования треугольника по трем сторонам.

1.4. Даны координаты диагонали прямоугольника. Найти его площадь.

1.5. Треугольник задан координатами вершин $(x_1;y_1),(x_2;y_2),(x_3;y_3)$. Найти площадь треугольника (используя формулу Герона).

1.6. Дан цилиндр, с заданным радиусом основания r и высотой h. Найти объем цилиндра.

1.7. Дано трехзначное число. Найти в нем число тысяч, десятков и единиц.

1.8. Найти периметр и площадь прямоугольного треугольника, если даны длины его катетов a и b.

1.9. Даны два ненулевых числа. Найти их сумму, разность, произведение и частное.

1.10. Ввести двузначное число a. Поменять цифры числа местами.

1.11. Дана масса в килограммах. Найти число полных центнеров в ней.

1.12. Дано расстояние в сантиметрах. Найти число полных метров в нем.

1.13.* Пользователь вводит число. Проверить является ли оно симметричным.

1.14.* Пользователь купил билет с шестизначным номером. Программа должна позволить проверить является ли билет счастливым.

1.15.* Пользователь вводит три угла треугольника. Программа проверяет существование такого треугольника по трем значениям его углов и определяет его вид: остроугольный, тупоугольный, прямоугольный, равнобедренный, равносторонний.

Задание 2. Найти значение функции для выражений из таблицы 2.4. Номер варианта определяется по номеру компьютера.

Таблица 2.4 –Условия для задания самостоятельной работы

Вариант	Выражение	Вариант	Выражение
1	$y=\ln\left(e^{x}+\sqrt{1+e^{2x}}\right).$	2	$y=\ln\frac{x^{2}}{\sqrt{1-ax^{4}}}.$
3	$y=\frac{\cos\ln 7\cdot\sin^{2}7x}{7\cos 14x}.$	4	$y=\mathrm{tg}\lg\frac{1}{3}+\frac{1}{4}\frac{\sin^{2}4x}{\cos 8x}.$
5	$y=\ln\left(\sqrt{x}+\sqrt{x+1}\right).$	6	$y=\ln\frac{\sqrt{x^{2}+1}+x\sqrt{2}}{\sqrt{x^{2}+1}-x\sqrt{2}}.$
7	$y=\frac{\cos\sin 5\cdot\sin^{2}2x}{2\cos 4x}.$	8	$y=2\sqrt{x}-4\ln\left(2+\sqrt{x}\right).$
9	$y=\ln\left(x+\sqrt{a^{2}+x^{2}}\right).$	10	$y=\ln\left(bx+\sqrt{a^{2}+b^{2}x^{2}}\right).$
11	$y=\ln\frac{a^{2}+x^{2}}{a^{2}-x^{2}}.$	12	$y=\ln\frac{\ln x}{\sin\left(1/x\right)}.$

3 Контрольные вопросы

3.1 Перечислите основные простые типы данных.

3.2 Чем отличаются данные целого и вещественного типов?

3.3 Для чего в программе используется раздел *var*?

3.4 Приведите пример описания переменных.

3.5 Как обозначается и для чего используется оператор присваивания?

3.6 Для чего предназначены операции *div*, *mod*? Приведите примеры.

3.7 Как записать на языке Pascal выражения $\sqrt[5]{x+5}, log_2(x+5)$?

3.8 Перечислите компоненты, которые используются для ввода и вывода данных, для расположения надписи на форме.

3.9 Объясните назначение функций *StrToInt*, *StrToFloa*, *IntToStr*, *FloatToStr*?

3.10 Поясните назначение каждой строки процедур в упражнениях 1-3.

ЛАБОРАТОРНАЯ РАБОТА № 3
ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ РАЗВЕТЛЯЮЩЕЙСЯ СТРУКТУРЫ

Цель работы: научиться использовать в приложении структуру ветвление.

1 Теоретическое введение

Ветвление (развилка) – 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 и логических операторов, таблица 3.1.

Таблица 3.1 - Операции отношения

Оператор	Действие	Оператор	Действие
>	больше	<>	не равно
<	меньше	>=	больше или равно
=	равно	<=	меньше или равно

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.

Таблица 3.2 - Логические операции

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

Ниже приведены примеры задания условий с помощью операций отношений и логических операторов:

- для x∈(7;) условие можно задать так: *x>7, not(x<=7);*
- для x∈(-4; 10]можно задать так:*(x>-4) and (x<=10), not((x<=-4) or(x>10));*
- дляx∈ (- ; -5)∪[8; – *(x<-5) or (x>=8), not((x>=-5) and (x<8)).*

Условный оператор изменяет естественный порядок выполнения операторов программы. На рисунке 3.1 показан алгоритм работы условного оператора.

Условие – это выражение или переменная логического типа. Если условие имеет значение *TRUE* (т.е. истинно), то выполняется оператор 1, если *FALSE* (т.е. ложно) – оператор 2.

Точка с запятой после *условия* и *оператора1,* т.е. перед служебным словом *ELSE,* не ставится.

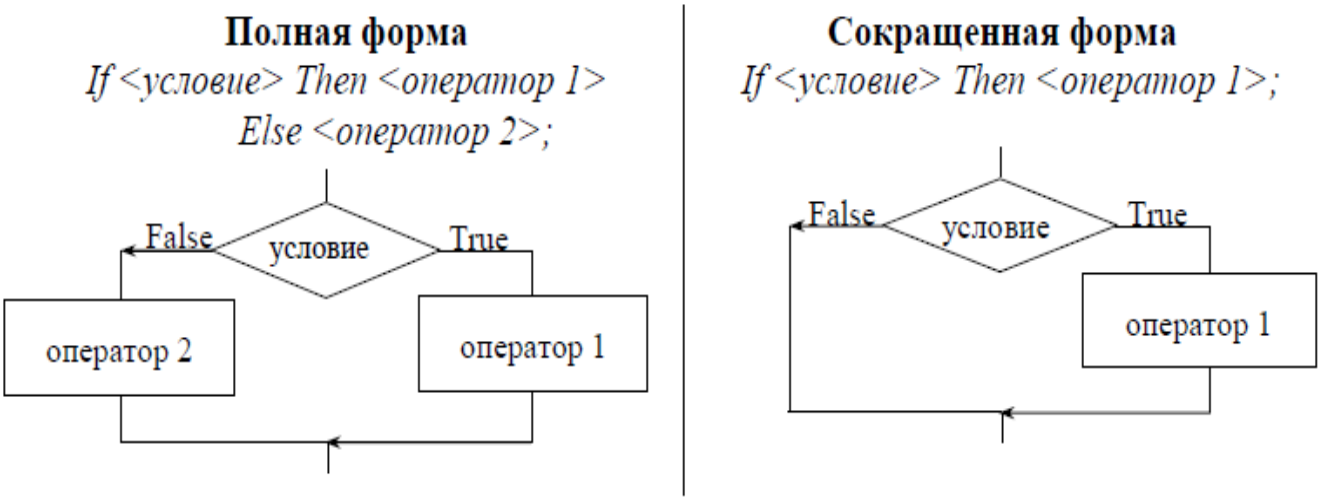


Рисунок 3.1 – Условный оператор If

Пример.

```
If x<>0
Theny:=k/x
ElseLabel1.Caption:='Деление на нуль запрещено';
```

Во втором случае если условие имеет значение *TRUE* (т.е. истинно), то выполняется оператор 1, если *FALSE* (т.е. ложно) - то выполняется оператор, следующий за оператором *IF*.

Пример.

```
Ifx>0
Theny:=sqrt(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, ограниченных словами *Begin* и *End*, разделенных между собой «;» (нельзя путать с составным оператором *Begin* и *End*, ограничивающими тело программы).

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

Составной оператор воспринимается как *единое целое* и обычно используется в качестве *составной части других операторов*, где требуется применение нескольких операторов *вместо одного*. Для облегчения чтения программы рекомендуется располагать *Begin* и *End* на одной позиции в программе.

Ниже показан пример реализации условного оператора в программе согласно условию указанному в блок-схеме на рисунке 3.2.

```
If x<>0
Then
Begin
y:=10/x;
Label1.Caption:=FloatToStr(y);
End
ElseLabel1.Caption:=’Ошибка’;
```

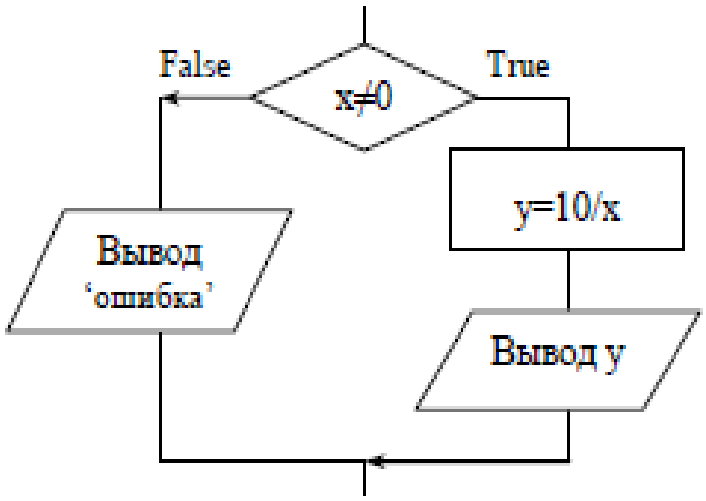


Рисунок 3.2 – Условный оператор и его реализация по блок-схеме

2 Выполнение работы

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

Определим используемые в задаче переменные и их тип:

- входные переменные: a, b, c (коэффициенты уравнения);
- промежуточные переменные: d(дискриминант);
- результат: x, x1, x2 (корни уравнения).

var a, b, c, d, x, x1, x2:real;

Построим алгоритм 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.

В зависимости от значения дискриминанта, рассмотрим в программе следующие решения.

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

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

Если $d < 0$, то выводится сообщение «Действительных корней нет, корни комплексные».

Если $d = 0$, то корень уравнения находится по формуле: $$X = \frac{-b}{2a}.$$

2.2 На основе вышеприведенного алгоритма постройте блок-схему.

2.3На форме расположите компоненты как показано на рисунке 3.3.

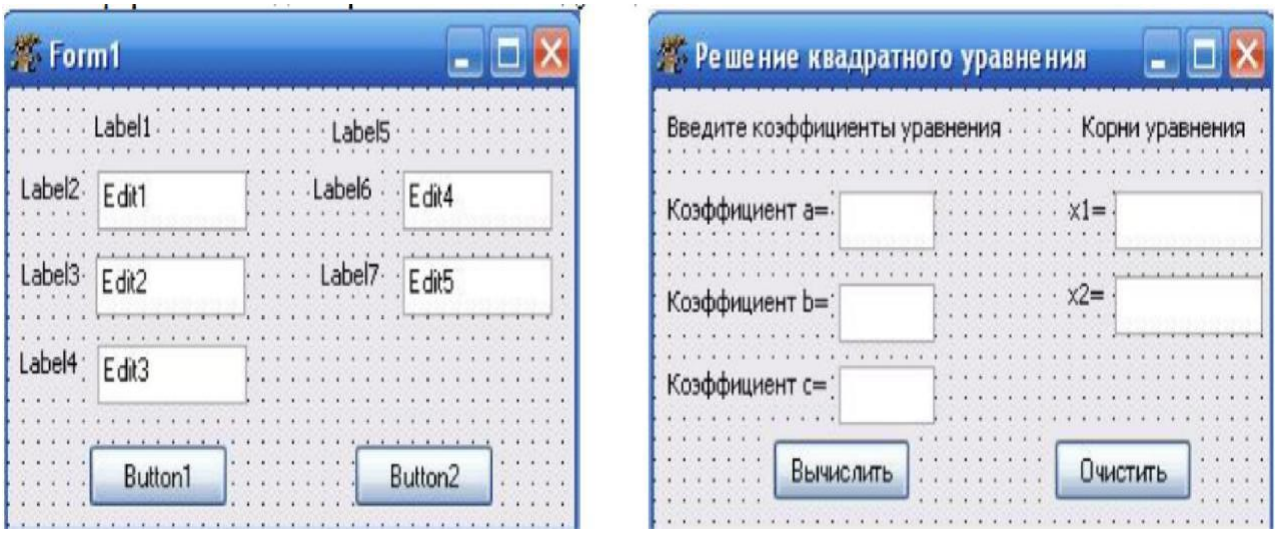


Рисунок 3.3 – Окно формы для решения задачи

2.4 Задайте свойства компонент в соответствии с таблицей 3.3.

Таблица 3.3 – Свойства компонент для упражнения 1

Название компонента	Свойство	Значение свойства
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

Процедура для вычисления значения функции будет иметь вид:

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,c,d,x,x1,x2:real;
begin
    a:=strtofloat(edit1.text);
    b:=strtofloat(edit2.text);
    c:=strtofloat(edit3.text);
    d:=sqr(b)-4*a*c;
    if d>0
    then
    begin
    x1:=(-b+sqrt(d))/(2*a);
    x2:=(-b-sqrt(d))/(2*a);
    edit4.text:=floattostr(x1);
    edit5.text:=floattostr(x2);
    end
    else
    if d=0
    then
    begin
    x:=-b/(2*a);
    edit4.text:=floattostr(x);
    label7. visible:=false;
    edit5.visible:=false;
```

```
    end
    else
    begin
    label7. visible:=false;
    edit4.text:='Решений нет';
    label7. visible:=false;
    edit5.visible:=false;
    end;
end;
```

2.5 Процедуру для кнопки «Очистить» написать самостоятельно.

2.6 Протестируйте программу подставив всевозможные значения a, b, c.

2.7 Разработайте приложение. Дано вещественное число x. Для функции, представленной на графике (рисунок 3.4), вычислить y=f(x).

Определим переменные задачи и их тип:

- входные переменные*: x: real;*
- результат: *y: real.*

Аналитически функцию, представленную на рисунке3.4 можно записать следующим образом:

$$y = \begin{cases} 4, x \le -2 \\ x^2, -2 < x < 1 \\ 1, x \ge 1 \end{cases}$$

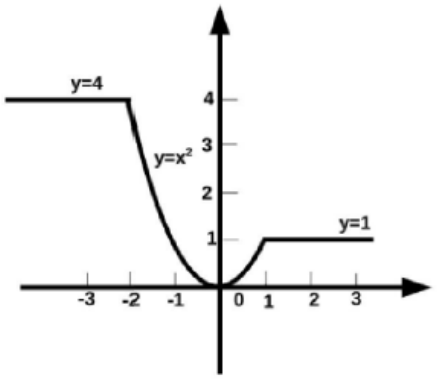


Рисунок 3.4 – График заданной функции

2.7 На форме необходимо расположить компоненты и задать им соответствующие свойства, в соответствии с рисунком 3.5.

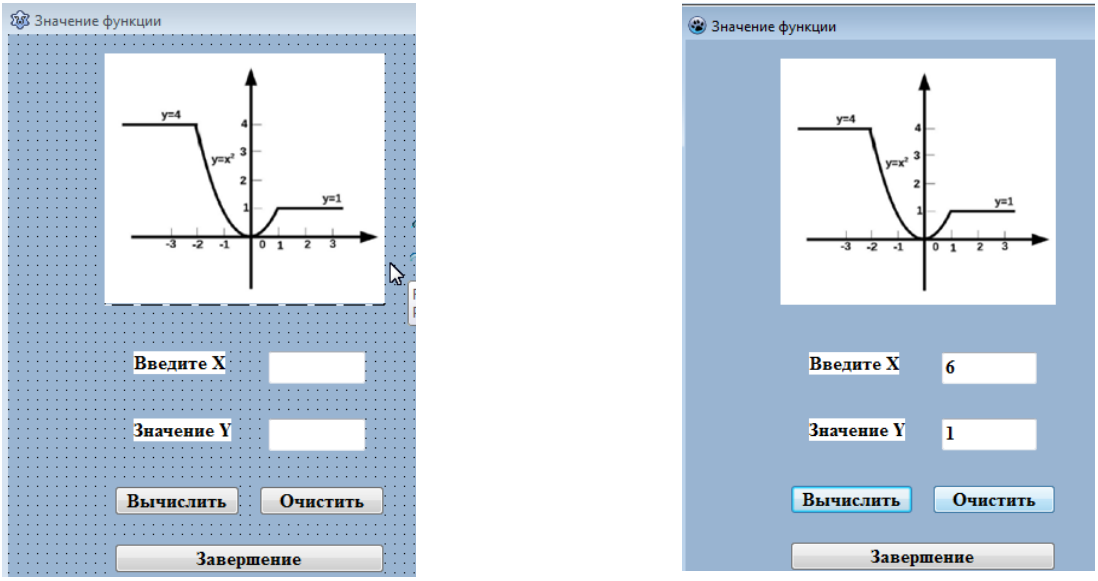


Рисунок 3.5 – Компоненты на форме

2.8 Набрать в инспекторе кода программу для вычисления значения функции.

Процедура для вычисления значения функции представлена ниже:

```
procedure TForm1.Button1Click(Sender: TObject);
      var x,y:real;
begin
     x:=strtofloat(edit1.text);
    if x<=-2
     then
        y:=4
    else
       if (x>1)
          then y:=1
       else y:=sqr(x);
     edit2.text:=floattostr(y);
end;
```

2.9 Процедуру для кнопки «Очистить» напишите самостоятельно.
На рисунке 3.6 показан алгоритм решения задачи.

2.10 Добавьте на форму компонент *Label* с заданной функцией, для которой вы находите значение

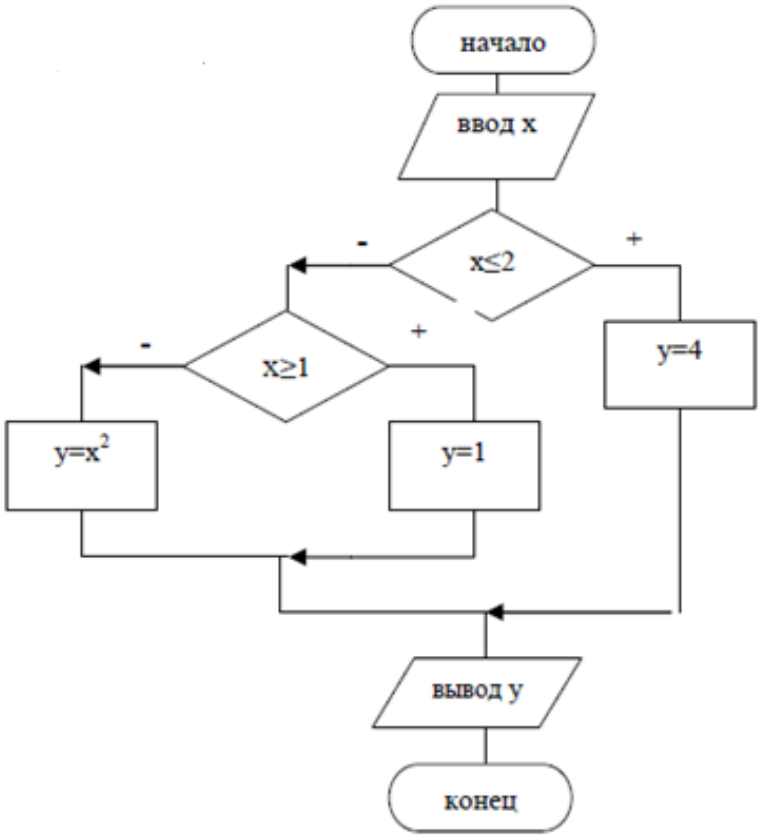


Рисунок 3.6 – Алгоритм решения задачи нахождения значения функции

Вариант реализации алгоритма и кода в приложении показан на рисунке 3.6'

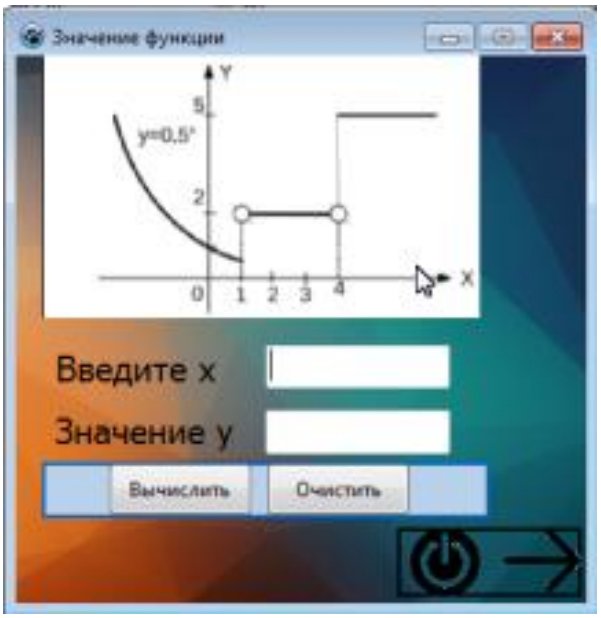


Рисунок 3.6' – Пример разработанного приложения для задания 2.7

2.10 Составить программу для вычисления квадрата и модуля числа.
В этом упражнении нужно научиться использовать компонент *CheckBox*

При реализации ветвления кроме известных компонент используется компонента *CheckBox* (флажок, выключатель)  . *CheckBox* позволяет пользователю выбрать/отменить определенную опцию.

Состояние флажка задается с использованием свойства *State*.

CheckBox может находиться в общем случае в трех состояниях:

– *флажок выбран*, в окне компоненты стоит галочка. Этому состоянию соответствует значение *State=cbChecked*;

– *флажок не выбран*, в окне компоненты галочка отсутствует. Этому состоянию соответствует значение *State=cbUnchecked*;

– *флажок выбран*, в окне компоненты стоит галочка. Само окно и галочка серого цвета. Этому состоянию соответствует значение *State=cbGrayed*. Это промежуточное состояние. Свойство *AllowGrayed=[true,false]* включает (выключает) состояние *cbGrayed*.

Для вычисления квадрата и модуля числа определим переменные и их тип:

– входные данные: *x: real;*

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

Для решения задачи расположите компоненты на форме как показано на рисунке 3.7.

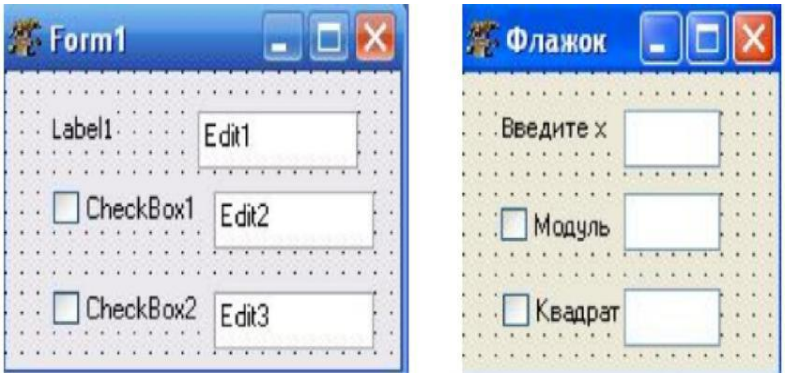


Рисунок 3.7 – Компоненты для 2.10

Компонентам задайте свойства в соответствии с таблицей 3.4.

Таблица 3.4 – Свойства компонент для упражнения 3

Название компонента	Свойство	Значение свойства
Label1	*Caption*	Введите *x*
CheckBox1	*Caption*	Модуль
CheckBox2	*Caption*	Квадрат
Edit1	*Text*	пусто
Edit2	*Text*	пусто
Edit3	*Text*	пусто

В программе будет происходить два события: щелчок мыши по компоненте «*Модуль*» и щелчок мыши по компоненте «*Квадрат*». При этом будет обрабатываться событие *OnChange* для компонентов *CheckBox1* и *CheckBox2*. Ниже показаны алгоритмы работы соответствующих процедур и сами процедуры.

Алгоритмы решения задания показан на рисунке 3.8.

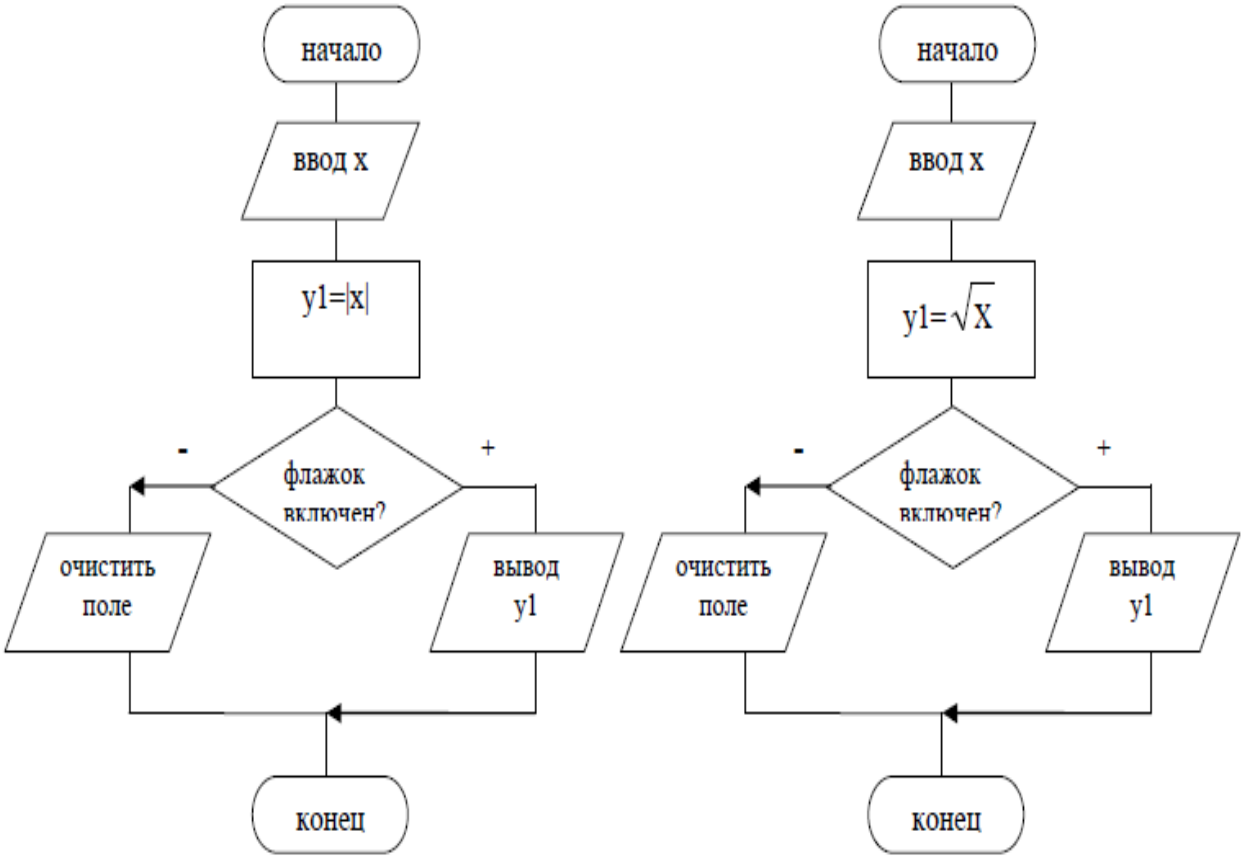


Рисунок 3.8 – Алгоритмы процедур для 2.10

2.11 Наберите тексты процедур и отладьте программу.

Процедуры для реализации решения задачи показаны ниже:

```
procedure TForm1.CheckBox1Change();
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();
 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;
```

Пример разработанного приложения по алгоритму для 2.10 показан на рисунках 3.9(а) и 3.9(б).

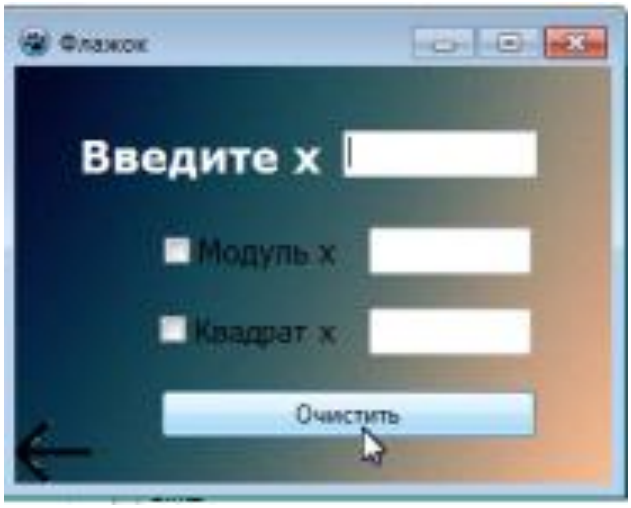


Рисунок 3.9(а) – Пример приложения для вычисления модуля и квадрата числа

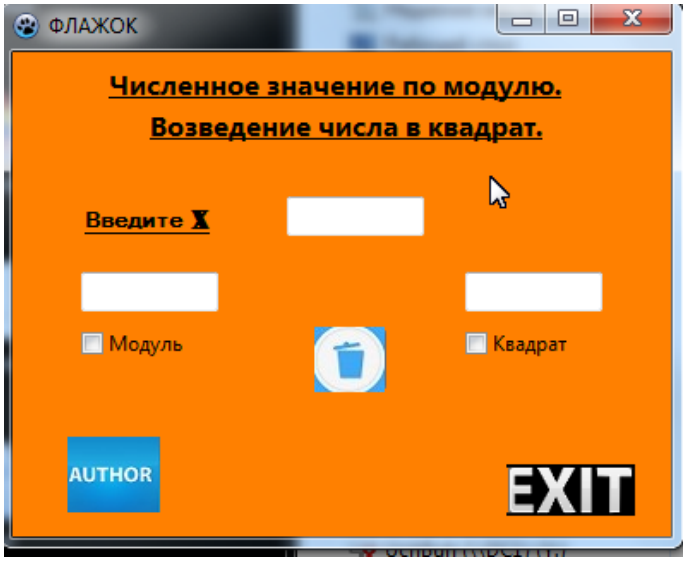


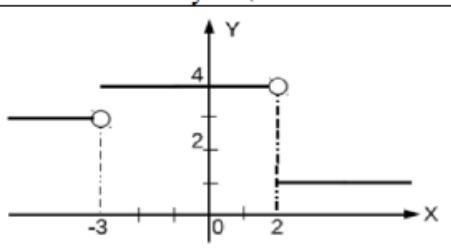
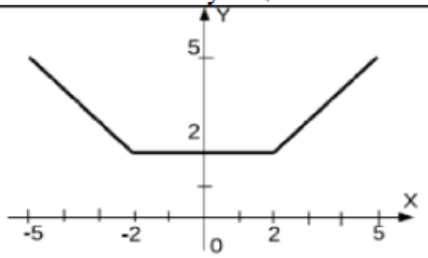
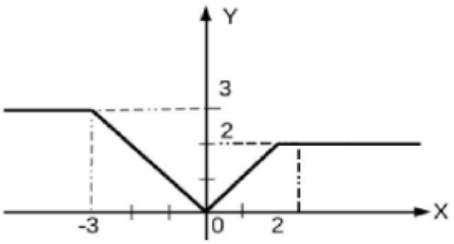
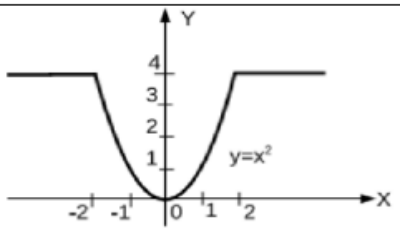
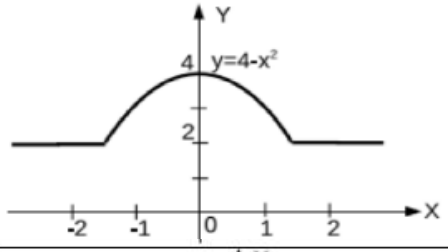
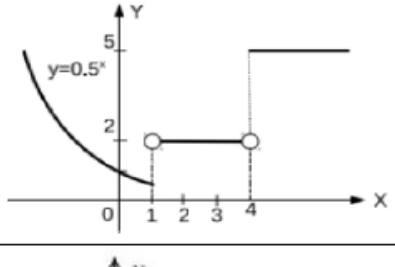
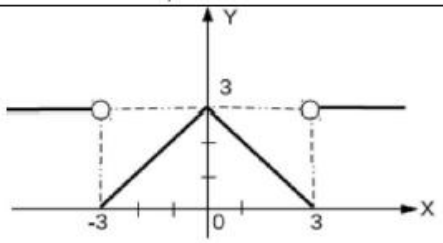
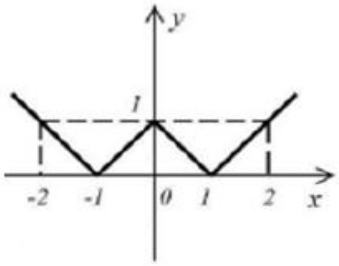
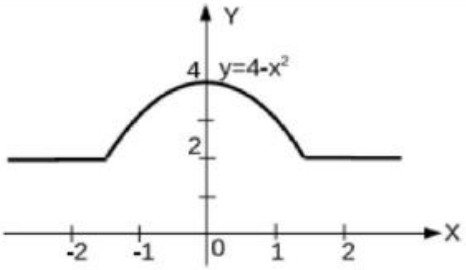
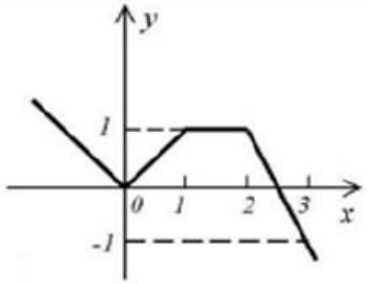
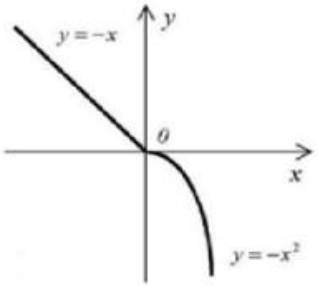
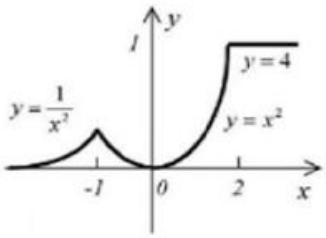
Рисунок 3.9(б) – Пример реализации интерфейса

Задания для самостоятельного выполнения

Задания выполняются по варианту. Вариант определяется по номеру компьютера.

Задание 1. Составить блок-схему и программу. Дано вещественное число *x*. Для функции, представленной графиком в таблице 3.5, вычислить *y=f(x)*.

Таблица 3.5 – Варианты для задания 1

Вариант	Функция	Вариант	Функция
1		2	
3		4	
5		6	
7		8	
9		10	
11		12	

Задание 2. Составить блок-схему и программу для задачи, соответствующей вашему варианту, таблица 3.6.

Таблица 3.6 - Варианты для задания 2

<table>
<tr><th>Вариант</th><th>Текст задачи</th></tr>
<tr><td>1</td><td>Определить стоимость набора конфет в зависимости от введенного названия и веса.
<table>
<tr><th>Наименование</th><th>Цена за 1 кг,</th><th>Наименование</th><th>Цена за 1 кг, руб.</th></tr>
<tr><td>Красная шапочка</td><td>350</td><td>Княжеские</td><td>240</td></tr>
<tr><td>Мишка на севере</td><td>380</td><td>Медуница</td><td>270</td></tr>
</table></td></tr>
<tr><td>2</td><td>Составьте программу для пожарного датчика в помещении, который анализирует температуру окружающей среды. Если, температура (вводиться с клавиатуры) в комнате превысила 60°С, то программа выводит сообщение 'Пожароопасная ситуация' и кнопка начинает мигать.</td></tr>
<tr><td>3</td><td>Из трех вводимых целых чисел выводится заключение о том, какое число больше и на сколько. находит сумму этих чисел.</td></tr>
<tr><td>4</td><td>Постройте модель, определяющую делится ли нацело введенное вами трехзначное число на сумму своих цифр и определяет его четность.</td></tr>
<tr><td>5</td><td>Товар расфасован в два пакета. Вес первого — m кг, второго — n кг. Вывести сообщение какой пакет тяжелее — первый или второй. Программа выводит размер перевеса и сумму, которую нужно заплатить за перевес.</td></tr>
<tr><td>6</td><td>Проверить, что введенное пользователем целое число является четным, найти его квадрат и квадратный корень из этого числа.</td></tr>
<tr><td>7</td><td>Проверить, верно ли утверждение, что введенное целое число делится без остатка на 3 и найти произведение его цифр.</td></tr>
<tr><td>8</td><td>Постройте модель для анализа возраста человека и отнесите его к одной из четырех групп: дошкольник, учащийся, работник, пенсионер. Возраст вводится с клавиатуры. Программа выводит сообщение сколько лет осталось до пенсии, тем кто не пенсионер.</td></tr>
<tr><td>9</td><td>Определить, входит ли введенная вами цифра в десятичную запись введенного трехзначного числа. Если входит, то программа находит квадрат числа иначе произведение этой цифры на число.</td></tr>
<tr><td>10</td><td>К финалу конкурса лучшего по профессии были допущены трое: Иванов, Петров, Сидоров. Соревнования проходили в три тура. Иванов в первом туре набрал m_l баллов, во втором — n_l, в третьем — p_1. Петров — соответственно -m_2, n_2, p_2; Сидоров — m_3, n_3, p_3. Определить, сколько баллов набрал победитель.</td></tr>
<tr><td>11</td><td>Постройте модель, реализующую фрагмент применения компьютера в интернет-магазине. Компьютер запрашивает количество покупаемого товара (например, трех видов), сравнивает с количеством товара на складе и выдает сообщение о возможности покупки такого количества товара. Если есть возможность продать такое количество товара, печатает на экране "спасибо за покупку"; если товара недостаточно, то печатает об этом сообщение.</td></tr>
<tr><td>12</td><td>Постройте модель, реализующую применение компьютера в интернет-магазине. Компьютер запрашивает количество покупаемого товара (например, 3 видов), согласно количеству, определяет стоимость покупки. Затем просит внести сумму денег, анализирует внесенную покупателем сумму; если сдачи не требуется, печатает на экране сумму кэш бека; если денег внесено больше, то печатает "возьмите сдачу" и указывает сумму сдачи; если денег недостаточно выводит сообщение.</td></tr>
</table>

3 Контрольные вопросы

3.1 Что представляет собой ветвление как алгоритмическая структура?

3.2 Перечислите операции отношения и логические операции. Приведите примеры их использования в программе.

3.3 Перечислите действия, реализуемые при выполнении условного оператора в программе.

3.4 Какие виды условного оператора *IF* существуют?

3.5 В каких случаях применяется сокращенная форма оператора *IF*?

3.6 В каких случаях применяется полная форма оператора *IF*?

3.7 Как организовать разветвление вычислений: а) на две ветви; б) на три ветви?

3.8 Для чего используется составной оператор? Приведите пример.

3.9 Как в программе можно использовать компонент *CheckBox*?

ЛАБОРАТОРНАЯ РАБОТА № 4
ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ РАЗВЕТЛЯЮЩЕЙСЯ СТРУКТУРЫ. ОПЕРАТОР МНОЖЕСТВЕННОГО ВЫБОРА

Цель работы: научиться использовать оператор множественного выбора при решении задач.

1 Теоретическое введение

1.1 Оператор выбора

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

Конструкция оператора выбора показана ниже:

Case*s *of **{**выбор**}**
c_1*:оператор1;* {при условии 1: выполняется действие 1}
c_2*:оператор2;* {при условии 2: выполняется действие 2}
…
c_n*:оператор n* {при условии *n*: выполняется действие *n*}
Else {иначе}
оператор;{выполнится действие *n+1*}
End*;*

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

Блок-схема данного оператора показана на рисунке 4.1. Если в наборе несколько значений, то они разделяются между собой «,». Можно указывать диапазоны значений, между которыми ставятся «..». Между набором значений с и соответствующим ему оператором должно стоять «:». Значения *c1 … cn* не должны повторяться.

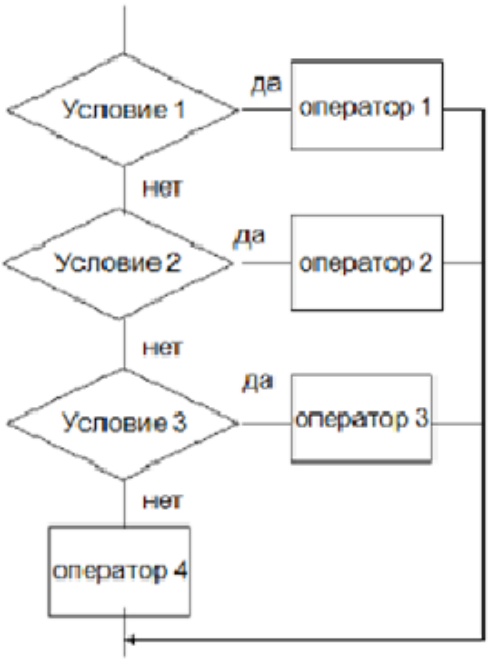


Рисунок 4.1 – Блок-схема оператора Case

Приведем пример использования оператора выбора. Дан фрагмент программы на алгоритмическом языке в таблице 4.1. Необходимо:

а) записать данный фрагмент, используя оператор *case*;

б) определить значения переменных *p* и *d* после выполнения алгоритма.

Таблица 4.1 - Запись задачи на алгоритмическом языке и в программе

Алгоритмический язык	На языке программирования
k:=10; p:=mod(k,11); *выбор* *при p=7: d:=sqr(k)* *при p>7: d:=sqr(2*k)* *при p<=5: d:=0* *иначе d:=10* *все*	*k:=10; p:=k mod 11;* *case p of* *7: d:=sqr(k);* *8,9,10: d:=sqr(2*k);* *0..5: d:=0* *elsed:=10;* *end;*

После выполнения алгоритма переменные *p* и *d* будут равны: *p*=10, *d*=400. Для работы с оператором выбора в *Lazarus* служат несколько компонент. Рассмотрим их.

1.2 Компонент *RadioGroup*

Компонент представляет собой комбинацию *GroupBox* с набором *RadioButton*, т. е. контейнер для размещения зависимых переключателей класса *TRadioButton*,

каждый переключатель помещается в специальный список *Items* и доступен по индексу. В таблице 4.2 показаны свойства этого компонента.

Таблица 4.2 - Основные свойства компонента *RadioGroup*

Свойство	Значение свойства
Caption	Заголовок
Colums	Количество столбцов переключателя
ItemIndex	Индекс выбранного переключателя (нумерация индексов начинается с нуля)
Items	Список строк с заголовками элементов

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

Особенности компонента *RadioGroup*:

- щелчок мыши выделяет строку в виде радиокнопки с точкой;
- 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.

1.3 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. В таблице 4.3 показаны свойства этого компонента.

Таблица 4.3 - Основные свойства компонента *ListBox*

Свойство	Значение свойства
ItemIndex	Индекс выбранной строки (нумерация индексов начинается с нуля)
Items	Список строк с заголовками элементов

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

Особенности компонента *ListBox*:

- щелчок мыши выделяет всю строку;
- выбор строки снимает выделение других строк;

– редактирование набора строк невозможно.

1.4 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 1020 строк. Текст хранится в свойстве *Lines* класса *TStrings*. В целом компонент представляет собой пронумерованный набор строк. В таблице 4.4 показаны свойства, а в таблице 4.5 методы этого компонента.

Таблица 4.4 - Основные свойства компонента Memo

Свойство	Значение свойства
Lines	Содержит строки текста (нумерация строк начинается с 0)
ScrollBar	Определяет наличие полос прокруток (*ssNone*, *ssHorizontal*, *ssVertical*, *ssBoth*)
MaxLength	Определяет максимальную длину текстовой строки. Если имеет значение 0,длина строки не ограничена
Text	Содержит весь текст

Таблица 4.5 - Основные методы компонента *Memo*

Свойство	Значение метода
Add(s)	Добавляет строку в набор данных последней и возвращает ее индекс
Clear	Очищает набор данных.
Delete(n)	Удаляет строку с индексом *n.*
Insert(n; s)	Вставляет строку *s* в набор с индексом *n*.
String[n]	Получить текст в строке с номером *n*
Count	Получить количество строк в *Memo*

2 Выполнение работы

2.1 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 расположите компоненты на форме в соответствии с рисунком 4.2.

Рисунок 4.2 – Интерфейс приложения для здания 1

2.2 Определим свойства для выбранных компонент.
Свойства выбранных компонент:

- *Form1 – Caption – Калькулятор*
- *RadioGroup1 – Name – RG1*
- *Caption – Операция*
- *Items – сумма*
  разность
  произведение
  частное
- *Label1 – Caption – первоечисло*
- *Label2 – Caption – второоечисло*
- *Label3 – Caption – Результат*
- *Edit1 – Text – пусто*
- *Edit2 – Text – пусто*
- *Edit3 – Text – пусто*
- *Button1 – Caption – Вычислить*
- *Button2 – Caption – Очистить*

2.3 Создадим действие для кнопки «Вычислить».
Процедура для кнопки 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
thenshowmessage('На нуль делить нельзя!')
else c:=a/b;
end;
edit3.text:=floattostr(c);
end;
```

2.4 Самостоятельно записать процедуру для кнопки «Очистить». И сохраните программу в своей папке для лабораторной работы 4, задания 1.

2.5 Написать в тетради блок-схему для решения задачи.

2.6 Имеется список студенческих групп, преподавателей и работников. Для выбраннойгруппы вывести день недели, в который группа проходит медосмотр: «БПИ» – понедельник; «БТМО» – в любой день, кроме субботы; «БМН» – четверг; «БЭК» – четверг; «БМТ» – пятница; «БЭЭ» – среда; «ППС» – вторник; «ИТР» – пятница, «БТТ» - среда. Вывод дня недели осуществить в многострочное окно ввода-вывода.

Определим переменные задачи:

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

n: integer

Для решения задачи расположим компоненты на форме в соответствии с рисунком 4.3.

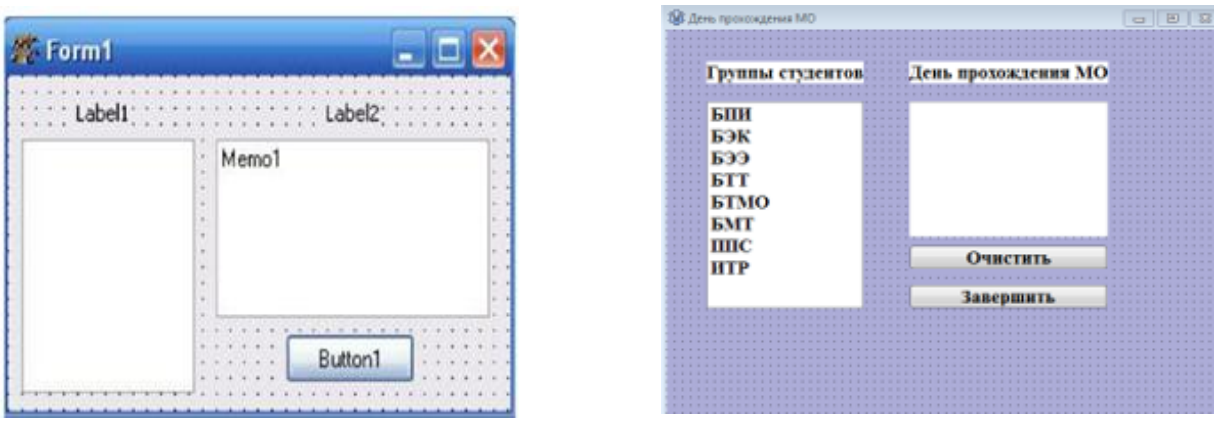


Рисунок 4.3 – Интерфейс приложения для 2.6

Выберем свойства для компонент формы:

ListBox11 – Items – БПИ
БЭК
БТМО
БЭЭ
БМТ
БМН
БТТ
ППС
ИТР

Label1 – Caption – Группы студентов
Label2 – Caption – День прохождения МО
Memo1 – Lines – пусто
Button1 – Caption – Очистить
Form1 – Caption – График прохождения медосмотра

В приложении осуществляется выбор одной из строк с названием студенческой группы, поэтому будет обрабатываться событие *OnClick* для компонента *ListBox1* и процедура-обработчик данного события будет иметь вид:

```
procedure TForm1.ListBox1.Click (Sender: TObject);
var n:integer;
begin
  n:= ListBox1.ItemIndex;
  case n of
       0:Memo1.lines.add('понедельник');
       1:Memo1.lines.add('ежедневно, кромесубботы');
       2:Memo1.lines.add('четверг');
       3:Memo1.lines.add('четверг');
       4:Memo1.lines.add('пятница');
       5:Memo1.lines.add('среда');
       6:Memo1.lines.add('вторник');
       7:Memo1.lines.add('пятница');
       8:Memo1.lines.add('среда');
   end;
end;
```

2.7 Самостоятельно записать процедуру для кнопки «Очистить». Сохраните программу в своей папке для лабораторной работы 4, задания 2.

2.8 Зарисовать в тетради блок-схему для решения данной задачи.

2.9 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. Создать программу, которая позволяет вводить номер дня недели и определить кинотеатр, работающий в этот день. Затем по выбору из списка фильмов, идущих в кинотеатре, щелчком мыши выбирается фильм, и программа выводит время киносеанса.

Определим переменные задачи:

промежуточная переменная: n (индекс выбранной строки)

n: integer;

Для решения задачи расположим компоненты на форме, как показано на рисунке 4.4.

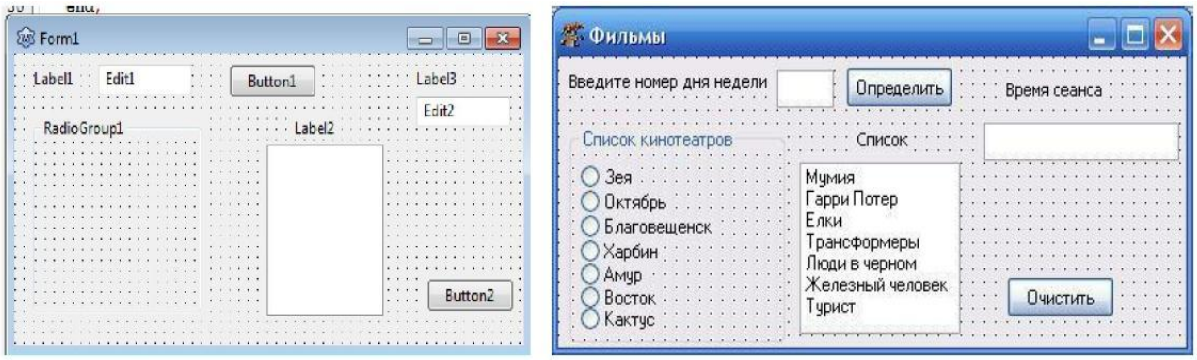


Рисунок 4.4 – Интерфейс приложения

Свойства выбранных компонент:

– *Form1 – Caption* – Фильмы

– *ListBox11 – Items* Мумия

Гарри Потер

Елки

Трансформеры

Люди в черном

Железный человек

Турист

– *Label1 – Caption* – Введите номер дня недели

– *Label2 – Caption* – Список

– *Label3 – Caption* – Время сеанса

– *Button1 – Caption* – Определить

– *RadioGroup1 – Items* – Зея

Октябрь

Благовещенск

Харбин
Амур
Восток
Кактус

- *Button2 – Caption – Очистить*
- *Edit1 – Text – пусто*
- *Edit2 – Text – пусто*
- *Memo1 – Lines – пусто*

Процедура определения работающего кинотеатра по номеру дня недели будет обрабатывать событие *OnClick* для кнопки *Button1u* будет иметь вид:

```
procedure TForm1.Button1Click(Sender: TObject);
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
  elseshowmessage('Введите номер дня недели от 1до 7');
  end;
end;
```

Процедура определения времени сеанса будет обрабатывать событие *OnClick* для списка *ListBox1* и будет иметь вид:

```
procedure TForm1.ListBox1Click(Sender: TObject);
begin
    0: case ListBox1.ItemIndex of
    1: edit2.text:='12-00, 16-00, 20-00';
    2: edit2.text:='12-30, 16-30, 20-30';
    3: edit2.text:='11-00, 15-00, 21-00';
    4: edit2.text:='11-30, 15-30, 21-30';
    5: edit2.text:='12-00, 15-00, 22-00';
```

```
        6: edit2.text:='11-00, 17-00, 21-00';
        edit2.text:='12-00, 15-00, 21-30';
      end;
end;
```

2.10 Самостоятельно записать процедуру для кнопки «Очистить».

2.11 Написать в тетради блок-схему для каждой из процедур.

Задания для самостоятельного выполнения

Задание 1. Составить блок-схему и программу для задачи, соответствующей вашему варианту по списку студенческой группы, таблица 4.5.

Таблица 4.5 Варианты для задания 1

Вариант	Текст задачи
1	2
1	Составить программу, которая при выборе пользователем названия фигуры или ее изображения (треугольник, ромб, прямоугольник, параллелограмм, конус, шар, цилиндр) выводит формулу для вычисления площади или объема этой фигуры.
2	Имеется расписание работы кабинета: понедельник 8^{00}-10^{00}, вторник 10^{00}-13^{00}, среда 9^{00}-12^{00}, четверг 8^{00}-10^{00}, пятница 8^{00}-12^{00}, суббота 9^{00}-11^{00}, воскресенье выходной. Программа выполняет следующее: для указанного дня недели выдает информацию о часах работы кабинета.
3	Составить программу, которая при вводе возраста ребенка определяет его как младший ясельный, младшая группа, средняя, старшая, подготовительная группа детского сада, начальные классы, средний школьный, подросток, юношество и выдает сообщение об особенностях каждого возраста.
4	В окне приложения расположены элементы круга - радиус, диаметр, длина окружности. Составить программу, которая по выбранному элементу, запрашивает его значение и вычисляет площадь круга.
5	Составить программу, которая запрашивает размеры помещения, размеры рулона обоев и рассчитывает расход обоев при заданных параметрах.
6	Пользователь вводит числа от 0 до 20. Составить программу, которая выделяет в списке соответствующее ей название на английском языке.
7	Программа запрашивает МРОТ и предлагает пользователю выбрать район проживания (центральный, уральский, дальневосточный, северный). В зависимости от выбора района начисляет процентную надбавку (0 %, 12 %, 50 %, 100 %) и выводит размер суммы в рублях за месяц.
8	Составить программу расчета стипендии. Пользователь вводит количество студентов и в списке с радиокнопками выбирает категорию сдачи сессии: все пятерки (X руб.); одна четверка и остальные пятерки (У руб.); все четверки (Z руб.). Программа выдает количество денег в рублях, расходуемых на стипендию студентов данной группы.

Продолжение таблицы 4.5

1	2
9	Составить программу расчета стоимости выбранных услуг в парикмахерской. Пользователь выбирает в списке вид услуги и видит ее стоимость. Учесть наличие мужского и женского зала.
10	Для суммы вклада в размере S рублей в списке радиокнопок выбирается вид вклада («Сохраняй» - 5,8 % годовых, «До востребования» - 2 % годовых, «Управляй» - 4,8 % годовых). Составить программу расчета суммы вклада через два года с учетом начисляемых процентов.
11	Определить остаток от деления целой части выражения $ln\left\lvert x^2+ab\right\rvert$ на 7. Значение x,a,b вводятся пользователем. В зависимости от полученного результата выделить в списке день недели с соответствующим номером.
12	Создайте калькулятор который при вводе аргумента функции рассчитывает логарифм, синус, косинус, тангенс числа.

3 Контрольные вопросы

3.1 Когда используется оператор множественного выбора?
3.2 Каков формат записи оператора множественного выбора?
3.3 Как множественный выбор можно представить в виде блок-схемы?
3.4 Когда используются компоненты *ListBox*, *Memo*, *RadioGroup*?

ДИАГНОСТИЧЕСКИЕ ЗАДАНИЯ ПО МОДУЛЮ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ» ДИСЦИПЛИНЫ «ИНФОРМАТИКА»

1 Проект Lazarus представляет собой свободную среду быстрой разработки программного обеспечения для компилятора:

1. Java;
2. FreePascal;
3. Си++;
4. VisualBasic.

2 Кроссплатформенное программное обеспечение – это программное обеспечение, работающее:

1. на одной аппаратной платформе и/или операционной системе;
2. на двух аппаратных платформах и/или операционных системах;
3. более чем на одной аппаратной платформе и/или операционной системе;
4. более чем на двух аппаратных платформах и/или операционных системах.

3 Процесс создания приложения можно разделить на следующие этапы:

1. написание программного кода, описание свойств элементов;
2. формирование окна программы;
3. отладка программы;
4. тестирование;
5. разработка справочной системы.

4 Укажите элементы, которые входят в главное окно проекта Lazarus:

1. меню;
2. панель инструментов;
3. палитра компонентов;
4. инспектор объектов;
5. окно редактора кода.

5 Визуальные и не визуальные компоненты программы находятся в окне:

1. Редактора кода;
2. Инспектора объектов;
3. Палитры компонентов
4. Проектировщика форм.

6 Инспектор объектов содержит следующие опции:

1. "Свойства" ("Properties");
2. "События" ("Events");
3. "Избранное" ("Favorites");
4. "Ограничения" ("Restricted");
5. "Процедуры" ("Procedures");
6. "Функции" ("Functions").

7 Общими для большинства компонентов Lazarusявляются свойства:

1. цвет;
2. имя;
3. размер;
4. интервал;
5. положение на экране.

8 Текст, который отображается на элементе или вблизи элемента соответствует свойству:

1. Style;
2. Caption;
3. Font;
4. Items;
5. Name.

9 Строка текста, которая представляет фактические данные, которые этот объект содержит, соответствует свойству:

1. Text;
2. Caption;
3. Font;
4. Size;
5. Name.

10 Выберите свойство, которое соответствует шрифту для написания текста, связанного с элементом управления:

1. Style;
2. Caption;
3. Font;
4. Items;
5. Name.

11 Выберите свойство, которое соответствует цвету, используемому для обрисовки элемента управления или текста, который в нем содержится:

1. Style;
2. Color;
3. Font;
4. Visible;
5. Canvas.

12 Действие при нажатии кнопки мыши, характеризует событие:

1. Click;
2. OnClick;
3. OnKeyPress;
4. OnEntry;
5. OnResize.

13 При нажатии на кнопку выполняется код, написанный в процедуре, вызываемой событием:

1. Click;
2. OnClick;
3. OnKeyPress;
4. OnEntry;
5. OnResize.

14 Программа на языке программирования пишется непосредственно в окне:

1. Редактора кода;
2. Инспектора объектов;
3. Палитры компонентов
4. Проектировщика форм.

15 Удобство редактирования текста программы реализовано тем, что …

1. все строки пронумерованы;
2. все служебные слова выделяются жирным шрифтом;
3. знаки препинания становятся красными;
4. строки с ошибками выделяются коричневым цветом;
5. комментарии могут заключаться в {} или (**), начинаются с // и выделяются синим;
6. строки с ошибками выделяются желтым цветом;
7. комментарии могут заключаться в [] или (""), начинаются с // и выделяются красным.

16 Определите части текста программы, которые работают независимо:
1. процедуры и функции;
2. константы и метки;
3. переменные и массивы;
4. строки и файлы;
5. процедуры и переменные.

17 Запуск приложения на выполнение, осуществляется нажатием:
1. клавиши F9;
2. кнопки на форме;
3. клавиши F5;
4. Ctrl+F9;
5. Alt+F5.

18 Окно приложения, на котором располагаются компоненты, называется:
1. формой;
2. модулем;
3. приложение;
4. редактором.

19 Укажите компонент, который служит для размещения текста на форме:
1. Label;
2. Button;
3. Edit;
4. Image;
5. GroupBox.

20 Компонент, который представляет собой текстовое окно, в которое можно вводить текст или числа во время выполнения программы размещения текста на форме называется …
1. Label;
2. Button;
3. Edit;
4. Image;
5. OpenDialog.

21 Компонент, который используется для организации выбора из нескольких взаимоисключающих возможностей (при этом допускается выбор только одного варианта из нескольких) называется …

1. GroupBox;
2. RadioButton;
3. ListBox;
4. ComboBox;
5. Timer.

22 Компонент, который может выполнять некоторый код через регулярные промежутки времени. Размещенный на форме, он остается невидимым во время работы приложения называется …

1. GroupBox;
2. RadioButton;
3. ListBox;
4. ComboBox;
5. Timer.

23 Выберите инструкцию, позволяющую реализовать множественный выбор:

1. If <условие> then
2. Case Селектор of
3. While <условие> do
4. For<логическое выражение><нач_знач>to<кон_знач>dobegin
5. Repeat // инструкции until

24 Для создания нового проекта нужно выполнить команду:

1. Проект - Создать проект–Приложение;
2. Проект - Файл –Создать;
3. Файл - Создатьпроект - выбрать Приложение.

25 Расположите процесс создания приложения в верном порядке:

1. Написание программного кода - описание свойств элементов, доступных только во время работы приложения, описание реакций на событие появления окна, нажатие на кнопку и других;
2. Формирование окна программы-расположение необходимых элементов, задание размеров, изменение свойств;
3. Отладка программы.

26 Инспектор объектов содержит …

1. 5 страниц
2. 4 страницы
3. 3 страницы

27 Для описания переменных используется служебное слово …

1. procedure
2. Var
3. Integer

28 Текстовая часть программы пишется в окне:

1. Инспектора объектов
2. Сообщения
3. Редакторакода
4. Окно формы

29 Кнопка это …

1. обработчик события OnClick;
2. элемент управления, предназначенный для запуска каких-то действий или команд;
3. любая клавиша.

30 Компонент TEdit служит для:

1. ввода исходных данных
2. чтения данных из полей ввода

31 Свойство, которое позволяет подобрать параметры шрифта для вывода надписи на кнопку:

1. Caption;
2. Name;
3. Font;

32 Квадратный корень из n определяется следующей стандартной функцией

1. Sqr(n)
2. Koren(n)
3. Sqrt(n)

33 Функция, возвращающая строковое представление значения выражения, указанного в качестве параметра функции.

1. TButton
2. FloatToStr
3. IntToStr

34 Функция, которая позволяет преобразовывать целочисленную переменную в ее строковое представление:

1. IntToStr
2. StrToInt
3. StrToFloat
4. FloatToStr

35 Одним из недостатков Lazarus является …

1. большой размер файла при компиляции.
2. отсутствие документации.

36 Вещественное число – это …

1. тип данных;
2. идентификатор;
3. служебное слово;
4. свойство переменных.

37 Функция, которая используется для конвертирования вещественной переменной в строковую …

1. StrToInt
2. StrToFloat
3. IntToStr
4. FloatToStr

38 Служебное слово, которое отвечает за подключение библиотек

1. Unit
2. uses
3. Type.
4. form

39 Математической функции x^2 в Lazarus соответствует функция:

1. abs(x);
2. sqrt(x);

3. sqr(x);
4. exp(x).

40 При нажатии на кнопку выполняется код, написанный в процедуре, вызываемой событием:

1. OnKeyPress;
2. Click;
3. OnClick;
4. OnEntry;
5. OnResize.

41 КомпонентеListBox соответствует пиктограмма:

1.  ;
2.  ;
3.  ;
4.  ;
5.  .

42 В каком разделе описывается константа:

1. const;
2. procedure;
3. var;
4. label;
5. function.

43 Язык высокого уровня - это

1. Реализация языка Ассемблера
2. Язык программирования, наиболее приближенный к человеческом языку
3. Машинный язык
4. Мнемоническое представление машинного языка

44 Функция, которая выводит на экран диалоговое окно, содержащее сообщение и поле ввода, устанавливает режим ожидания ввода текста пользователем или нажатия кнопки, а затем возвращает значение типа String, содержащее текст, введенный в поле:

1. ShowMessage;
2. OpenDialog;

3. InputBox;
4. MessageDlg;
5. ListBox.

45 Компонент, представляющий сочетание выпадающего списка и однострочного текстового поля, которое позволяет пользователю ввести значение вручную или выбрать из списка:

1. GroupBox;
2. RadioButton;
3. ListBox;
4. ComboBox;
5. Timer.

46 ShowMessage() осуществляет;

1. Вывод сообщения
2. Результат компилирования
3. Вызов окна сообщения ошибок

47 Свойство формы Hint отвечает за …

1. показ всплывающей подсказки (разрешает/запрещает)
2. состояние окна
3. высота формы
4. текст всплывающей подсказки

48 Компоненте Edit соответствует пиктограмма:

1.  ;
2.  ;
3.  ;
4.  ;
5.  .

49 В среде разработки Lazarus есть следующие виды переменных:

1. Глобальные и локальные,
2. Вещественные, целочисленные, строковые.
3. Глобальные и встроенные
4. Локальные и вещественные

50 Строка текста в элементе Label, который задает размер соответствует свойству:

1. text
2. caption
3. Font.
4. size
5. Name

51 Процедура, которая используется для вывода простых сообщений, не требующих ответа от пользователя:

1. MessageDlg;
2. ShowMessage;
3. InputBox;
4. OpenDialog;
5. ListBox.

52 КомпонентеButton соответствует пиктограмма:

1.  ;
2.  ;
3.  ;
4.  ;
5.  .

53 Компоненте RadioButton соответствует пиктограмма:

1.  ;
2.  ;
3.  ;
4.  ;
5.  .

54 Выберите правильный вариант объявления переменной в Lazarus

1. Int x=1;
2. Var a, b: integer;
3. Var 1a, 3b: real;
4. Var $a=1;

55 При создании новой формы к проекту добавляется следующая пара файлов:

1. *.LPI и *.LPR
2. *.PAS и *.LFM
3. *.DLL и *.TXT
4. *.RES и *.EXE

55 Идентификатором является...

1. последовательность латинских букв, цифр и знаков подчеркивания, которая начинается с буквы или символа подчеркивания и не содержит пробелов;
2. последовательность любых символов, которая начинается с буквы или символа подчеркивания и не содержит пробелов;
3. последовательность букв и цифр, которая начинается с буквы и не содержит пробелов.

56 Укажите правильно записанный идентификатор.

1. Center
2. @5S
3. 7Colors

57 Укажите правильную последовательность разделов процедур в Lazarus.

1. раздел описания переменных;
2. тело процедуры;
3. заголовок процедуры.

58 К простым относятся следующие типы данных:

1. вещественный;
2. строковый;
3. целый.

59 Из перечисленных типов данных целым является …

1. real;
2. integer;
3. word.

60 Выберите из предложенных структур процедур, верно, реализованную.

1. begin
```
procedure TForm1.Button1Click(Sender: TObject);
  x:=y+100;
  Label1.Caption:=IntToStr(x);
```

```
end;
```
2.
```
procedure TForm1.Button1Click(Sender: TObject);
    var x,y: integer;
    begin
      x:=y+100;
     Label1.Caption:=IntToStr(x);
end;
```
3.
```
procedure TForm1.Button1Click(Sender: TObject);
    begin
      var x,y: integer;
      x:=y+100;
      Label1.Caption:=IntToStr(x);
    end;
```

61 Результатом выполнения операции 5 Div 2 является число...

Запишите число: _____

62 Результатом выполнения операции 5 mod 2 является число...

Запишите число: _____

63 Укажите, что в списке можно рассматривать как идентификатор.

1. FIO2
2. _Rezult
3. 2Summa

64 Укажите верно записанный оператор:

1. if a:=b then b<0;
2. if b<0 then a=b;
3. if b<0 then a:=b.

65 Выберите верные результаты выполнения операций:

1. 4 div 3 = 1
2. 10 div 3 = 3
3. 11 mod 5 = 2

66 Результатом выполнения выражения (10+6)*3/2 будет число...

Запишите число: _____.

67 Введите выражение на Паскале с использованием стандартных арифметических функций, вычисляющего Tg5+Ln3.

Запишите ответ:_____.

68 Оператором называется...

1. предложение языка программирования, задающее полное описание некоторого действия, которое необходимо выполнить.
2. конструкция языка, задающая порядок выполнения действий над элементами данных.
3. совокупность величин и выражений, над которыми производится операция.

69 Выберите верные утверждения:

1. цикл While работает пока условие ложно;
2. цикл Repeat завершается, когда условие становится истинным;
3. цикл Repeat обязательно выполняется как минимум один раз.

70 Укажите ложное утверждение:

1. цикл While работает пока условие истинно
2. цикл Repeat завершается, когда условие становится ложным
3. цикл Repeat обязательно выполняется как минимум один раз

71 Укажите верные утверждения для цикла For:

1. цикл For - это цикл с предусловием;
2. цикл For -это цикл с заданным числом повторений;
3. шаг изменения параметра цикла всегда равен 1 или -1.

72 Алгоритмом называется...

1. конечный набор правил, позволяющий механически решать любую конкретную задачу из некоторого класса однотипных задач.
2. точное предписание, определяющее последовательность действий исполнителя, направленных на решение поставленной задачи.
3. программа для компьютера.

73 В результате работы процедуры на форме появится число ____:

```
procedure TForm1.Button1Click(Sender: TObject);
        var i,s: integer;
begin
i:=7;
s:=0;
```

```
if   i>6 Then s:=s+i;
Label1.Caption:=IntToStr(s);
end;
```

74 В результате работы процедуры на форме появится следующий результат ___

```
procedure TForm1.Button1Click(Sender: TObject);
        var i,s: integer;
begin
s:=0;
for i:=1 to 5 do  s:=s+i;
Label1.Caption:=IntToStr(s);
end;
```

75 Введите строку программы на Паскале для задания константы а, хранящей значение 3,45.

Запишите ответ:_____

76 Оперативная память предназначена для…

1. долговременного хранения программ и данных.
2. хранения исполняемых в данный момент программ и данных.
3. хранения арифметических и логических операции.

77 Для обозначения блока ДЕЙСТВИЯ в блок-схемах используется (введите нужное слово)…

Запишите ответ:_____

78 Для обозначения блока УСЛОВИЯ в блок-схемах используется (введите нужное слово)…

Запишите ответ:_____

79 Для обозначения блока ВВОДА в блок-схемах используется (введите нужное слово)…

Запишите ответ:_____

80 Алфавит языка Pascal включает в себя …

1. латинские буквы;
2. русские буквы;
3. десятичные цифры.

81 Русские буквы в программе…

1. используются для написания операторов;
2. заключаются в апострофы;
3. заключаются в круглые скобки.

82 Укажите, какие из нижеперечисленных правил относятся к правилам написания идентификаторов:

1. Идентификатор начинается с буквы или знака подчеркивания.
2. Идентификатор может содержать буквы, цифры и знак подчеркивания.
3. При написании идентификаторов используются только прописные буквы.

83 Константами называются…

1. данные, значения которых могут не изменяться в процессе работы программы.
2. данные, значения которых не изменяются в процессе работы программы.
3. данные, изменяющиеся в процессе работы программы.

84 Укажите истинность или ложность вариантов объявления константы:

1. const MYN=’ПетяИванов’
2. const Max=1000
3. N=86

85 Переменными называются…

1. величины, которые не изменяют свои значения в программе.
2. величины, всегда изменяющие свои значения в программе.
3. величины, которые могут изменять свои значения в программе.

86 Для организации повторения некоторого набора действий с шагом 1 используется оператор…

1. Repeat …Untile
2. For… To…Do
3. While …Do

87 Укажите верную последовательность этапов решения задач с помощью компьютера.

1. постановка задачи;
2. разработка алгоритма;
3. анализ и исследование задачи;
4. сопровождение программы;
5. программирование;

6. тестирование и отладка;
7. анализ результатов решения задачи.

88 Укажите файлы, которые включаются в состав вновь созданного пользовательского проекта Lazarus:

1. файл проекта (*.lpi)
2. файл модуля (*.pas)
3. текстовый файл (*.txt)

89 Верно ли утверждение «Компонент Memo не входит в главные составные части среды программирования Lazarus».

1. да;
2. нет.

90 Выберите то, что является характеристикой объекта в среде программирования Lazarus:

1. свойство;
2. палитра компонент;
3. метод.

91 Реакция на событие это...

1. действия объекта, что объект может делать.
2. описания действий, которые необходимо совершить при данном событии.
3. изменения в окружающей объект обстановке.

92 Для написания программы в Lazarus необходимо.... (определите последовательность):

1. задать реакцию на событие;
2. разместить необходимые компоненты на форме;
3. задать свойства выбранных компонент;
4. определить событие.

93 Укажите соответствия:

__ Для свойства Caption компоненту Label можно использовать для ...
__ Компоненту Edit можно использовать для ...
__ Компоненту Button можно использовать для ...
__ Компоненту Panel можно использовать для ...

из предложенных вариантов ответа:

1. ввода/вывода чисел и текста в программу.

2. размещения на ней сгруппированных компонент.
3. задания заголовка надписи, выводимой на форме.
4. задания реакции на событие.

94 Укажите соответствия для следующих понятий:
__ оператор присваивания
__ mod
__ в конце каждого оператора в программе
__ div
из предложенных вариантов ответа:
1. ;
2. целочисленное деление
3. :=
4. нахождение остатка от деления

95 Для обозначения целого типа данных в Lazarus используется имя ...
Запишите ответ:_____

96 В программе для описания переменных используется раздел...
Запишите ответ:_____

97 Для обозначения вещественного типа данных в Lazarus используется имя ...
Запишите ответ:_____

98 Для вывода окна сообщения используется...
1. компонента Label;
2. команда ShowMessage('текст');
3. компонента Edit.

99 Результат выполнения строки программы Panel1.Color:=clRed; будет следующим:
1. на компоненте Panel1появится надпись clRed
2. цвет компоненты Panel1 будет меняться на красный
3. на компоненте Panel1появится надпись Color

100 Компоненте Panel соответствует пиктограмма:
1. 

;
2. 
;

3.  ;
4.  ;
5.  .

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алексеев Е. Р. FreePascal и Lazarus: учебник по программированию. М.: ДМК-пресс, 2010. – 438 с.

2. Васильев П.П. TurboPascal в примерах и задачах: учебн.пособие. - М.: Финансы и статистика,2002.

3. Информатика. Базовый курс: учебник / С.В. Симонович. - 3-е изд. - СПб.: Питер, 2015

4. СухаревМ. TurboPascal: Учебник. - СПб.: Наука и техника, 2007

5. СухаревМ. TurboPascal 7.0. Теория и практика программирования: учебн.пособие. - СПб.: Наука и техника, 2007

АБДУЛВЕЛЕЕВА РАУЗА РАШИТОВНА

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ LAZARUS

Лабораторный практикум

для студентов направлений подготовки:

09.03.03 Прикладная информатика,

13.03.02 Электроэнергетика и электротехника,

15.03.02 Технологические машины и оборудование,

13.03.01 Теплоэнергетика и теплотехника,

22.03.02 Металлургия,

18.03.01 Химическая технология,

очной и заочной форм обучения

Подписано в печать 16.09.2020 г.		
Формат 60x90 1/16 Рег.№ 154	Печать цифровая Тираж 30 экз.	Уч.-изд.л. 4,38

ФГАОУ ВО
Национальный исследовательский технологический университет «МИСиС»
Новотроицкий филиал
462359, Оренбургская обл., г. Новотроицк, ул. Фрунзе, 8.
E-mail: nfmisis@yandex.ru
Контактныйтел. 8 (3537) 679729.