

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

А. А. Сорокин

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ
LAZARUS (FREE PASCAL)**
**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
(ЛАБОРАТОРНЫЙ ПРАКТИКУМ)**

Направление подготовки

210700.62 – Инфокоммуникационные технологии и системы связи
Профиль подготовки «Сети связи и системы коммутаций»

Бакалавриат



Ставрополь
2014

УДК 681.3.06(075.8)
ББК 32.973-018.1я73
С 65

Печатается по решению
редакционно-издательского совета
Северо-Кавказского федерального
университета

- Сорокин А. А.
С 65 **Объектно-ориентированное программирование. LAZARUS (Free Pascal): лабораторный практикум.** – Ставрополь: Изд-во СКФУ, 2014. – 216 с.

Лабораторный практикум по дисциплине «Объектно-ориентированное программирование» разработан в соответствии с учебной программой и ГОС ВПО и содержит рекомендации по выполнению лабораторных работ, а также варианты заданий и требования к оформлению отчетов.

Целью лабораторного практикума является формирование у студентов теоретических знаний и практических навыков разработки систем объектно-ориентированного программирования (LAZARUS: Free Pascal) с использованием методов визуального программирования.

Предназначен для студентов, обучающихся по направлению 210700.62 – Инфокоммуникационные технологии и системы связи, а также студентов, аспирантов и магистрантов соответствующих специальностей.

УДК 681.3.06(075.8)
ББК 32.973-018.1я73

Рецензенты:
д-р техн. наук, профессор *И. А. Калмыков*,
д-р экон. наук, профессор *А. С. Мараховский*

© ФГАОУ ВПО «Северо-Кавказский
Федеральный университет», 2014

ПРЕДИСЛОВИЕ

Lazarus – свободная среда разработки программного обеспечения для компилятора Free Pascal Compiler. Интегрированная среда разработки предоставляет возможность кроссплатформенной разработки приложений в Delphi-подобном окружении. На данный момент является единственным инструментом позволяющий достаточно несложно переносить Delphi-программы с графическим интерфейсом в различные операционные системы: Linux, FreeBSD, Mac OS X, Microsoft Windows.

Основан на библиотеке визуальных компонентов Lazarus Component Library (LCL). В настоящее время практически полностью поддерживает виджеты Win32, GTK1, GTK2, Carbon. В разработке находятся виджеты Qt и WinCE.

Функции:

- Поддерживает преобразование проектов Delphi.
- Реализован основной набор элементов управления.
- Редактор форм и инспектор объектов максимально приближены к Delphi.
- Встроенный отладчик.
- Простой переход для DELPHI-программистов благодаря близости LVL к VCL.
- Полностью юникодированный (UTF-8) интерфейс и редактор и поэтому отсутствие проблем с портированием кода, содержащего национальные символы.
- Мощный редактор кода, включающий систему подсказок, гипертекстовую навигацию по исходным текстам, автозавершение кода и рефакторинг.
- Поддержка множества типов синтаксиса Pascal: Object Pascal, Turbo Pascal, Mac Pascal, Delphi (поддерживаются со стороны компилятора).
- Имеет собственный формат управления пакетами.
- Поддерживаемые для компиляции ОС: Linux, Microsoft Windows (Win32, Win64), MacOS X, FreeBSD, WinCE, OS/2.

Базовыми дисциплинами для лабораторного практикума являются: информатика; программирование; программирование на языках высокого уровня.

Лабораторные работы по дисциплине «Объектно-ориентированное программирование» позволяют студенту научиться решать актуальные на сегодняшний день задачи связанные с проектированием приложений в среде Lazarus, а дисциплина занимает важное место в учебном процессе, поскольку она способствует получению навыков и знаний, необходимых студенту при изучении других дисциплин. При выборе среды объектно-ориентированного программирования учитывалось то, что в качестве базового языка к дисциплине «Программирование» был избран FreePascal. Программирование в Lazarus (FreePascal) является логическим следствием изучения языков в высокого уровня (Pascal, Basic), способствует закреплению знаний основных алгоритмов и выходу на качественно новый виток развития. При этом хочется отметить, что на первых порах при изучении ООП студенты испытывают трудности перехода к новой концепции программирования. Поначалу требуется забыть привычные представления о программировании.

Каждая лабораторная работа предваряется краткой теорией, необходимой для выполнения задания. Более подробно теоретический материал рассматривается на лекциях.

ЛАБОРАТОРНЫЕ РАБОТЫ

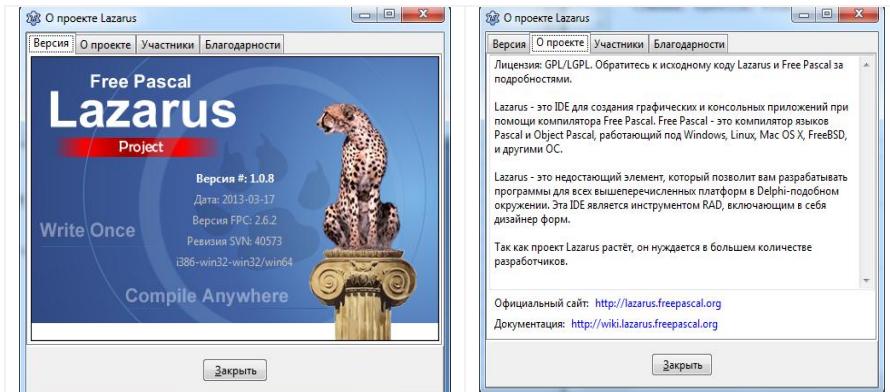
1. ЗНАКОМСТВО С ИНТЕГРИРОВАННОЙ СРЕДОЙ LAZARUS

Цель: знакомство с интегрированной средой разработки программного обеспечения Lazarus. Создание нового проекта.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория



Пользовательский интерфейс Lazarus. Интегрированная среда разработки Lazarus представляет собой многооконную систему, вид (пользовательский интерфейс) которой может различаться в зависимости от настроек. После загрузки интерфейс Lazarus представлен на рис.1.1.

К основным окнам среды программирования Lazarus относятся:

1. главное окно;
2. окно Инспектора объектов;
3. окно Формы, или Конструктора формы;
4. окно Редактора кода;
5. окно Сообщений.

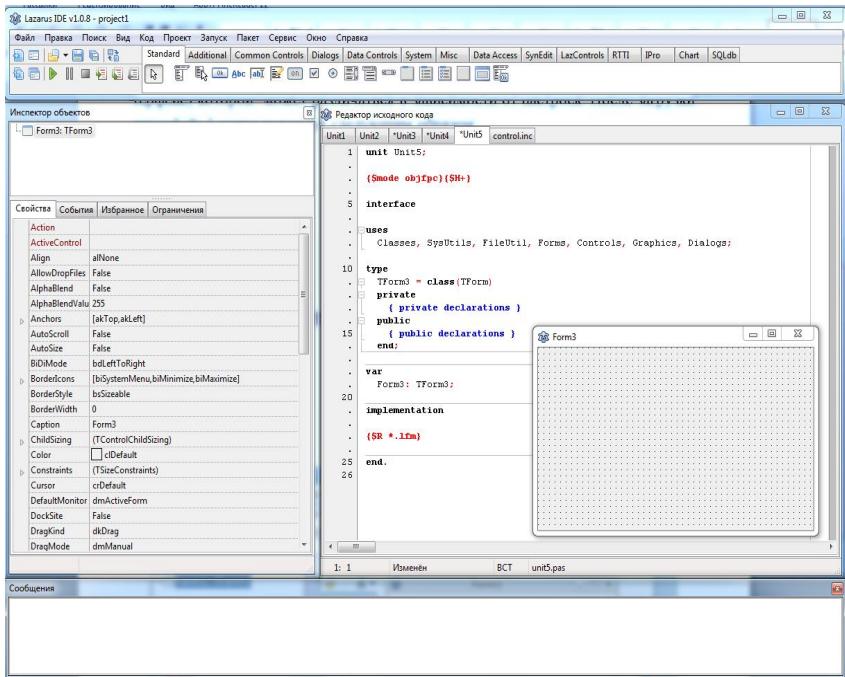


Рис. 1.1. Основные окна среды программирования Lazarus

Главное окно Lazarus содержит:

1. **главное меню;**
2. **панели инструментов;**
3. **палитру компонентов.**

Главное меню содержит обширный набор команд для доступа к функциям Lazarus.

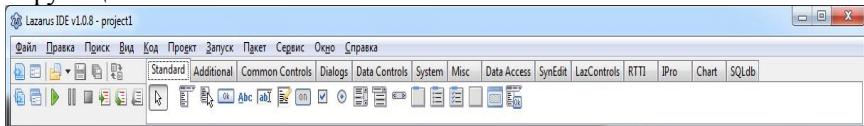


Рис. 1.2. Главное меню среды программирования Lazarus

Панели инструментов находятся под главным меню в левой части главного окна и содержат кнопки быстрого доступа к наиболее частым командам главного меню.

Палитра компонентов находится под главным меню в правой части главного окна и содержит множество компонентов, размещаемых в создаваемых формах. *Компоненты* являются своего рода строительными блоками, из которых конструируются формы приложения.

Окно формы (или *Конструктора формы*) первоначально находится в центре экрана и имеет заголовок *Form1*. В нем выполняется проектирование формы, путем размещения на форме различных компонентов.

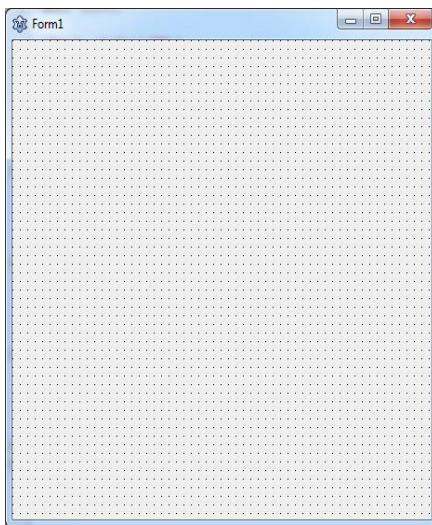


Рис. 1.3. Окно формы (или Конструктора формы)

Окно *Редактора кода* после запуска системы программирования находится под (или поверх) окна *Формы*. Редактор кода представляет собой обычный текстовый редактор, с помощью которого можно редактировать текст программы (*листинг*).

Окно *Инспектора объектов* расположено под *Главным окном* в левой части экрана и состоит из двух частей: в верхней части окна *Инспектора объектов* располагается список всех созданных объектов (форма и все компоненты, которые расположены на форме); в нижней части отображены свойства и события объектов для текущей формы или компонента. Вкладка *Свойства* отражает свойства выбранного компонента или формы, а вкладка *События* – процедуры, которые должны быть выполнены при возникновении указанного события.

The screenshot shows the Delphi IDE's code editor window. The title bar reads "Редактор исходного кода". The menu bar has items: Unit1, Unit2, *Unit3, *Unit4, *Unit5, control.inc. The code editor displays the following Pascal code:

```
1 unit Unit5;
.
.
.
5 interface
.
.
.
10 type
.   TForm3 = class(TForm)
.     private
.       { private declarations }
.     public
.       { public declarations }
15 end;
.
.
.
var
  Form3: TForm3;
20
implementation
.
.
.
(SR *.lfm)
.
25
end.
```

Рис. 1.4. Окно Редактора кода

Основная форма программы – содержит доступ ко всем остальным формам программы. Если закрыть подчиненное окно, то проект остается открытым. Если закрыть главную форму программы – закроется все приложение.

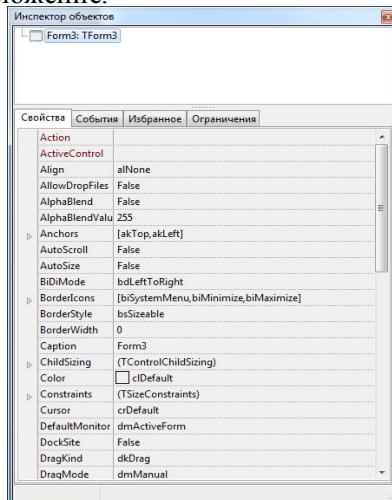


Рис. 1.5. Окно Инспектора объектов

Главное меню программы – имеет древовидную структуру и содержит доступ ко всем элементам программы от «Файл» до «Справка».

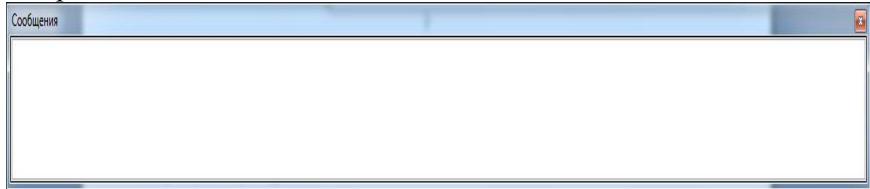


Рис. 1.6. Окно ошибок и подсказок

Управление проектом: назначение кнопок управление проектом (рис.1.7).

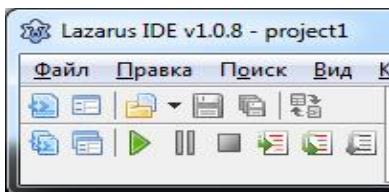


Рис. 1.7. Основные кнопки управления проектом

Верхний ряд: слева – направо:

1. **Создать модуль** – создает новый модуль форму.
2. **Открыть** – открывает проект, модуль или любой другой файл (понимаемый средой Lazarus).
3. **Сохранить** – сохраняет текущий модуль на диск, перед первым сохранением – запросит имя и папку, куда сохранить файл. Если текущий файл был сохранен и не редактировался, то кнопка серого цвета.
4. **Сохранить все** – Сохраняет все модули проекта, если модуль ни разу не сохранялся – предложит указать имя и папку для файлов.
5. **Создать форму** – создает новую форму программы и прикрепляет ее к проекту.
6. **Переключить форму/модуль** – осуществляет переход между окном формы и окном модуля. Можно переключаться с помощью клавиши <F12>.

Примечания:

- Среда Lazarus плохо понимает русские буквы в именах модулей (это связано с синтаксисом языка Free Pascal).
- По умолчанию при создании нового проекта к нему присоединена пустая форма. Проект может содержать несколько форм, в том числе и ни одной (консольное приложение, сервис, библиотека и т.д.). Первая форма, созданная в проекте, называется главной. Остальные подчиненные. При закрытии главной формы закрывается весь проект.
- Не все модули содержат форму проекта. Многие модули не имеют визуального представления, например модуль математики Math.

Нижний ряд кнопок слева – направо:

1. **Показать модули** – отображает список всех модулей проекта и позволяет переключаться на любой из них;
2. **Показать формы** – отображает список всех форм проекта и позволяет переключаться на любую из форм.
3. **Запуск проекта** – осуществляет сборку проекта и его запуск в виде исполняемого файла под отладчиком, т. е. жизнью проекта управляет среда программирования Lazarus. (Если кнопка серая – проект запущен и находится в списке запущенных программ Windows).
4. **Пауза** – работу запущенного проекта можно приостановить – поставить на «паузу».
5. **Останов** – принудительная остановка проекта, можно остановить проект сочетанием клавиш **<Ctrl + F2>**.
6. **Шаг со входом** – отладка проекта пошаговая с заходом во все процедуры и последовательная отладка этих процедур.
7. **Шаг в обход** – пошаговая отладка проекта без отладки вызываемых процедур и функций, каждая вызываемая процедура выполняется за один шаг отладки.

Примечание:

- Можно зайти как в процедуры своего проекта, так и библиотечные функции Lazarus, но в системные процедуры Windows (API Windows) попасть не получится.

Закладки компонентов – содержат наборы различных компонентов сгруппированные по различным признакам. В примерах используется компоненты из вкладок «*Standard*» и «*Additional*».

Компоненты – готовые настраиваемые блоки программ, которые можно устанавливать на форму и подключать к модулям. Для установки компонента на форму, необходимо один раз «кликнуть» на нем в панели компонентов, затем второй раз на форме – в том месте, где необходимо его разместить. Установленные компоненты можно выбирать «кликнув» по нему или с помощью рамки выделения.

Примечание:

- Выбранные компоненты имеют по краям черные прямоугольные маркеры, за которые можно перемещать объект по форме и изменять его размеры. Некоторые компоненты можно только перемещать.
- Свойства, как и типы данных, могут быть: целыми, вещественными, строковыми, логическими, множествами или сложными. События у компонентов создаются пустыми.

Инспектор объектов – форма позволяющая настроить свойства каждого компонента индивидуально. Инспектор объектов имеет несколько закладок.

Закладка «Свойство» отображает большинство свойств объекта, хотя и не все. Свойство объекта это имя и значение. Левый столбец – имя свойства, правый – значение.

Закладка «События» – позволяет посмотреть список большинства событий, на которые может реагировать компонент, а также процедуры привязанные к каждому из событий.

Форма программы – форма на которой разрабатывается интерфейс программы с помощью компонентов.

Модуль программы – окно в котором содержится исходный код на языке Object Pascal.

Окно ошибок – окно в котором отображаются все ошибки и подсказки при сборке проекта.

Примечание:

- Среда программирования Lazarus автоматически вносит изменения в код программы при добавлении компонент на форму и создании обработчиков событий. Программист создавая тело программы вносит свои изменения в код.

Ошибки могут быть:

1. **Синтаксические** – когда исходный текст не понимает среда программирования Lazarus.

2. **Логические** – когда код с точки зрения среды программирования написан верно, но программа выполняет не те действия, которые ожидает пользователь от программы.

Первый тип ошибок может отследить среда программирования и программист, второй тип – только программист.

Проект рекомендуется сохранять в индивидуальную папку, (проект – группа связанных файлов).

Для каждого проекта рекомендуется создавать следующую структуру: Для доступа к проекту по сети разместим в папке общие документы (Мой компьютер/ Общие документы), создадим папку с номером группы (например «ИТС-б-о-111»), далее фамилия (например «Иванов»), далее «1» для первой лабораторной, для второй «2» и т. д.

2. Методика и порядок выполнения работы

Основным инструментом, которым пользуется программист в процессе разработки приложения, является *Палитра Компонент* (рис.1.8).

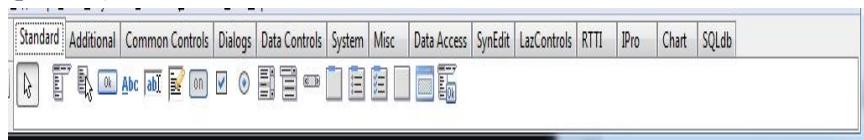


Рис. 1.8. Палитра Компонент

Палитра Компонент, позволяет выбрать нужные объекты для размещения их на *Дизайнере Форм*. Для использования *Палитры Компонент* просто первый раз щелкните мышкой на один из объектов и потом второй раз – на *Дизайнере Форм*. Выбранный объект появится на проектируемом окне, и им можно манипулировать с помощью мыши.

Палитра Компонент использует постраничную группировку объектов. Внизу Палитры находится набор закладок – *Standard*, *Additional*, *Dialogs* и т.д. Если щелкнуть мышью на одну из закладок, то можно перейти на следующую страницу *Палитры Компонент*. Принцип разбивки на страницы широко применяется в среде программирования Lazarus, и его легко можно использовать в своей программе.

Предположим, помещаете компонент *TEdit* на форму; можно двигать его с места на место. Также можно использовать границу, прорисованную вокруг объекта, для изменения его размеров. Большинством других компонент можно манипулировать тем же обра-

зом. Однако, невидимые во время выполнения программы компоненты (типа *TMenu* или *TDataBase*) не меняют своей формы.

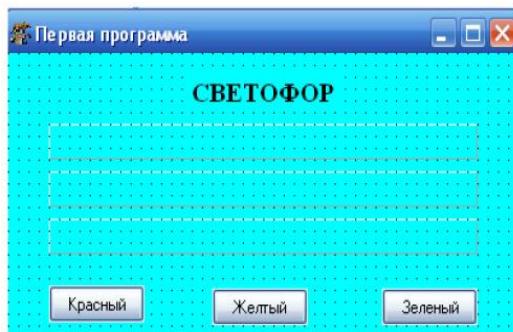
Каждый компонент является настоящим объектом, и можно менять его вид и поведение с помощью *Инспектора Объектов*, который состоит из двух страниц, каждую из которых можно использовать для определения поведения данного компонента.

Первая страница – это список свойств, вторая – список событий. Если нужно изменить что-нибудь, связанное с определенным компонентом, то это обычно делается это в *Инспекторе Объектов*. К примеру, можно изменить имя и размер компонента *Tlabel*, изменения свойства *Caption*, *Left*, *Top*, *Height*, и *Width*.

Можно использовать закладки внизу *Инспектора Объектов* для переключения между страницами свойств и событий. Страница событий связана с *Редактором*; если дважды щелкнуть мышкой на правую сторону какого-нибудь пункта, то соответствующий данному событию код автоматически запишется в *Редактор*, сам *Редактор* немедленно получит фокус, и сразу же имеете возможность добавить код обработчика данного события.

Задача 2.1.

1. Изменить заголовок окна формы с *Form1* на «Первая программа», используя свойство *Caption*.
2. Изменить цвет формы на *clAqua*, используя свойство *Color*.
3. Разместить в центре формы компоненту *Label*. Задать: надпись метки – «СВЕТОФОР», цвет метки – серый. Изменить свойство *Font*: шрифт – Times New Roman, начертание – жирный, размер – 14.
4. Расположить на форме компоненты *Panel1*, *Panel2*, *Panel3*, для которых поочередно задать свойство *Caption* пустым.
5. Расположить на форме три командные кнопки *Button1*, *Button2*, *Button3*. Задать надписи на этих кнопках «Красный», «Желтый», «Зеленый». В результате должна получиться форма, согласно рисунка.



6. Задать событие для первой командной кнопки *Button1*. Для этого необходимо выделить данный компонент и перейти в *Инспекторе Объектов* на страницу *События*. Затем на против события *OnClick* дважды щелкнуть левой кнопкой мыши. После выбора события автоматически открывается окно кода программы.

7. 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:

```
Procedure TForm1.Button1Click (Sender:TObject);
Begin
  Panel1.Color:=clRed;
  Panel2.Color:=clWhite;
  Panel3.Color:=clWhite;
End;
```

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

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

10. На диске *D*: создать папку с номером группы (в скобках указать номер подгруппы). В папке с номером группы создать папку с именем *Первая программа*. Сохранить свою программу в папке *Первая программа*.

11. Добавить появление на компоненте *Panel1* при нажатии на командную кнопку *Button1* информации «СТОЙТЕ» белым цветом, жирным шрифтом, размер шрифта – 12. Для этого необходимо в имеющейся процедуре добавить следующие действия:

```
Procedure TForm1.Button1Click (Sender:TObject);
Begin
  Panel1.Color:=clRed;
  Panel2.Color:=clWhite;
  Panel3.Color:=clWhite;
  Panel1.Caption:='СТОЙТЕ'; {задание на панели надписи}
  Panel1.Font.Color:=clWhite; {задание цвета шрифта}
  Panel1.Font.Size:=12; {задание размера шрифта}
  Panel1.Font.Style:=[fsBold]; {задание начертания шрифта}
End;
```

12. Добавить появление на компоненте *Panel2* при нажатии на командную кнопку *Button2* информации «ВНИМАНИЕ» белым цветом, жирным шрифтом, размер шрифта – 12.

13. Добавить появление на компоненте *Panel3* при нажатии на командную кнопку *Button3* информации «ИДИТЕ» белым цветом, жирным шрифтом, размер шрифта – 12.

14. Сохранить изменения в программе и запустить ее на исполнение.

15. На форме добавить командную кнопку *Button4*. Задать для нее надпись «Автор». При нажатии на кнопку должно выводится 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;
```

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. № варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Поместите на Форму элемент управления *Tbutton*. При нажатии на эту кнопку должен производится:

Вариант	Условие задачи
1.	Расчет функции возведения в квадрат суммы двух целых чисел.
2.	Расчет функции возведения в квадрат разности двух целых чисел.
3.	Расчет функции возведения в квадрат произведения двух целых чисел.
4.	Расчет функции возведения в квадрат частного двух целых чисел.
5.	Расчет функции возведения в куб целого числа.
6.	Расчет функции возведения в куб суммы двух целых чисел.
7.	Расчет функции возведения в куб разности двух целых чисел.
8.	Расчет функции возведения в куб произведения двух целых чисел.
9.	Расчет функции возведения в куб частного двух целых чисел.

10.	Расчет функции вычисления суммы двух вещественных чисел.
11.	Расчет функции вычисления разности двух вещественных чисел.
12.	Расчет функции вычисления произведения двух вещественных чисел.
13.	Расчет функции вычисления частного двух вещественных чисел.
14.	Расчет функции возведения в квадрат суммы двух вещественных чисел.
15.	Расчет функции возведения в квадрат разности двух вещественных чисел.
16.	Расчет функции возведения в квадрат произведения двух вещественных чисел.
17.	Даны три числа, проверить которое из них кратно трем и вывести об этом сообщение.
18.	Проверка целого числа на четность по щелчку на кнопке с символом «результат».
19.	Вычисление факториала $N!$ по заданному натуральному числу N .
20.	Вычисление суммы первых N натуральных чисел. Сумма должна выводится по щелчку на кнопке с символом «сумма».

Пример индивидуального задания

Условие задачи	Интерфейс
<i>Создать простейший пользовательский интерфейс, для программы сложения двух чисел ($a + b = c$), содержащий объекты Button, TextBox - для ввода значений переменных a и b и вывода значения c, Label - для поясняющих надписей.</i>	

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и

ициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Главные составные части среды программирования LAZARUS.
2. Каковы основные элементы палитры компонент ?
3. Как производится выравнивание объектов ?
4. Как производится изменение размеров объектов ?
5. Как создается новый проект ?
6. Назначение инспектора объектов ?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

2. РЕАЛИЗАЦИЯ В IDE LAZARUS ПРОСТЕЙШИХ АЛГОРИТМОВ

Цель: научится управлять проектом в среде Lazarus, а также добавлять и удалять формы и модули в проект.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория

Структура проекта Lazarus

Любой проект в Lazarus – это совокупность файлов, из которых создается единый исполняемый файл. В простейшем случае список файлов проекта имеет вид:

- файл описания проекта (**.lpi**);
- файл проекта (**.lpr**);
- файл ресурсов (**.lrs**);
- модуль формы (**.lfm**);
- программный модуль (**.pas**).

После компиляции программы из всех файлов проекта создается единый исполняемый файл, имя этого файла всегда совпадает с именем проекта.

Программный модуль, или просто модуль, – это отдельно компилируемая программная единица, которая представляет собой набор типов данных, констант, переменных, процедур и функций. Любой модуль имеет следующую структуру:

- **unit ...; //** Заголовок модуля
- **interface //** Раздел описаний
- **implementation //** Раздел реализаций
- **end. //** Конец модуля

Заголовок модуля – это зарезервированное слово **unit**, за которым следует имя модуля и точка с запятой. В разделе описаний, который открывается служебным словом *interface*, описывают программные элементы – типы, классы, процедуры и функции:

- **Interface**
- **uses** ..., ..., ...; // Список модулей
- **const** ..., ..., ...; // Список констант
- **var** ..., ..., ...; // Список переменных
- **procedure** ...; // Имя процедуры
- ...
- **function** ...; // Имя функции
- ...

Раздел *implementation* содержит программный код, реализующий механизм работы описанных программных элементов (тексты процедур обработки событий, процедуры и функции, созданные программистом). Процедуры и функции в Lazarus также построены по модульному принципу.

Наряду с визуальными приложениями, Lazarus позволяет разрабатывать и обычные консольные приложения, которые также могут быть созданы в оболочке Free Pascal и в текстовом редакторе *Geany*.

Рассмотрим подробно структуру консольного приложения. Она имеет следующий вид:

- **program** ...; // Заголовок программы
- **uses** ..., ..., ...; // Список модулей
- **const** ..., ..., ...; // Список констант
- **type** ..., ..., ...; // Список типов
- **var** ..., ..., ...; // Список переменных
- **procedure** ...; // Имя процедуры
- **function** ...; // Имя функции
- **Begin**
- ... // Тело программы
- **end.**

Заголовок программы состоит из служебного слова **program**, имени программы, образованного по правилам использования идентификаторов, и точки с запятой.

Предложение **uses** ..., ..., ...; предназначено для подключения модулей. В модулях находятся функции и процедуры языка. Для использования функций и процедур, находящихся в модуле, необходимо в тексте программы подключить их с помощью предложения **uses**.

В языке Free Pascal должны быть описаны все переменные, типы, константы, которые будут использоваться программой. В стандартном языке Pascal порядок следования разделов в программе жестко установлен, во Free Pascal такого строгого требования нет. В программе может быть несколько разделов описания констант, переменных и т.д., но все они должны быть до тела программы.

Тело программы начинается со слова *begin*, затем следуют операторы языка Pascal, реализующие алгоритм решаемой задачи.

Операторы в языке Pascal отделяются друг от друга точкой с запятой и могут располагаться в одну строчку или начинаться с новой строки (в этом случае их также необходимо разделять точкой с запятой).

Назначение символа « ; » – отделение операторов друг от друга.

Тело программы заканчивается служебным словом *end*. Несмотря на то, что операторы могут располагаться в строке как угодно, рекомендуется размещать их по одному в строке, а в случае сложных операторов – от каждого в несколько строк.

Объекты формы и их свойства. На палитре компонентов располагаются различные объекты пользовательского интерфейса. Наиболее востребованные объекты:



TButton – Кнопка.



TEdit – Текстовое поле.



TLabel – Метка.

Данные объекты расположены на вкладке *Standart* Палитры компонентов.

Все объекты характеризуются:

- свойствами (цвет, положение на экране и пр.),
- методами (действия или задачи, которые выполняет объект)
- событиями (на какое событие должен реагировать объект).

2. Методика и порядок выполнения работы

Задача 2.1. Теперь можно приступить к визуальному программированию. У нас есть один объект – форма *Form1*. Изменим некоторые его свойства с помощью *Инспектора объектов*. Перейдем в окно *Инспектора объектов* и найдем там свойство *Caption* (*Заголовок*). По умолчанию это свойство имеет значение *Form1*. Изменим его на слово *ПРИМЕР1* (рис. 2.1). Это изменение сразу же отразится на форме – в поле заголовка появится надпись – *ПРИМЕР1* (рис. 2.2). Аналогично можно изменить размеры формы, присвоив новые значения свойствам *Height* (*Высота*) и *Width* (*Ширина*), еще проще это сделать, изменив положение границы формы с помощью кнопки мыши, как это делается со всеми стандартными окнами.

Изменим еще одно свойство формы – *Position* (*Положение формы на экране*). Значения этого свойства не вводятся вручную, а выбираются из списка (рис. 2.3). Если выбрать свойство *poScreenCenter*, то форма всегда будет появляться в центре экрана.

Напомним, что пишем программу про кнопку. Пришло время поместить ее на форму. Для этого обратимся к *Панели компонентов* и найдем там компонент *Tbutton*. Чтобы разместить компонент на форме, нужно сделать два щелчка мышкой: первый – по компоненту, второй – по окну формы. В результате форма примет вид, представленный на рис. 2.4.

Теперь у нас два объекта: форма *Form1* и кнопка *Button1*. Напомним, что изменения значений свойств в окне *Инспектора объектов* относятся к выделенному объекту. Поэтому выделим объект *Button1* и изменим его заголовок *Caption* и размеры *Height*, *Width*. В заголовке напишем фразу «*Щелкни мышкой!*», установим ширину 135 пикселей, а высоту – 25 пикселей. Сохраним проект (*Проект* ⇒ *Сохранить проект*).

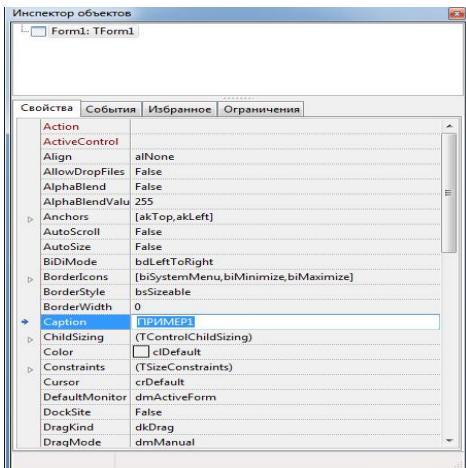


Рис. 2.1. Изменение свойства формы
Caption

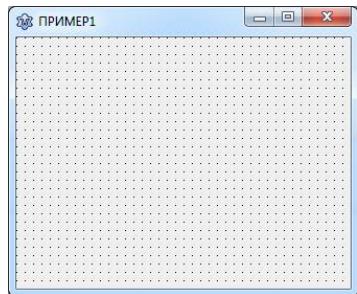


Рис. 2.2. Изменение заголовка
формы

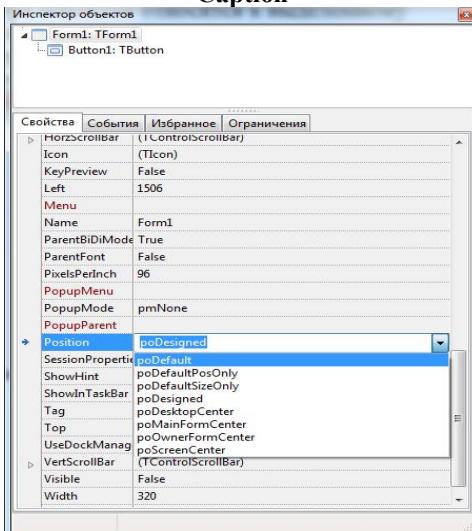


Рис. 2.3. Изменение свойства
Position

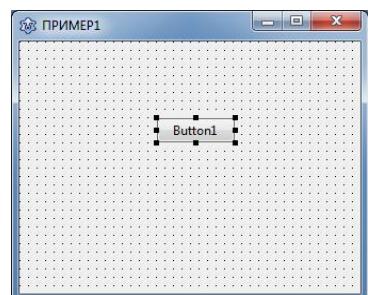


Рис. 2.4. Размещение кнопки
на форме

Теперь можно запустить проект на компиляцию и сборку. Для того чтобы посмотреть, как работает наша программа, ее необходимо запустить на выполнение. Сделать это можно командой «Запуск»→«Запуск», функциональной клавишей «F9» или кнопкой «Запуск» в панели инструментов (рис. 2.5).

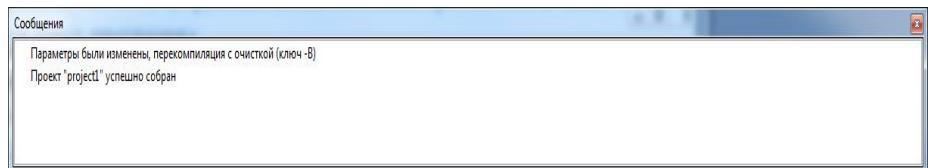


Рис. 2.5. Кнопка Запуск в панели инструментов Lazarus

Созданное окно с кнопкой должно выглядеть, как на рис. 2.6.

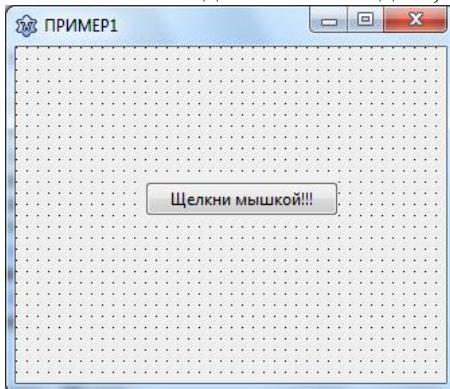


Рис.2.6. Результат работы программы

Если щелкнуть по кнопке и то убедитесь, что ничего не произойдет. Ведь пока не написано ни одной строчки программного кода. Можно сказать, что разработана только внешняя часть программы, но отсутствует функциональная. Несмотря на это, в окне редактора появился текст программы:

```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs,
StdCtrls;
type
```

```
{ TForm1 }
TForm1 = class(TForm)
Button1: TButton;
private
{ private declarations }
public
{ public declarations }
end;
var
Form1: TForm1;
implementation
initialization
{$I unit1.lrs}
end.
```

События и процедуры обработки событий. Большинство компонентов могут реагировать на определенные действия пользователя – *события*. Наиболее распространенными событиями являются: щелчок кнопкой мыши на кнопке, ввод данных в поле ввода, наведение курсора мыши на компонент и т.д. При этом именно разработчик программы решает, как компонент будет реагировать на возникшее событие. Реакция программы на определенное событие реализуется с помощью *процедур обработки событий*.

Рассмотрим структуру процедуры обработки события. Первая строка включает в себя служебное слово *procedure*, которое означает начало процедуры, затем идет имя процедуры, которое состоит из двух частей: Тип объекта-родителя (в нашем случае это тип *TForm1*, так как объектом-родителем является форма *Form1*) и, через точку, имя самого события. В скобках указан объект-инициатор события. Затем идут операторные скобки *begin ... end*, внутри этих скобок и будет содержаться код, описывающий действия, которые должны происходить при возникновении события нажатия кнопки.

Выполнив двойной щелчок на кнопке *Button1*, расположенной на форме мы вызовем обработчик события, который создаст «скелет» процедуры обработки события *OnClick* нажатия кнопки в окне *Редактора кода*.

Вернемся к программе и подумаем, что же мы хотим от нашей кнопки. После запуска программы на экране появилось окно, на котором расположена кнопка с надписью «Щелкни мышкой!». Предположим, что если щелкнуть по ней, она «обрадуется» и сообщит: «УРА! ЗАРАБОТАЛО!».

Для воплощения этой идеи завершим работу программы, закрыв окно с кнопкой обычным способом (щелчком по крестику в правом верхнем углу) и перейдем в режим редактирования. Убедимся в том, что объект кнопка «**Button1**» выделен, и обратимся к вкладке «**События**» Инспектора объектов. Напомним, что здесь расположены описания событий. Выберем событие «**OnClick**» – *обработка щелчка мыши и дважды щелкнем в поле справа от названия*.

В результате действий в окне редактора появиться следующий текст:

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

На данном этапе изложения материала данный текст – это фрагмент программного кода, именуемый *подпрограммой*. О назначении этой подпрограммы можно догадаться по ее имени `TForm1.Button1Click`: на форме `Form1` для объекта кнопка `Button1` обрабатывается событие «щелчок мыши» `Click`. Все команды, написанные между словами `begin` и `end`, будут выполняться при наступлении указанного события.

Теперь установим курсор между словами `begin` и `end` созданной подпрограммы и напишем: `Button1.Caption:='УРА! ЗАРАБОТАЛО!';`

Эта запись означает изменение свойства кнопки. Только теперь мы выполнили его не с помощью инспектора объектов, а записав оператор языка программирования. Прочитать его можно так: присвоить (`:=`) свойству «`Caption`» объекта «`Button1`» значение «`УРА! ЗАРАБОТАЛО!`». Поскольку присваиваемое значение – строка, оно заключено в одинарные кавычки. Теперь, текст подпрограммы в окне редактора имеет вид:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Caption:='УРА! ЗАРАБОТАЛО!';
end;
```

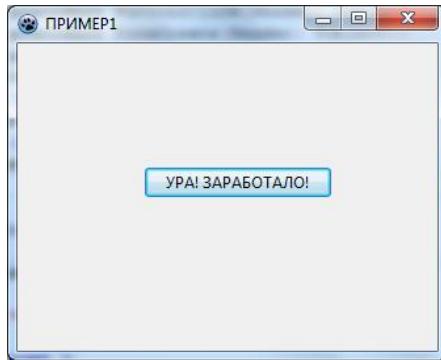


Рис. 2.7. Окончательный результат работы программы

Сохраним созданную программу, запустим ее на выполнение и убедимся, что кнопка реагирует на щелчок мыши (рис. 2.7).

Закрыть проект можно командой «Проект⇒Закрыть проект».

Задача 2.2. Известны длины сторон треугольника a , b и c . Вычислить площадь S , периметр P и величины углов α , β и γ треугольника (рис. 2.8).

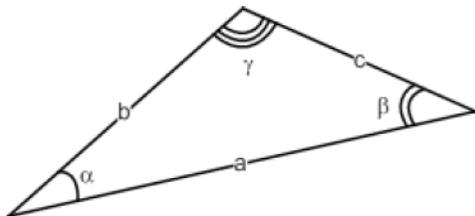


Рис. 2.8. Иллюстрация к задаче 2.2

Для вычисления площади треугольника применим теорему Герона:

$$S = \sqrt{r(r-a)(r-b)(r-c)},$$

где полупериметр: $r = (a+b+c)/2$;

один из углов найдем по теореме косинусов: $\cos(\alpha) = (b^2 + c^2 - a^2) / 2 \cdot b \cdot c$;

второй – по теореме синусов: $\sin(\beta) = (b/a) \cdot \sin(\alpha)$;

третий – по формуле: $\gamma = \pi - (\alpha + \beta)$.

Самостоятельно разработайте внешний вид данной программы. Разместите на форме десять меток, три поля ввода и одну кнопку. Измените их заголовки (свойство *Caption*).

Свойства выбранных компонент (рис. 2.9):

- **Form1** – Caption – Параметры треугольника;
- **Label1** – Caption – Введите длины сторон;
- **Label2** – Caption – a=;
- **Label3** – Caption – b=;
- **Label4** – Caption – c=;
- **Label5** – Caption – Величины углов;
- **Label6** – Caption – alfa=;
- **Label7** – Caption – betta=;
- **Label8** – Caption – gamma=;
- **Label9** – Caption – Периметр P=;
- **Label10** – Caption – Площадь S=;
- **Button1** – Caption – ВЫЧИСЛИТЬ;

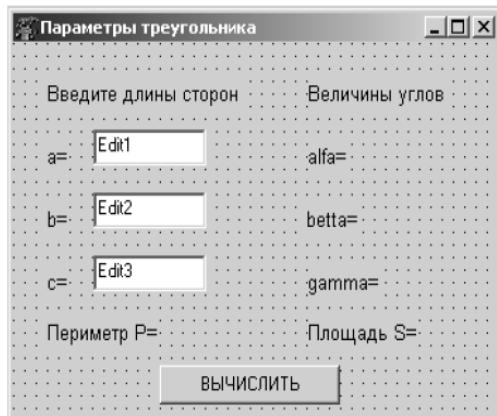


Рис. 2.9. Проект формы к задаче 3.2

Итак, проект формы готов. В окне программного кода среды Lazarus автоматически сформировал структуру модуля, перечислив названия основных разделов. Двойной щелчок по кнопке «Вычислить» приведет к созданию процедуры *TForm1.Button1Click* в разделе *implementation*:

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

и ее описанию в разделе *interface*. Понятно, что созданная процедура не содержит ни одной команды. Задача программиста заполнить шаблон описаниями и операторами. Все команды, указанные в процедуре между словами *begin* и *end*, будут выполнены при щелчке по кнопке «Выполнить».

В нашем случае процедура *TForm1.Button1Click* будет иметь вид:

```
procedure TForm1.Button1Click(Sender: TObject);
//Описание переменных:
// a, b, c – стороны треугольника;
// alfa, betta, gamma – углы треугольника;
// S – площадь треугольника;
// r – полупериметр треугольника
//Все переменные вещественного типа.
var a, b, c, alfa, betta, gamma, S, r: real;
begin
//Из полей ввода Edit1, Edit2, Edit3
//считываются введенные строки, с помощью функции
StrToFloat(x)
//преобразовываются в вещественные числа и записываются в
переменные a, b, c.
a:=StrToFloat(Edit1.Text);
b:=StrToFloat(Edit2.Text);
c:=StrToFloat(Edit3.Text);
//Вычисление значения полупериметра.
R:=(a+b+c)/2;
//Вычисление значения площади, для вычисления применяется
функция:
// sqrt(x) – корень квадратный из x.
S:=sqrt(r*(r-a)*(r-b)*(r-c));
//Вычисление значения угла alfa в радианах.
//Для вычисления применяем функции: arccos(x) – арккосинус x;
// sqr(x) – возведение x в квадрат.
alfa:=arccos((sqr(b)+sqr(c)-sqr(a))/2/b/c);
//Вычисление значения угла betta в радианах.
```

```

//Для вычисления применяем функции: arcsin(x) – арксинус x;
betta:=arcsin(b/a*sin(alfa));
//Вычисление значения угла гамма в радианах.
//Математическая постоянная определена функцией без аргу-
ментов pi.
gamma:=pi-(alfa+betta);
//Перевод радиан в градусы.
alfa:=alfa*180/pi;
betta:=betta*180/pi;
gamma:=gamma*180/pi;
//Для вывода результатов вычислений используем операцию
слияния строк «+»
//и функцию FloatToStrF(x), которая преобразовывает веществен-
ную переменную x
//в строку и выводит ее в указанном формате, в нашем случае
под переменную
// отводится три позиции, включая точку и ноль позиций после
точки.
//Величины углов в градусах выводятся на форму
//в соответствующие объекты типа надпись.
Label6.Caption:='alfa=' + FloatToStrF(alfa,ffFixed,3,0);
Label7.Caption:='betta=' + FloatToStrF(betta,ffFixed,3,0);
Label8.Caption:='gamma=' + FloatToStrF(gamma,ffFixed,3,0);
//Используем функцию FloatToStrF(x) для форматированного
вывода,
// в нашем случае под все число отводится пять позиций,
//включая точку, и две позиций после точки.
//Значения площади и периметра выводятся на форму.
Label9.Caption:='Периметр P=' + FloatToStrF(2*r,ffFixed,5,2);
Label10.Caption:='Площадь S=' + FloatToStrF(S,ffFixed,5,2);
end;

```

Обратите внимание, что было написано всего десять команд, предназначенных для решения поставленной задачи. Остальной текст в окне редактора создается автоматически. В результате весь программный код имеет вид:

```

unit Unit1;
{$mode objfpc} {$H+}
interface

```

```

uses
  Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs,
StdCtrls, Math;
type
  { TForm1 }
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label1: TLabel;
    Label10: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;
  var
    Form1: TForm1;
  implementation
    { TForm1 }
    procedure TForm1.Button1Click(Sender: TObject);
    var a, b, c, alfa, betta, gamma, S, r: real;
    begin
      a:=StrToFloat(Edit1.Text);
      b:=StrToFloat(Edit2.Text);
      c:=StrToFloat(Edit3.Text);
      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='+
FloatToStrF(alfa,ffFixed,3,0);
Label7.Caption:='betta='+
FloatToStrF(betta,ffFixed,3,0);
Label8.Caption:='gamma='+
FloatToStrF(gamma,ffFixed,3,0);
Label9.Caption:='Периметр P='+
FloatToStrF(2*r,ffFixed,5,2);
Label10.Caption:='Площадь S='+
FloatToStrF(S,ffFixed,5,2);
end;
initialization
{$I unit1.lrs}
end.

```

На рис. 2.10 представлено диалоговое окно, которое появится, если запустить эту программу, щелкнув по кнопке «Вычислить».

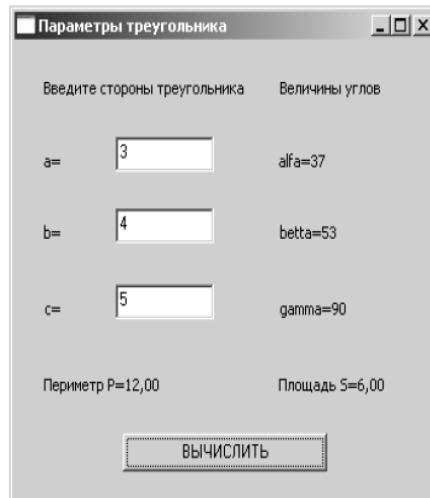


Рис. 2.10. Результат работы программы к задаче 2.2

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Создайте приложение вычисления квадрата разности двух целых чисел.
2.	Известна гипотенуза с и прилежащий угол а прямоугольного треугольника. Найти площадь треугольника.
3.	Известна диагональ квадрата d. Вычислить площадь S и периметр Р квадрата.
4.	Известна диагональ прямоугольника d и угол а между диагональю и большей стороной. Вычислить площадь S прямоугольника.
5.	Треугольник задан величинами своих сторон – a, b, c. Найти углы треугольника – α , β , γ .
6.	Создайте приложение нахождения наименьшего из трех целых чисел. Результат должен выводиться по щелчку на кнопке с символом «наименьшее».
7.	В треугольнике известен катет а и площадь S. Найти величину гипотенузы с, второго катета b и углов а и b.
8.	Известна площадь квадрата S. Вычислить сторону квадрата а, диагональ d и площадь S1 описанного вокруг квадрата круга.
9.	Создайте приложение ввода двух целых чисел и вычисления их произведения, суммы квадратов и разности квадратов этих чисел. Каждая операция должна выполняться по нажатию кнопки соответствующего действия и выводить результат.
10.	Создайте приложение «Угадай число», в котором компьютер будет загадывать случайное целое число в диапазоне от 0 до 10, а пользователь должен угадать это число. В случае угадывания пользователем загаданного компьютером числа следует вывести сообщение «Вы угадали», в противном случае «Вы проиграли».

11.	Заданы два катета прямоугольного треугольника. Вычислить его площадь и периметр.
12.	Известна гипотенуза с и противолежащий угол а прямоугольного треугольника. Найти периметр треугольника.
13.	Известна диагональ ромба d. Вычислить его площадь S и периметр P.
14.	Известна длина диагоналей прямоугольника d и угол а между ними. Вычислить площадь S прямоугольника.
15.	В прямоугольном треугольнике известен катет b и площадь S. Вычислить периметр треугольника.
16.	Известно значение периметра Р равностороннего треугольника. Вычислить его площадь.
17.	Задан периметр квадрата Р. Вычислить сторону квадрата a, диагональ d и площадь S.
18.	В равнобедренном треугольнике известно основание с и высота h. Найти площадь треугольника S и периметр P.
19.	Создайте приложение нахождение наибольшего числа из трех целых чисел. Результат должен выводиться по щелчку на кнопке с символом «наибольшее».
20.	Металлический слиток имеет форму цилиндра, площадь поверхности S, высота h, плотность а. Вычислить массу m слитка.

Задание 3.2. Вычислить значение функции трех переменных при заданных значениях параметров:

Вариант	Условие задачи	
1.	$x = 2y + 3t - z$	при $y = 2; t = 5/(1+y^2); z = 4$
2.	$x = 3y^2 / (4z - 2t^2)$	при $y = t + 2z; t = 0.5; z = 6$
3.	$x = 4y^2 / (4y - 2t^3)$	при $y = t^2; t = 1; z = 3$
4.	$x = (4y^2 - z) / t$	при $y = (t+z)^2; t = 2; z = 3$
5.	$x = 6y^2 - (z+1)t$	при $y = 2; t = (2z+z^2)^3; z = 4$

6.	$x = 3y + 7t^3 - z^2$	<i>npru</i> $y = t^2 + z^2; \quad t = 2; \quad z = 1$
7.	$x = 12y + 3(t-1) - z$	<i>npru</i> $y = 3; \quad t = 2; \quad z = t + 1$
8.	$x = 3y^3 + 3t - z$	<i>npru</i> $y = 3; \quad t = 1; \quad z = (t + 2)^5$
9.	$x = 2y^7 - 3t - z^2$	<i>npru</i> $y = 3; \quad t = 2; \quad z = t^2 + 1$
10.	$x = 2y^2 + 3t^2 - z^2$	<i>npru</i> $y = 5; \quad t = 1; \quad z = (t + 2)$
11.	$x = 5y^2(14z - 12t^2)$	<i>npru</i> $y = 2; \quad t = 5/(1+y^2); \quad z = 4$
12.	$x = 22y + 30t - z$	<i>npru</i> $y = t + 5z; \quad t = 0.5; \quad z = 5$
13.	$x = 3y^2 / (4y - 12t^3)$	<i>npru</i> $y = t^2; \quad t = 13; \quad z = 3$
14.	$x = (3y^2 - 2z) / 2t$	<i>npru</i> $y = (t + 2z)^2; \quad t = 11; \quad z = 5$
15.	$x = 3y^3 - (2z + 1)t$	<i>npru</i> $y = 2; \quad t = (2z + z^2)^3; \quad z = 2$
16.	$x = 2y + 7t^3 - 3z^2$	<i>npru</i> $y = 3t^2 + z^2; \quad t = 2; \quad z = 1$
17.	$x = 10y + 3(t-5) - 2z$	<i>npru</i> $y = 5; \quad t = 2; \quad z = t + 1$
18.	$x = 2y^3 + 2t - 2z$	<i>npru</i> $y = 3; \quad t = 1; \quad z = (t + 2)^5$
19.	$x = y^5 - 2t - 3z^2$	<i>npru</i> $y = 3; \quad t = 2; \quad z = t^2 + 1$
20.	$x = y^2 + 11t^2 - 2z^2$	<i>npru</i> $y = 5; \quad t = 1; \quad z = (t + 2)$

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Структура проекта LAZARUS.
2. Элементы языка Pascal (символы, идентификаторы, ключевые слова, комментарии) и типы данных (состав и правила описания).
3. Константы и переменные (раздел констант и правила описания констант).
4. Новые типы данных (раздел новых типов и правила описания новых типов).
5. Что такое окно формы?
6. Что такое редактор кода?
7. Назначение *Инспектора объектов* (Object Inspector).
8. Что такое интерфейс приложения?
9. Для какой цели используется палитра компонентов?
- 10.Что такая функциональность приложения?
- 11.Какой язык программирования используется при работе с Lazarus?
- 12.Как назначать события для различных элементов формы?
- 13.Какие из элементов *Палитры компонентов* вы знаете?
- 14.Стандартные операции (целочисленное деление, остаток от деления, операции отношения, логические операции).
- 15.Стандартные функции Lazarus.

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

3. ПРОГРАММИРОВАНИЕ ЗАДАЧ ЛИНЕЙНОЙ СТРУКТУРЫ

Цель: освоить написание и отладку программ задач линейной структуры в интегрированной среде разработки программного обеспечения Lazarus.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория

Алгоритм линейной структуры – это алгоритм, в котором блоки выполняются в указанном порядке, последовательно друг за другом. Программа линейной структуры реализует соответствующий линейный алгоритм.

Чаще всего линейные алгоритмы используются для программирования вычислений по формулам.

2. Методика выполнения работы

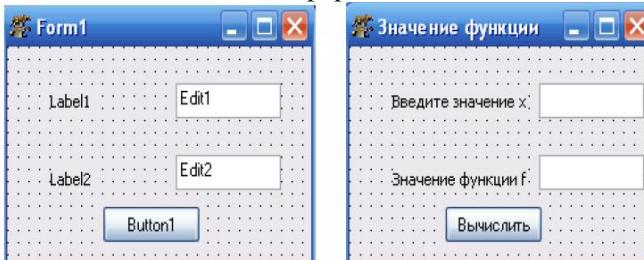
Задача 2.1. Решение задачи на вычисление значения функции.

Вычислить значение функции $f = 1/((x+1)^2 + 2) + x^5 - \sqrt[3]{x}$. Решение любой задачи начинается с определения входных и выходных переменных.

входные переменные: x: integer

выходные переменные: f:real

Размещаем компоненты на форме и задаем их свойства.



- **Form1** – Caption – Значение функции;
- **Label1** – Caption – Введите значение x;
- **Label2** – Caption – Значение функции f;
- **Button1** – Caption – Вычислить;
- **Edit1** – Text – пусто;
- **Edit2** – Text – пусто;

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

Процедура для вычисления значения функции будет иметь вид (обратите внимание, что при написании программы все служебные слова выделяются автоматически жирным шрифтом):

```
procedure TForm1.Button1Click(Sender: TObject);
var x:integer;
f:real;
begin
x:=strtoint(edit1.text);
f:=1/(sqr(x+1)+2)+exp(5*ln(x))-sqrt(x);
edit2.text:=floattostr(f);
end;
```

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

Задача 2.2. Решение задачи на нахождение площади и периметра треугольника.

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

Прежде чем приступить к написанию программы, вспомним математические формулы, необходимые для решения задачи.

Для вычисления площади треугольника применим теорему Герона:

$$S = \sqrt{r(r-a)(r-b)(r-c)},$$

где полупериметр: $r = (a+b+c)/2$;

Первый угол α найдем по теореме косинусов: $\cos(\alpha) = (b^2 + c^2 - a^2) / (2 \cdot b \cdot c)$.

Второй угол β найдем по теореме синусов: $\sin(\beta) = (b/a) \cdot \sin(\alpha)$.

Третий угол γ найдем по формуле: $\gamma = \pi - (\alpha + \beta)$.

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

Определим в программе входные и выходные переменные:

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

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

Размещаем компоненты на форме и задаем их свойства.

– **Form1** – Caption – Параметры треугольника;

– **Label1** – Caption – Введите длины сторон;

– **Label2** – Caption – $a =$;

– **Label3** – Caption – $b =$;

– **Label4** – Caption – $c =$;

– **Label5** – Caption – Величины углов;

– **Label6** – Caption – $\alpha =$;

– **Label7** – Caption – $\beta =$;

– **Label8** – Caption – $\gamma =$;

– **Label9** – Caption – Периметр $P =$;

– **Label10** – Caption – Площадь $S =$;

– **Button1** – Caption – Вычислить;

– **Button2** – Caption – Очистить;

– **Edit1** – Text – пусто;

– **Edit2** – Text – пусто;

– **Edit3** – Text – пусто;

– **Edit4** – Text – пусто;

– **Edit5** – Text – пусто;

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

uses Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, StdCtrls, Buttons, Math;

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);
r:=(a+b+c)/2;
P:=r*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;
```

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. № варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Создайте приложение нахождения объема пирамиды.
2.	Создайте приложение нахождения объема цилиндра.
3.	Создайте приложение нахождения объема куба.
4.	Создайте приложение нахождения объема конуса.
5.	Создайте приложение нахождения площади круга.
6.	Создайте приложение нахождения площади прямоугольника.
7.	Создайте приложение нахождения периметра прямоугольника.
8.	Создайте приложение нахождения периметра квадрата.
9.	Создайте приложение нахождения площади пирамиды.
10.	Создайте приложение нахождения площади цилиндра.
11.	Создайте приложение нахождения площади куба.
12.	Создайте приложение вычисления суммы и произведения двух вещественных чисел.
13.	Создайте приложение вычисления показательной функции для вещественного числа.
14.	Создайте приложение вычисления функции $\sin(x)$ для пяти значений аргумента.
15.	Создайте приложение вычисления функции $\cos(x)$ для семи значений аргумента.
16.	Создайте приложение вычисления корней квадратного уравнения. Если решение не существует, то вывести об этом сообщение в отдельном окне.
17.	Создайте приложение нахождения наибольшего из трех чисел. Сообщение выводить в отдельном окне.
18.	Создайте приложение нахождения наименьшего из трех чисел. Сообщение выводить в отдельном окне.
19.	Заданы два катета прямоугольного треугольника. Вычислить его площадь и периметр.
20.	Известна гипотенуза с и противолежащий угол а прямоугольного треугольника. Найти периметр треугольника.

Задание 3.2. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи	Вариант	Условие задачи
1.	$r = a(b - 4c)$	2.	$r = 3a + bc$
3.	$r = a + b(-c)$	4.	$r = 2ab - 3c$
5.	$r = (a + b)c$	6.	$r = 3abc$
7.	$r = -a + 5bc$	8.	$r = -6a - 2b + 4c$
9.	$r = 5a + 6bc$	10.	$r = -2a + bc$
11.	$r = 5b - 11c$	12.	$r = 3a(b - 2c)$
13.	$r = 3a + 4c$	14.	$r = 4a - b(-c)$
15.	$r = 2a + 3b + 4c$	16.	$r = (3a + 2b)c$
17.	$r = 6(-a)(b + c)$	18.	$r = a + 7b + 2c$
19.	$r = 2a + 5b - 3c$	20.	$r = 7a + 5b - 3c$

Задание 3.3. Линейные алгоритмы

Вариант	Условие задачи
1.	Даны два ненулевых числа. Найти их сумму, разность, произведение и частное.
2.	Даны два числа. Найти среднее арифметическое их квадратов и среднее арифметическое их модулей.
3.	Скорость лодки в стоячей воде V км/ч, скорость течения реки U км/ч ($U < V$). Время движения лодки по озеру T1 ч, а по реке (против течения) — T2 ч. Определить путь S, пройденный лодкой.
4.	Скорость первого автомобиля V1 км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили удаляются друг от друга.
5.	Скорость первого автомобиля V1 км/ч, второго — V2 км/ч, расстояние между ними S км. Определить расстояние между ними через T часов, если автомобили первоначально движутся навстречу друг другу.
6.	Найти периметр и площадь прямоугольного треугольника, если даны длины его катетов a и b.
7.	Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
8.	Найти длину окружности и площадь круга заданного радиуса R. В качестве значения Pi использовать 3.14.

9.	Найти площадь кольца, внутренний радиус которого равен R1, а внешний радиус равен R2 ($R1 < R2$). В качестве значения Pi использовать 3.14.
10.	Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей.
11.	Дана длина окружности. Найти площадь круга, ограниченного этой окружностью. В качестве значения Pi использовать 3.14.
12.	Дана площадь круга. Найти длину окружности, ограничивающей этот круг. В качестве значения Pi использовать 3.14.
13.	Найти периметр и площадь равнобедренной трапеции с основаниями a и b ($a > b$) и углом alpha при большем основании (угол дан в радианах).
14.	Найти периметр и площадь прямоугольной трапеции с основаниями a и b ($a > b$) и острым углом alpha (угол дан в радианах).
15.	Найти расстояние между двумя точками с заданными координатами (x_1, y_1) и (x_2, y_2) .
16.	Даны координаты трех вершин треугольника $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Найти его периметр и площадь.
17.	Найти корни квадратного уравнения $A \cdot x^2 + B \cdot x + C = 0$, заданного своими коэффициентами A, B, C (коэффициент A не равен 0), если известно, что дискриминант уравнения неотрицателен.
18.	Найти решение системы уравнений вида $A_1 \cdot x + B_1 \cdot y = C_1, A_2 \cdot x + B_2 \cdot y = C_2$, заданной своими коэффициентами $A_1, B_1, C_1, A_2, B_2, C_2$, если известно, что данная система имеет единственное решение.
19.	Дано целое четырехзначное число. Используя операции div и mod, найти сумму его цифр.
20.	Дано целое четырехзначное число. Используя операции div и mod, найти произведение его цифр.

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Элементы языка Pascal (символы, идентификаторы, ключевые слова, комментарии) и типы данных (состав и правила описания).
2. Константы и переменные (раздел констант и правила описания констант).
3. Новые типы данных (раздел новых типов и правила описания типов).
4. Как назначать события для различных элементов формы?
5. Стандартные операции (целочисленное деление, остаток от деления, операции отношения, логические операции).
6. Перечислите состав проекта Lazarus и опишите назначение различных файлов?
7. Каково назначение обработчиков событий?
8. Каким образом можно инициировать создание процедуры-обработчика события?
9. Какими способами можно изменить свойства компонентов?
10. Перечислите компоненты, используемые в работе и опишите их назначение?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

4. ПРОГРАММИРОВАНИЕ ЗАДАЧ ВЕТВЯЩЕЙСЯ СТРУКТУРЫ

Цель: освоить написание и отладку программ задач ветвящейся структуры в интегрированной среде разработки программного обеспечения Lazarus.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория

Разветвляющийся алгоритм – это алгоритм, содержащий хотя бы одно условие; он позволяет, в зависимости от условий, выполнять команды, содержащиеся в ветвях алгоритма.

К *разветвляющимся программам* приводят задачи, в которых, в зависимости от некоторого условия, вычисления производятся тем или иным путем.

Условный оператор – это оператор, который выполняет команду или группу команд в зависимости от определенного условия. Условный оператор служит для ветвлений в программе и имеет полную и сокращенную форму записи.

2. Методика и порядок выполнения работы

Задача 2.1. Решение задачи на нахождение корней квадратного уравнения.

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

Определим переменные задачи:

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

промежуточные переменные d(дискриминант), y1, y2 (вспомогательные переменные): real

результат: x1, x2 (корни уравнения): real

Можно выделить следующие этапы решения задачи:

1) Ввод коэффициентов квадратного уравнения a, b и c.

2) Вычисление дискриминанта d по формуле $d = b^2 - 4ac$.

3) Проверка знака дискриминанта. Если $d \geq 0$, то корни уравнения действительные и находятся по формулам: $x_{1,2} = (-b \pm \sqrt{|d|})/2a$.

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

$$x1,2 = (-b \pm \sqrt{|d|})/2a.$$

Для нахождения корней будем использовать вспомогательные переменные:

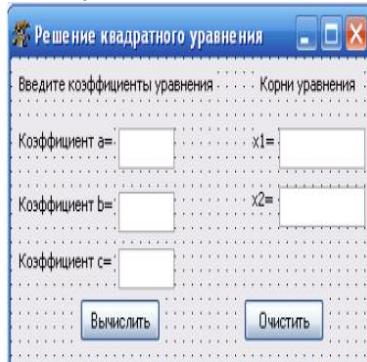
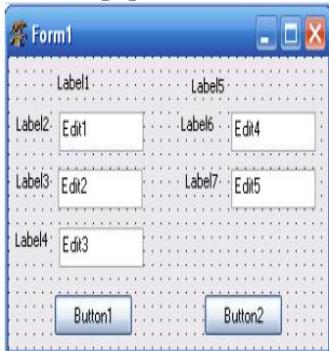
$$y1=(-b)/2a, x2 = \sqrt{|d|}/2a.$$

Тогда корни уравнения примут вид:

действительные – $x1 = y1 + y2, x2 = y1 - y2;$

комплексные – $x1 = y1 + iy2, x2 = y1 - iy2.$

На форме необходимо расположить следующие компоненты:



Свойства выбранных компонент:

- **Form1** – Caption – Решение квадратного уравнения;
- **Label1** – Caption – Введите коэффициенты уравнения;
- **Label2** – Caption – Коэффициент а=;
- **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,x1,x2,y1,y2:real;
begin
a:=strtofloat(edit1.text);
b:=strtofloat(edit2.text);
c:=strtofloat(edit3.text);
d:=sqr(b)-4*a*c;
y1:=-b/(2*a);
y2:=sqrt(abs(d))/(2*a);
if d>=0
then
begin
x1:=y1+y2;
x2:=y1-y2;
edit4.text:=floattostr(x1);
edit5.text:=floattostr(x2);
end
else
begin
edit4.text:=floattostr(y1)+i'+floattostr(y2);
edit5.text:=floattostr(y1)-i'+floattostr(y2);
end;
end;

```

Процедуру для кнопки «*Очистить*» написать самостоятельно.

Задача 2.2. Вычисление значения функции, представленной графиком.

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

Определим переменные задачи:

входные переменные: $x: \text{real}$,

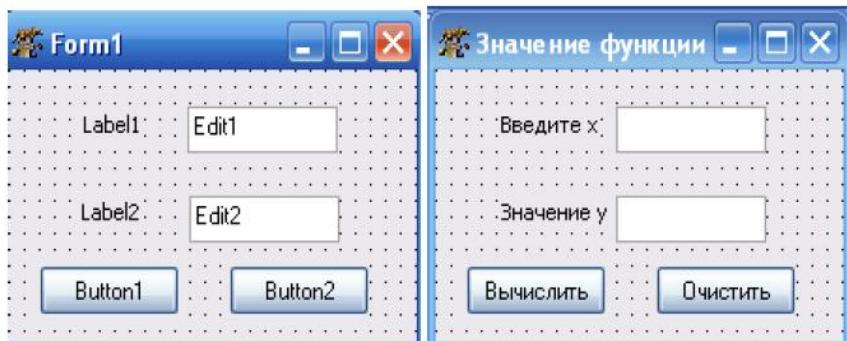
результат: $y: \text{real}$.

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

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

На форме необходимо расположить следующие компоненты со следующими свойствами:

- **Form1** – Caption – Значение функции;
- **Label1** – Caption – Введите x;
- **Label2** – Caption – Значение y;
- **Button1** – Caption – Вычислить;
- **Button2** – Caption – Очистить;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.



Процедура вычисления значения функции будет иметь вид:

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

Процедуру для кнопки «*Очистить*» написать самостоятельно.

Задача 2.3. Составить программу «Флажок» для вычисления квадрата и модуля числа.

Определим переменные задачи:

входные данные: x: real,

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

Для решения задачи расположим компоненты на форме.



Свойства выбранных компонент:

- **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»; щелчок мыши по компоненте «Квадрат».

```
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;

```

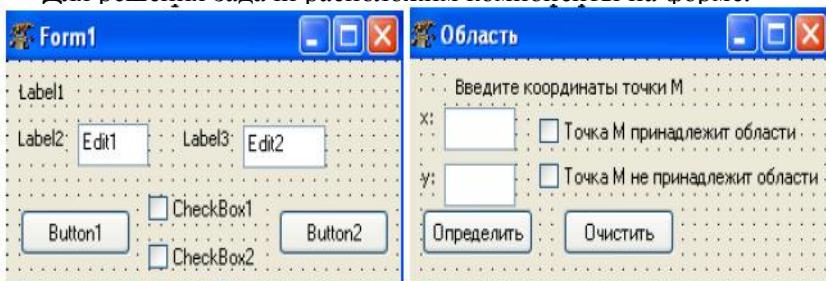
Задача 2.4. Решение задачи на определение принадлежности некоторой точки M с произвольными координатами (x,y) закрашенной области.

Определим переменные задачи:

входные данные: x, y: real,

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

Для решения задачи расположим компоненты на форме.



Свойства выбранных компонент:

- **Form1** – Caption – Область;
- **Label1** – Caption – Введите координаты точки M;
- **Label2** – Caption – x:
- **Label3** – Caption – y:
- **Button1** – Caption – Вычислить;
- **Button2** – Caption – Очистить;
- **CheckBox1** – Caption – Точка M принадлежит области;
- **CheckBox2** – Caption – Точка M не принадлежит области;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.

В программе будет происходить два события:

1. Щелчок мыши по компоненте «Вычислить».
2. Щелчок мыши по компоненте «Очистить».

```

procedure TForm1.Button1Click(Sender: TObject);
var x,y:real;
begin
x:=strtofloat(edit1.text);
y:=strtofloat(edit2.text);
if (x>=-1) and (x<=3) and (y>=-2) and (y<=4)
then
checkbox1.state:=cbchecked
else checkbox2.state:=cbchecked;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
edit1.clear;
edit2.clear;
edit1.setfocus;
checkbox1.state:=cbunchecked;
checkbox2.state:=cbunchecked;
end;
```

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Даны три целых числа. Возвести в квадрат отрицательные числа и в третью степень – положительные (число 0 не изменять).
2.	Из трех данных чисел выбрать наименьшее. Из трех данных чисел выбрать наибольшее.
3.	Из трех данных чисел выбрать наименьшее и наибольшее.
4.	Перераспределить значения переменных X и Y так, чтобы в X оказалось меньшее из этих значений, а в Y – большее.
5.	Значения переменных X, Y, Z поменять местами так, чтобы они оказались упорядоченными по возрастанию.

6.	Значения переменных X , Y , Z поменять местами так, чтобы они оказались упорядоченными по убыванию.
7.	Даны две переменные целого типа: A и B . Если их значения не равны, то присвоить каждой переменной сумму этих значений, а если равны, то присвоить переменным нулевые значения.
8.	Даны две переменные целого типа: A и B . Если их значения не равны, то присвоить каждой переменной максимальное из этих значений, а если равны, то присвоить переменным нулевые значения.
9.	Даны три переменные: X , Y , Z . Если их значения упорядочены по убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное.
10.	Даны три переменные: X , Y , Z . Если их значения упорядочены по возрастанию или убыванию, то удвоить их; в противном случае заменить значение каждой переменной на противоположное.
11.	Даны целочисленные координаты точки на плоскости. Если точка не лежит на координатных осях, то вывести 0. Если точка совпадает с началом координат, то вывести 1. Если точка не совпадает с началом координат, но лежит на оси OX или OY , то вывести соответственно 2 или 3.
12.	Даны вещественные координаты точки, не лежащей на координатных осях OX и OY . Вывести номер координатной четверти, в которой находится данная точка.
13.	На числовой оси расположены три точки: A , B , C . Определить, какая из двух последних точек (B или C) расположена ближе к A , и вывести эту точку и ее расстояние от точки A .
14.	Даны четыре целых числа, одно из которых отлично от трех других, равных между собой. Вывести порядковый номер этого числа.
15.	Дан номер некоторого года (положительное целое число). Вывести соответствующий ему номер столетия, учитывая, что, к примеру, началом 20 столетия был 1901 год.

16.	Дан номер некоторого года (положительное целое число). Вывести число дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный – 366 дней. Високосным считается год, делящийся на 4.
17.	Для данного x вычислить значение следующей функции f , вещественные значения: -1 если $x \leq 0$, $f(x) = x$ если $0 < x < 2$, 4 , если $x \geq 2$.
18.	Для данного x вычислить значение следующей функции f , принимающей значения целого типа: 0 , если $x < 0$, $f(x)$, если x принадлежит $(0,1) = (2,3), \dots$, -1 если x принадлежит $(1,2, 3,4)$.
19.	Дано целое число, лежащее в диапазоне от -999 до 999 . Вывести строку – словесное описание данного числа вида «отрицательное двузначное число», «нулевое число», «положительное однозначное число» и т. д.
20.	Дано целое число, лежащее в диапазоне от 1 до 9999 . Вывести строку – словесное описание числа вида «четное двузначное число», «нечетное четырехзначное число» и т. д.

Задание 3.2. Разработать программу в среде программирования Lazarus для вычисления значений функции $y = f(x)$ при произвольных значениях x . Получить результат работы программы для двух заданных значений x .

Вариант	Функция	Исходные данные
1.	$y = \begin{cases} b + 2 \ln x & \text{при } x \leq 3, \\ x^2 / (x^2 + a) & \text{при } x > 3 \end{cases}$	$a = 10,2; b = 13,4;$ 1) $x = 4,5; 2) x = 1,72$
2.	$y = \begin{cases} a + (1/2)e^{-x} & \text{при } x > 0, \\ \cos(bx + 1) & \text{при } x \leq 0 \end{cases}$	$a = 8,53; b = 17,1;$ 1) $x = 2,5; 2) x = -3,1$
3.	$y = \begin{cases} 1/(a^2 + x^2) & \text{при } x \leq 1, \\ b \cdot \ln x & \text{при } x > 1 \end{cases}$	$a = 7,2; b = 5,7;$ 1) $x = 2,92;$ 2) $x = -3,57$
4.	$y = \begin{cases} (a + x^2) / (b + \ln(x + 1)) & \text{при } x \leq 2, \\ e^x + x^2 & \text{при } x > 2 \end{cases}$	$a = 9,1; b = 3,6;$ 1) $x = 5,41; 2) x = 0,71$

5.	$y = \begin{cases} a \sin^2 x + \sqrt{x} & npu x \leq 1, \\ be^{x^2} & npu x > 1 \end{cases}$	a = 1,1; b = 3,2; 1) x = 4,3; 2) x = 0,93
6.	$y = \begin{cases} a \cdot \operatorname{tg}(x^2) & npu x \leq -1, \\ b + x^2 / (x^2 + a) & npu x > -1 \end{cases}$	a = 9,5; b = 3,8; 1) x = -4,52; 2) x = 1,83
7.	$y = \begin{cases} (a+x) \operatorname{arctg}(ax) & npu x > 3, \\ \cos^2(b+x^3) & npu x \leq 3 \end{cases}$	a = 4,1; b = 2,9; 1) x = 6,81; 2) x = 2,17
8.	$y = \begin{cases} \sin^3(a+x) & npu x < 5, \\ \ln \sqrt{ b-x } & npu x \geq 5 \end{cases}$	a = 1,9; b = 3,4; 1) x = 7,39; 2) x = 0,62
9.	$y = \begin{cases} \sqrt{1+x\sqrt{ax}} & npu x \geq 2, \\ \sin(bx) + 3 & npu x < 2 \end{cases}$	a = 4,6; b = 3,2; 1) x = 3,78; 2) x = 1,54
10.	$y = \begin{cases} \sqrt{e^{2x-b}} - 1 & npu x \leq 0, \\ 1/x^2 + a & npu x > 0 \end{cases}$	a = 6,7; b = 1,8; 1) x = -0,24; 2) x = 2,13
11.	$y = \begin{cases} \sqrt{a+ \sin x } & npu x > 4, \\ \operatorname{tg}(bx) & npu x \leq 4 \end{cases}$	a = 3,9; b = 4,8; 1) x = 5,17; 2) x = -2,35
12.	$y = \begin{cases} 2x^2 + a \cos(bx) & npu x \leq 1, \\ e^x + \operatorname{tg} x^3 & npu x > 1 \end{cases}$	a = 1,71; b = 0,83; 1) x = -2,16; 2) x = 3,37
13.	$y = \begin{cases} \ln(a+x^2) & npu x \geq 2, \\ e^{\sin x} + 2b & npu x < 2 \end{cases}$	a = 5,9; b = 6,1; 1) x = 6,72; 2) x = 1,23
14.	$y = \begin{cases} 0,2x^3 + a & npu x > -1, \\ bx^2 + \ln x+3 & npu x \leq -1 \end{cases}$	a = 2,9; b = 1,6; 1) x = 3,18; 2) x = -1,17
15.	$y = \begin{cases} \sin(x+a^2) & npu x < 2, \\ \ln(x^2 + 2x + b) & npu x \geq 2 \end{cases}$	a = 1,39; b = 2,76; 1) x = 3,68; 2) x = 0,91
16.	$y = \begin{cases} a - b^2 x & npu x \leq -3, \\ 1/(x^2 + e^{bx}) & npu x > -3 \end{cases}$	a = 7,5; b = 1,4; 1) x = -4,13; 2) x = 0,77
17.	$y = \begin{cases} \sqrt{ \sin ax } & npu x < -1, \\ \ln \sqrt{1+(bx)^2} & npu x \geq -1 \end{cases}$	a = 1,57; b = 2,38; 1) x = -0,1; 2) x = -4,25

18.	$y = \begin{cases} \sqrt{(a+x)^3} & \text{при } x \geq 1, \\ e^{bx-2} & \text{при } x < 1 \end{cases}$	$a = 4,92; b = 5,18;$ 1) $x = 5,13;$ 2) $x = -1,32$
19.	$y = \begin{cases} \sqrt{2 x + \cos^2 x} & \text{при } x \leq 6, \\ b \sin^3(ax) & \text{при } x > 6 \end{cases}$	$a = 4,49; b = 5,18;$ 1) $x = 4,41; 2) x = 7,69$
20.	$y = \begin{cases} \sqrt{2+ x } + \cos(b+x) & \text{при } x \leq -3, \\ a \sin(x^2) & \text{при } x > -3 \end{cases}$	$a = 1,89; b = 2,7;$ 1) $x = -2,37;$ 2) $x = -5,72$

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Когда применяется алгоритм разветвляющейся структуры?
2. Какие виды оператора IF существуют?
3. Когда применяется полная форма оператора IF?
4. Каков формат полной формы оператора IF? Его блок-схема?
5. Опишите процесс выполнения условного оператора полной формы.
6. Когда применяется сокращенная форма оператора IF? Его блок-схема?

7. Опишите процесс выполнения условного оператора сокращенной формы.

8. Дайте определение и приведите пример вложенного условного оператора.

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

5. ПРОГРАММИРОВАНИЕ ЗАДАЧ МНОЖЕСТВЕННОГО ВЫБОРА

Цель: освоить написание и отладку программ задач множественного выбора в интегрированной среде разработки программного обеспечения Lazarus.

Содержание:

- 1. Краткая теория.**
- 2. Методика и порядок выполнения работы.**
- 3. Индивидуальные задания.**
- 4. Содержание отчета и его форма.**
- 5. Контрольные вопросы и защита работы.**

1. Краткая теория

Инструкция множественного выбора *switch* позволяет выполнять различные части программы в зависимости от того, какое значение будет иметь некоторая целочисленная переменной (её называют «переменной-переключателем», а «switch» с английского переводится как раз как «переключатель»).

Для выполнения множественного выбора используются переключатели. Lazarus для работы с переключателями предлагает следующие компоненты:

- переключатель с независимой фиксацией (*CheckBox*), флажок этой компоненты можно переключать щелчком мыши;
- переключатели с зависимой фиксацией – *RadioButton* (кнопки выбора), *RadioGroup* (группа переключателей *RadioButton*).

Если в группе зависимых переключателей выбран один, то в отличие от независимого переключателя, его состояние нельзя изменить повторным щелчком. Для отмены выбора зависимого переключателя нужно выбрать другой переключатель из этой группы.

2. Методика и порядок выполнения работы

Задача 2.1. Создание приложения «Калькулятор»

Создать приложение (Калькулятор), обеспечивающее ввод двух целых чисел и выполнение над ними арифметических операций: сложения, вычитания, умножения и вещественного деления.

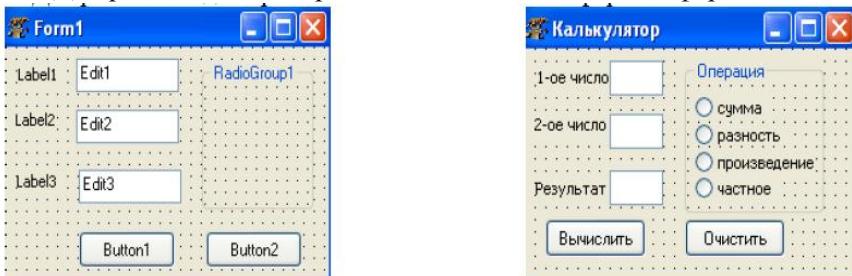
Для выбора операции используется набор переключателей.
Вывести сообщение об ошибке при вводе делителя, равного нулю.

Определим переменные задачи:

входные данные: a, b: integer

результат: c: real

Для решения задачи расположим компоненты на форме.



Свойства выбранных компонент:

- **Form1** – Caption – Калькулятор;
- **RadioGroup1** – Name – RG1
- Caption – Операция
- Items – сумма, разность, произведение, частное;
- **Label1** – Caption – 1-ое число;
- **Label2** – Caption – 2-ое число;
- **Label3** – Caption – Результат;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.
- **Edit3** – Text – пусто.
- **Button1** – Caption – Вычислить;
- **Button2** – Caption – Очистить;

Процедура для кнопки «Вычислить» будет иметь вид:

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b:integer;
c:real;
begin a:=strtoint(edit1.text);
b:=strtoint(edit2.text);
```

```

case RadioGroup1.ItemIndex of 0:c:=a+b;
1:c:=a-b;
2:c:=a+b;
3: if b=0
then showmessage('На ноль делить нельзя!')
else c:=a/b;
end;
edit3.text:=floattostr(c);
end;

```

Самостоятельно записать процедуру для кнопки «*Очистить*».

Задача 2.2. Решение задач на ввод и выбор данных из предложенных списков.

Имеется список периодических изданий.

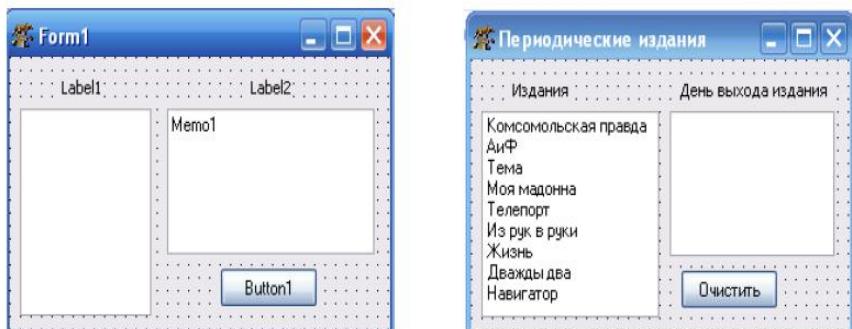
Для выбранного издания вывести день недели, в который он выходит.

Вывод дня недели осуществить в многострочное окно ввода-вывода.

Определим переменные задачи:

промежуточная переменная: n (индекс выбранной строки):
integer

Для решения задачи расположим компоненты на форме.



Свойства выбранных компонент:

- **ListBox1** – Items – Комсомольская правда, АиФ, Тема, Моя мадонна, Телепорт, Из рук в руки, Жизнь, Дважды два, Навигатор;
- **Label1** – Caption – Издания;
- **Label2** – Caption – Дни выхода издания;

- **Memo1** – Lines – пусто;
- **Button1** – Caption – Очистить;
- **Form1** – Caption – Периодические издания.

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

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

Самостоятельно записать процедуру для кнопки «Очистить».

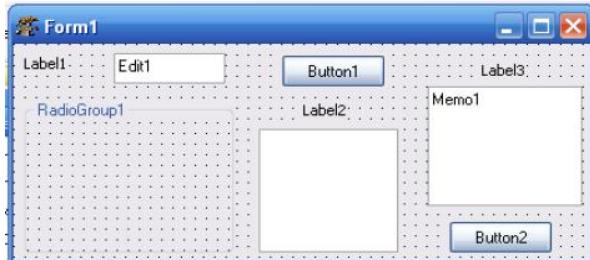
Задача 2.3. Решение задачи на ввод и выбор данных из предложенных списков.

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

Определим переменные задачи:

промежуточная переменная: n (индекс выбранной строки):
integer

Для решения задачи расположим компоненты на форме.



Свойства выбранных компонент:

- **ListBox11** – Items – Комсомольская правда, АиФ, Тема, Моя ма-
донна, Телепорт, Из рук в руки, Жизнь, Дважды два, Навигатор;
- **Form1** – Caption – Фильмы;
- **RadioGroup1** – Items –
Зея, Октябрь, Благовещенск, Харбин, Амур, Восток, Кактус;
- **Label1** – Caption – Введите номер дня недели;
- **Label2** – Caption – Список;
- **Label3** – Caption – Время сеанса;
- **Button1** – Caption – Определить;
- **Button2** – Caption – Очистить;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.
- **Memo1** – Lines – пусто;

Процедуры для определения кинотеатра и для определения
времени сеанса будут 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 else showmessage ('Введите номер
дня недели от 1 до 7');
end;
end;

procedure TForm1.ListBox1Click(Sender: TObject);
begin
case ListBox1.ItemIndex of
0: edit2.text:='12-00, 16-00, 20-00';
1: edit2.text:='12-30, 16-30, 20-30';
2: edit2.text:='11-00, 15-00, 21-00';
3: edit2.text:='11-30, 15-30, 21-30';
4: edit2.text:='12-00, 15-00, 22-00';
5: edit2.text:='11-00, 17-00, 21-00';
6: edit2.text:='12-00, 15-00, 21-30';
end;
end;

```

Самостоятельно записать процедуру для кнопки «*Очистить*».

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Напишите программу, которая получает номер месяца и выводит его название и время года.
2.	Напишите программу, которая получает день и номер месяца и определяет дату следующего дня. Считайте, что год невисокосный (365 дней).
3.	Напишите программу, которая получает день и номер месяца, а выводит количество дней, оставшихся до Нового Года. Считайте, что год невисокосный (365 дней). Учтите, что слово «дней» может иметь также формы «день» и «дня».
4.	Цикл с условием. Напишите программу, которая получает два целых числа A и B ($0 < A < B$) и выводит квадраты всех натуральных чисел в интервале от A до B.
5.	Дан номер месяца (1 – январь, 2 – февраль, ...). Вывести название соответствующего времени года ("зима", "весна" и т. д.).

6.	Дан номер месяца (1 – январь, 2 – февраль, ...). Вывести число дней в этом месяце для невисокосного года.
7.	Дано целое число в диапазоне 0 – 9. Вывести строку – название соответствующей цифры на русском языке (0 – "ноль", 1 – "один", 2 – "два", ...).
8.	Дано целое число в диапазоне 1 – 5. Вывести строку – словесное описание соответствующей оценки (1 – "плохо", 2 – "неудовлетворительно", 3 – "удовлетворительно", 4 – "хорошо", 5 – "отлично").
9.	Арифметические действия над числами пронумерованы следующим образом: 1 – сложение, 2 – вычитание, 3 – умножение, 4 – деление. Дан номер действия и два числа А и В (В не равно нулю). Выполнить над числами указанное действие и вывести результат.
10.	Единицы длины пронумерованы следующим образом: 1 – дециметр, 2 – километр, 3 – метр, 4 – миллиметр, 5 – сантиметр. Дан номер единицы длины и длина отрезка L в этих единицах (вещественное число). Вывести длину данного отрезка в метрах.
11.	Единицы массы пронумерованы следующим образом: 1 – килограмм, 2 – миллиграмм, 3 – грамм, 4 – тонна, 5 – центнер. Дан номер единицы массы и масса тела M в этих единицах (вещественное число). Вывести массу данного тела в килограммах.
12.	Робот может перемещаться в четырех направлениях ("С" – север, "З" – запад, "Ю" – юг, "В" – восток) и принимать три цифровые команды: 0 – продолжать движение, 1 – поворот налево, -1 – поворот направо. Дан символ С – исходное направление робота и число N – посланная ему команда. Вывести направление робота после выполнения полученной команды.
13.	Локатор ориентирован на одну из сторон света ("С" – север, "З" – запад, "Ю" – юг, "В" – восток) и может принимать три цифровые команды: 1 – поворот налево, 1 – поворот направо, 2 – поворот на 180 градусов. Дан символ С – исходная ориентация локатора и числа N1 и N2 – две посланные ему команды. Вывести ориентацию локатора после выполнения данных команд.

14.	Элементы окружности пронумерованы следующим образом: 1 – радиус (R), 2 – диаметр (D), 3 – длина (L), 4 – площадь круга (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данной окружности (в том же порядке). В качестве значения Pi использовать 3.14.
15.	Элементы равнобедренного прямоугольного треугольника пронумерованы следующим образом: 1 – катет (a), 2 – гипотенуза (c), 3 – высота, опущенная на гипотенузу (h), 4 – площадь (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).
16.	Элементы равностороннего треугольника пронумерованы следующим образом: 1 – сторона (a), 2 – радиус вписанной ппжсфмнqrh (R1), 3 – радиус описанной окружности (R2), 4 – площадь (S). Дан номер одного из этих элементов и его значение. Вывести значения остальных элементов данного треугольника (в том же порядке).
17.	Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, предшествующей указанной.
18.	Даны два целых числа: D (день) и M (месяц), определяющие правильную дату невисокосного года. Вывести значения D и M для даты, следующей за указанной.
19.	Дано целое число в диапазоне 20 – 69, определяющее возраст (в годах). Вывести строку – словесное описание указанного возраста, обеспечив правильное согласование числа со словом "год", например: 20 – "двадцать лет", 32 – "тридцать два года", 41 – "сорок один год".
20.	Дано целое число в диапазоне 100 – 999. Вывести строку – словесное описание данного числа, например: 256 – "двести пятьдесят шесть", 814 – "восемьсот четырнадцать".

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. С помощью каких операторов можно организовать многовариантное ветвление? (IF...THEN...ELSE, CASE...OF). Ставится ли перед ELSE ;?
2. Может ли отсутствовать ELSE в операторе выбора?
3. Как «работает» оператор выбора?
4. В каких задачах используются разветвляющиеся алгоритмы?
5. Чем линейный алгоритм отличается от разветвленного?
6. Что такое условие разветвления?
7. Изобразите блок-символ операции условного перехода.
8. Что такое множественный выбор?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

6. ПРОГРАММИРОВАНИЕ ЗАДАЧ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Цель: освоить написание и отладку программ задач циклической структуры в интегрированной среде разработки ПО Lazarus.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория

Циклический алгоритм – это такой алгоритм, в котором некоторая группа действий выполняется некоторое число раз.

Циклом в программирование называют повторение одних и тех же действий (шагов).

Переменная, изменяющаяся с некоторым шагом, называется *переменной цикла*. Переменная цикла определяется своим *начальным значением, конечным значением и шагом изменения*.

Например, если переменная цикла изменяется следующим образом:

$i = 10, 20, 30, \dots, 90$, тогда она однозначно определяется своим начальным значением $in = 10$, конечным значением $ik = 90$ и шагом $di = 10$.

Последовательность действий, которые повторяются в цикле, называют *телом цикла*. Количество повторений цикла определяется условием окончания цикла. Если условие окончания цикла расположено перед телом цикла, то такой цикл называется *циклом с предусловием*. Если условие окончания цикла расположено после тела цикла, то такой цикл называется *циклом с постусловием*.

При написании условных циклических алгоритмов следует помнить следующее. Во-первых, чтобы цикл имел шанс когда-нибудь закончиться, содержимое его тела должно обязательно влиять на условие цикла. Во-вторых, условие должно состоять из корректных выражений и значений, определенных еще до первого выполнения тела цикла.

Существует 3 типа алгоритмов *циклической структуры*:

1. WHILE – цикл с предусловием;
2. REPEAT – цикл с постусловием;
3. FOR – цикл со счетчиком (с параметром или с заданным числом повторений).

Если число повторений оператора заранее неизвестно, а задано лишь условие его повторения (или окончания), используются операторы *while*, *repeat*. Оператор *for* используется, если число повторений заранее известно.

Для всех операторов цикла характерны следующие особенности:

- повторяющиеся вычисления записываются только один раз;
- вход в цикл возможен только через его начало;
- возможен досрочный выход из цикла.

Оператор цикла с предусловием. Синтаксис (формат) оператора:

WHILE <условие> DO <тело цикла>;

где: *while*, *do* – ключевые слова (перев. с англ. *пока* и *делать*);

<условие> – логическое выражение типа сравнения, которое может принимать значение *True* или *False*, используемое для выхода из цикла.

<тело цикла> – простой или составной оператор.

Выполнение оператора цикла с предусловием

1. Перед каждым выполнением тела цикла вычисляется значение логического выражения.
2. Если выражение принимает значение *True*, то выполняется тело цикла, затем происходит переход на начало цикла.
3. Если значение выражения равно *False*, происходит выход из цикла и переход к оператору программы, следующему за оператором цикла.
4. Если перед первым выполнением цикла значение выражения равно *False*, тело цикла не выполняется ни разу и происходит переход к оператору, следующему за оператором цикла.

В теле цикла обязательно должен быть оператор, изменяющий значение переменной цикла, которая определяет условие выхода из цикла, иначе цикл получится бесконечным, например:

While n = 1 Do Write ('Бесконечный цикл');

Очевидно, что результат выражения $n=1$ всегда равен *True*, так как значение переменной цикла n не изменяется в теле цикла, состоящем из оператора *Write*, поэтому цикл будет выполняться бесконечно.

Оператор цикла с постусловием. Синтаксис (формат) оператора:

REPEAT <тело цикла> UNTIL <условие>;

где: *repeat*, *until* – ключевые слова (англ. *повторять, до тех пор пока*);

<тело цикла> – любые операторы;

<условие> – логическое выражение типа сравнения, которое может принимать значение *True* или *False*, используемое для выхода из цикла.

Выполнение оператора цикла с постусловием

1. Выполняется тело цикла.
2. Вычисляется значение логического выражения.
3. Если значение выражения равно *False*, то тело цикла выполняется еще раз.
4. Если значение выражения равно *True*, то происходит выход из цикла и переход к оператору, следующему за оператором цикла.

Отличительные особенности оператора цикла *Repeat*:

- оператор цикла с постусловием выполняется, по крайней мере, один раз, независимо от значения выражения;
- тело цикла выполняется, пока значение выражения равно *False*;
- в теле может находиться произвольное количество операторов без операторных скобок *Begin ... End*.

По крайней мере, один из операторов тела цикла должен влиять на значение выражения в условии окончания цикла, иначе цикл будет выполняться бесконечно. Так как условие окончания цикла про-

веряется после выполнения тела цикла, то тело цикла выполняется, по крайней мере, один раз. Поэтому важно четко проследить правильность входления в цикл и выхода из него. При неправильной организации выхода может возникнуть ошибка переполнения, т. к. наращивание какой-либо величины в теле цикла может продолжаться до бесконечности.

Оператор цикла со счетчиком

В случаях, когда число повторений может быть заранее известно, для организации циклической обработки информации применяется оператор цикла со счетчиком FOR. Часто этот оператор называют оператором цикла с параметром, так как число повторений задается переменной, называемой параметром цикла, или управляющей переменной.

Синтаксис оператора цикла со счетчиком следующий:

1. **FOR** *<i - параметр цикла>:= n1 TO n2 DO <оператор>;*

2. **FOR** *<i - параметр цикла>:= n1 DOWNTO n2 DO <оператор>;*

где: *for, to (downto), do* – ключевые слова (перев. с англ. *для, к, выполнить*соответственно);

<i - параметр цикла> – это переменная, которая изменяется внутри цикла по определенному закону и влияет на его окончание;

n1 и *n2* – начальное и конечное значения параметра цикла.

В первом случае *n1<n2*, шаг изменения параметра цикла равен 1. Во втором случае *n1>n2*, шаг изменения параметра цикла равен (-1);

FOR.. .DO – заголовок цикла;

<оператор> – тело цикла.

Тело цикла может быть простым или составным оператором. Оператор *FOR* обеспечивает выполнение тела цикла до тех пор, пока не будут перебраны все значения параметра цикла от начального до конечного.

Заголовок оператора повтора *FOR* определяет:

- диапазон изменения значений управляющей переменной (параметра цикла) и одновременно число повторений оператора, содержащегося в теле цикла;
- направление изменения значения параметра цикла (возрастание на 1 – *TO* или убывание на (-1) – *DOWNTO*).

Выполнение оператора цикла с параметром

1. Перед первым выполнением тела цикла вычисляется значение параметров цикла $n1$ и $n2$ (в общем случае они могут быть заданы арифметическими выражениями).
2. Параметру цикла присваивается значение выражения $n1$.
3. Если полученное значение параметра цикла не превышает конечного значения $n2$, то выполняется тело цикла.
4. Выполняется возврат на начало оператора цикла.
5. Автоматически значение параметра цикла увеличивается на величину шага.
6. Если полученное значение параметра цикла не превышает конечного значения $n2$, то еще раз выполняется тело цикла.
7. Если полученное значение параметра цикла превысило конечное значение $n2$, то выполняется переход к оператору, следующему за оператором цикла.
8. Если перед первым выполнением оператора, параметр цикла превышает конечное значение $n2$, то тело цикла не выполняется ни разу и происходит переход к следующему оператору программы.

На использование параметра цикла *FOR* налагаются следующие ограничения:

- в качестве параметра должна использоваться простая переменная, описанная в текущем блоке (локальная);
- управляющая переменная должна иметь дискретный тип (полярный);
- начальные и конечные значения диапазона должны иметь тип, совместимый с типом управляющей переменной. При этом допустим любой простой тип, кроме вещественного;
- в теле цикла запрещается явное изменение значения управляющей переменной (например, оператором присваивания).

После нормального завершения оператора цикла значение параметра цикла равно конечному значению. Если оператор цикла не выполнялся ни разу, значение параметра цикла не определено.

Не допускается изменение параметра цикла в теле цикла, так как он изменяется автоматически только на величину ± 1 . Однако это не является большим недостатком, т. к. произвольный шаг изменения переменной цикла можно задать при использовании операторов цикла с предусловием или с постусловием.

2. Методика выполнения работы

Задача 2.1. Вычисление значения факториала. Найти значение $n!$.

Определим переменные задачи:

Входные данные: n: integer.

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

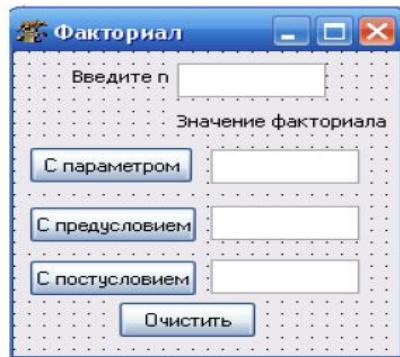
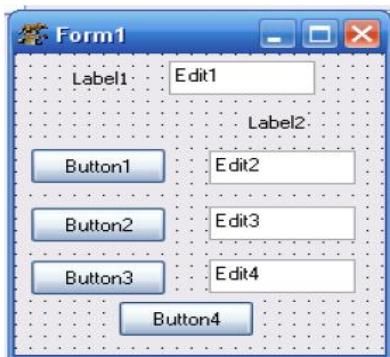
Результат: p: real.

Задачу решим с использованием всех трех циклических структур.

Расположим компоненты на форме.

Свойства выбранных компонент:

- **Form1** – Caption – Факториал;
- **Label1** – Caption – Введите n;
- **Label2** – Caption – Значение факториала;
- **Button1** – Caption – С параметром;
- **Button2** – Caption – С предусловием;
- **Button3** – Caption – С постусловием;
- **Button4** – Caption – Очистить;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.
- **Edit3** – Text – пусто.
- **Edit4** – Text – пусто.

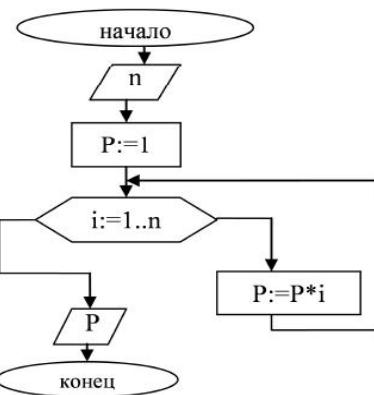


Запомните! При решении задач на циклические структуры первоначальное значение суммы S=0, а произведения P=1.

Цикл с параметром

procedure

```
TForm1.Button1Click(Sender: TObject);
var i,n:integer;
p:real;
begin
n:=strtoint(edit1.text);
P:=1;
for i:=1 to n do p:=p*i;
edit2.text:=floattostr(P);
end;
```



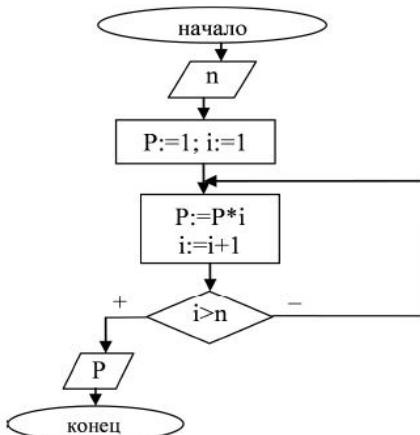
Цикл с предусловием

```
procedure TForm1.Button2Click(Sender: TObject);
var n,i:integer;
p:real;
begin
n:=strtoint(edit1.text); p:=1; i:=1;
while i<=n do
begin
p:=p*i; i:=i+1; end;
edit3.text:=floattostr(p);
end;
```

```

Цикл с постусловием
procedure
TForm1.Button3Click(Sender:
 TObject);
var n,i:integer;
p:real;
begin
n:=strtoint(edit1.text);
p:=1;
i:=1;
repeat p:=p*i;
i:=i+1;
until i>n;
edit4.text:=floattostr(p);
end;

```



Процедуру для кнопки «Очистить» написать самостоятельно.

Задача 2.2. Нахождение суммы ряда.

Составить программу, находящую сумму ряда $\sum \sin(i)$, $i=1, n$.

Условие задачи определяет ряд, вида $\sin(1) + \sin(2) + \sin(3) + \dots + \sin(n)$.

Обозначим отдельный элемент переменной a .

Можно сказать, что каждый элемент $a = \sin(i)$, где $i=1, 2, 3, \dots, n$.

Определим переменные задачи:

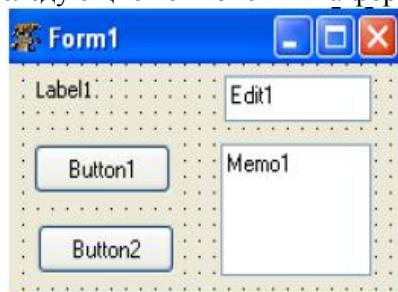
Входные данные: $n: integer$.

Промежуточные переменные: $i: integer$; $a: real$.

Результат: $S: real$.

Решение данной задачи рассмотрим для цикла с постусловием.

Расположим следующие компоненты на форме.



Свойства выбранных компонент:

- **Form1** – Caption – Сумма ряда;
- **Label1** – Caption – Введите n;
- **Button1** – Caption – Вычислить;
- **Button2** – Caption – Очистить;
- **Edit1** – Text – пусто.
- **Memo1** – Lines – пусто.

Процедура для кнопки «Вычислить»:

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

Процедуру для кнопки «Очистить» написать самостоятельно.

Задача 2.3. Нахождение произведения ряда.

Составить программу для нахождения произведения ряда $\prod (3+x^i)$, $i=1, n$. Условие задачи определяет ряд, вида $(3+x) * (3+x^2) * (3+x^3) * \dots * (3+x^n)$.

Как и в примере 3 обозначим отдельный элемент переменной a , где $a = 3+x^i$, где $i = 1, 2, 3, \dots, n$. Обозначим $t = x^i$. Тогда $a=3+t$. Первоначально $t=1$, а далее в цикле каждый раз увеличивается на степень $t = t * x$.

Определим переменные задачи:

Входные данные: n: integer; x: real.

Промежуточные переменные: i: integer; a, t: real.

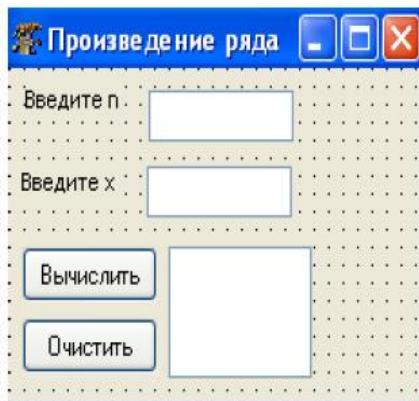
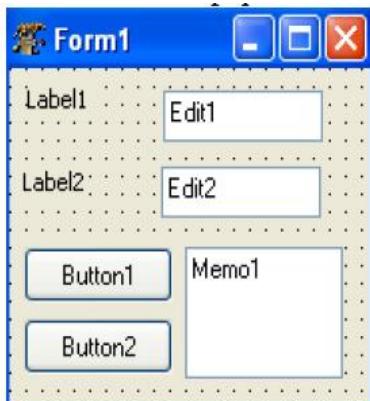
Результат: P: real.

Решение данной задачи рассмотрим на примере цикла с предусловием.

Расположим компоненты на форме.

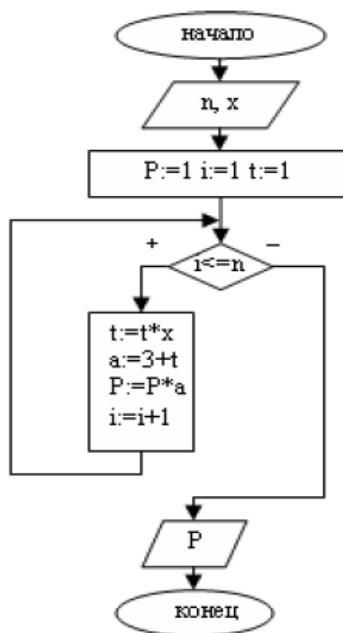
Свойства выбранных компонент:

- **Form1** – Caption – Произведение ряда;
- **Label1** – Caption – Введите n;
- **Label2** – Caption – Введите x;
- **Button1** – Caption – Вычислить;
- **Button2** – Caption – Очистить;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.
- **Memo1** – Lines – пусто.



Процедура для кнопки «Вычислить»:

```
procedure
TForm1.Button1Click();
var n, i, x:integer;
a, t, P:real;
begin
n:=StrToInt (edit1.text);
x:=StrToInt (edit2.text);
P:=1;
i:=1;
t:=1;
while i<=n do
begin
t:=t*x;
a:=3+t;
P:=P*a;
i:=i+1;
end;
Memo1.Lines.Add (FloatTo-
Str(P));
end;
```



Процедуру для кнопки «Очистить» написать самостоятельно.

Компонент *Memo* часто используется, когда необходимо вывести значения функции на некотором интервале с заданным шагом. Задачи такого типа называются задачами на табулирование, т. е. при решении таких задач результаты выводятся на каждом шаге.

Задача 2.4. Нахождение значения функции на интервале с пошаговым выводом результата.

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

Определим переменные задачи:

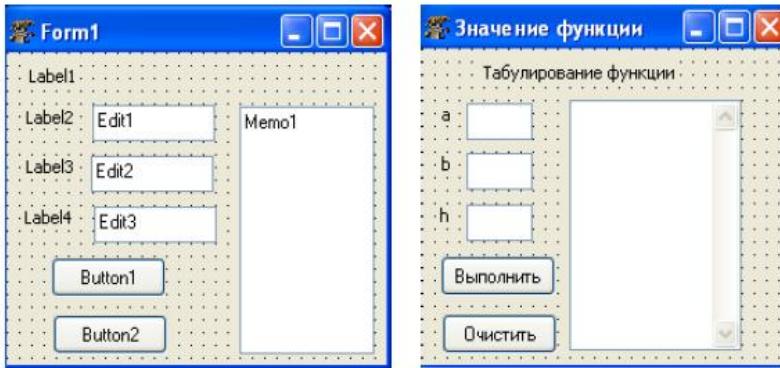
входные данные: a, b, h: real

промежуточные переменные: x: real

результат: y: real

Решение данной задачи рассмотрим на примере цикла с предусловием.

Расположим компоненты на форме.



Свойства компонент:

- **Form1** – Caption – Значение функции;
- **Label1** – Caption – Табулирование функции;
- **Label2** – Caption – a;
- **Label3** – Caption – b;
- **Label4** – Caption – h;
- **Button1** – Caption – Выполнить;
- **Button2** – Caption – Очистить;
- **Memo1** – Lines – пусто.

Процедура для кнопки «Вычислить»:

```
procedure TForm1.Button1Click();
var a,b,h,y,x:real;
begin
  a:=StrToFloat(Edit1.Text);
  b:=StrToFloat(Edit2.Text);
  h:=StrToFloat(Edit3.Text);
  x:=a;
  while x<=b do
  begin
    y:=exp(x)-exp(-x)-2;
    Memo1.Lines.Add('x=' + FloatToStr(x) + ' y=' + FloatToStr(y));
    x:=x+h;
  end;
end;
```

Процедуру для кнопки «Очистить» написать самостоятельно.

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Даны два целых числа A и B ($A < B$). Вывести все целые числа, расположенные между данными числами (включая сами эти числа), в порядке их возрастания, а также количество N этих чисел.
2.	Даны два целых числа A и B ($A < B$). Вывести все целые числа, расположенные между данными числами (не включая сами эти числа), в порядке их убывания, а также количество N этих чисел.
3.	Дано вещественное число A и целое число N (> 0). Вывести A в степени N: $AN = A \cdot A \cdot \dots \cdot A$ (числа A перемножаются N раз).
4.	Дано вещественное число A и целое число N (> 0). Вывести все целые степени числа A от 1 до N.
5.	Дано вещественное число A и целое число N (> 0). Вывести $1 + A + A^2 + A^3 + \dots + A^N$.
6.	Дано вещественное число A и целое число N (> 0). Вывести $1 - A + A^2 - A^3 + \dots + (-1)^N A^N$.
7.	Дано целое число N (> 1). Вывести наименьшее целое K, при котором выполняется неравенство $3K > N$, и само значение 3K.
8.	Дано целое число N (> 1). Вывести наибольшее целое K, при котором выполняется неравенство $3K < N$, и само значение 3K.
9.	Дано вещественное число A (> 1). Вывести наименьшее из целых чисел N, для которых сумма $1 + 1/2 + \dots + 1/N$ будет больше A, и саму эту сумму.
10.	Дано вещественное число A (> 1). Вывести наибольшее из целых чисел N, для которых сумма $1 + 1/2 + \dots + 1/N$ будет меньше A, и саму эту сумму.

11.	Дано целое число $N (> 0)$. Вывести произведение $1 \cdot 2 \cdot \dots \cdot N$. Чтобы избежать целочисленного переполнения, вычислять это произведение с помощью вещественной переменной и выводить его как вещественное число.
12.	Дано целое число $N (> 0)$. Если N — нечетное, то вывести произведение $1 \cdot 3 \cdot \dots \cdot N$; если N — четное, то вывести произведение $2 \cdot 4 \cdot \dots \cdot N$. Чтобы избежать целочисленного переполнения, вычислять это произведение с помощью вещественной переменной и выводить его как вещественное число.
13.	Дано целое число $N (> 0)$. Вывести сумму $2 + 1/(2!) + 1/(3!) + \dots + 1/(N!)$ (выражение $N!$ — "N факториал" — обозначает произведение всех целых чисел от 1 до N : $N! = 1 \cdot 2 \cdot \dots \cdot N$). Полученное число является приближенным значением константы $e = \exp(1) (= 2.71828183\dots)$.
14.	Дано вещественное число X и целое число $N (> 0)$. Вывести $1 + X + X^2/2! + \dots + X^N/N!$ ($N! = 1 \cdot 2 \cdot \dots \cdot N$). Полученное число является приближенным значением функции \exp в точке X .
15.	Дано вещественное число X и целое число $N (> 0)$. Вывести $X - X^3/3! + X^5/5! - \dots + (-1)^N X^{2N+1}/(2N+1)!$ ($N! = 1 \cdot 2 \cdot \dots \cdot N$). Полученное число является приближенным значением функции \sin в точке X .
16.	Дано вещественное число X и целое число $N (> 0)$. Вывести $1 - X^2/2! + X^4/4! - \dots + (-1)^N X^{2N}/(2N)!$ ($N! = 1 \cdot 2 \cdot \dots \cdot N$). Полученное число является приближенным значением функции \cos в точке X .
17.	Дано вещественное число $X (X < 1)$ и целое число $N (> 0)$. Вывести $X - X^2/2 + X^3/3 - \dots + (-1)^{N-1} X^N/N$. Полученное число является приближенным значением функции \ln в точке $1+X$.
18.	Дано вещественное число $X (X < 1)$ и целое число $N (> 0)$. Вывести $X - X^3/3 + X^5/5 - \dots + (-1)^N X^{2N+1}/(2N+1)$. Полученное число является приближенным значением функции \arctg в точке X .
19.	Дано целое число $N (> 2)$ и две вещественные точки на числовой оси: $A, B (A < B)$. Отрезок $[A, B]$ разбит на равные отрезки длины H с концами в N точках вида $A, A+H, A+2H, A+3H, \dots, B$. Вывести значение H и набор из N точек, образующий разбиение отрезка $[A, B]$.

20.	Дано целое число N (> 2) и две вещественные точки на числовой оси: A, B (A < B). Функция F(X) задана формулой F(X) = 1 - sin(X). Вывести значения функции F в N равноотстоящих точках, образующих разбиение отрезка [A, B]: F(A), F(A + H), F(A + 2H), ..., F(B).
21.	Дано число D (> 0). Последовательность чисел AN определяется следующим образом: A1 = 2, AN = 2 + 1/AN-1, N = 2, 3, ... Найти первый из номеров K, для которых выполняется условие AK - AK-1 < D, и вывести этот номер, а также числа AK-1 и AK.
22.	Дано число D (> 0). Последовательность чисел AN определяется следующим образом: A1 = 1, A2 = 2, AN = (AN-2+ AN-1)/2, N = 3, 4, ... Найти первый из номеров K, для которых выполняется условие AK - AK-1 < D, и вывести этот номер, а также числа AK-1 и AK.

Задание 3.2. Создать в соответствии с вариантом задания программу циклической структуры. В программе должны быть использованы 4-5 визуальных компонентов. Поработать со свойствами компонент и формы.

Вариант	$y = f(x)$	Исходные данные
1.	$y = \frac{\sqrt{ax}}{b + ax\sqrt{x}}$	$x_1 = 1; x_n = 2;$ $\Delta x = 0,2; a = 3,5; b = 1,2$
2.	$y = \sin(ax) + 3 \cos^2(bx^2 + 1)$	$x_1 = 0; x_n = 5;$ $\Delta x = 0,5; a = 0,5; b = 0,7$
3.	$y = \frac{1 + a(x + b)}{3 + \cos(ax)}$	$x_1 = 1; x_n = 3;$ $\Delta x = 0,2; a = 3,9; b = 2,3$
4.	$y = bx\sqrt{1 + a^2 \ln x}$	$x_1 = 2; x_n = 3;$ $\Delta x = 0,1; a = 4; b = 7$
5.	$y = \frac{b \cos x}{1 + a^2 \sin^3 x}$	$x_1 = 1; x_n = 6;$ $\Delta x = 0,5; a = 0,57; b = 9$
6.	$y = a \left(\frac{b}{x} - \frac{\ln ax}{b^2} \right)$	$x_1 = 2; x_n = 5;$ $\Delta x = 0,5; a = 1,5; b = 4,8$
7.	$y = \sqrt{1 + ax + b \cos x}$	$x_1 = 2; x_n = 8;$ $\Delta x = 0,7; a = 4,2; b = 1,5$

8.	$y = ax(1 + ae^{-x})$	$x_1 = 2; x_n = 7;$ $\Delta x = 0,5; a = 3,5;$
9.	$y = b \ln(ax^2) + b \ln^2 x$	$x_1 = 1; x_n = 4;$ $\Delta x = 0,3; a = 4,3; b = 5,4$
10.	$y = \frac{\ln(ax^2 + b)}{ax + 1}$	$x_1 = 2; x_n = 4;$ $\Delta x = 0,4; a = 1,4; b = 2,5$
11.	$y = \frac{\cos(ax^2)}{1 + \tan^3(bx)}$	$x_1 = 0; x_n = 1;$ $\Delta x = 0,1; a = 2,1; b = 0,3$
12.	$y = a \ln \frac{x}{bx^2 + 2}$	$x_1 = 3; x_n = 6;$ $\Delta x = 0,3; a = 1,9; b = 1,1$
13.	$y = \sqrt{\frac{ax}{b + \cos^2 x}}$	$x_1 = 3; x_n = 5;$ $\Delta x = 0,2; a = 1,9; b = 1,1$
14.	$y = \sqrt{\frac{\ln(a^2)}{x+a} + \cos^2 a}$	$x_1 = 5; x_n = 8;$ $\Delta x = 0,2; a = 5,3;$
15.	$y = \frac{bx^2}{e^{ax} + x}$	$x_1 = 2; x_n = 8;$ $\Delta x = 0,6; a = 1,9; b = 1,1$
16.	$y = \ln(x + \sqrt{b \sin^2(bx) + 1})$	$x_1 = 1; x_n = 5; \Delta x = 0,4;$ $b = 5,7$
17.	$y = (1 - e^{-ax}) \cdot \ln \frac{ax^2 - 1}{ax^2 + 2}$	$x_1 = 4; x_n = 7;$ $\Delta x = 0,3; a = 3,8$
18.	$y = \frac{\sin(ax + e^x)}{\sqrt{ax^2 + 3}}$	$x_1 = 3; x_n = 9;$ $\Delta x = 0,6; a = 2,7$
19.	$y = \sqrt{(1 + ax) \ln(a + x)}$	$x_1 = 1; x_n = 5;$ $\Delta x = 0,4; a = 5,3$
20.	$y = \frac{x^2 + \cos(ax)}{\sqrt{ax + e^x}}$	$x_1 = 2; x_n = 4;$ $\Delta x = 0,1; a = 4,5$
21.	$y = \frac{3 \sin(ax)}{\sqrt{x^2 + 2}}$	$x_1 = 3; x_n = 5;$ $\Delta x = 0,1; a = 4,5$

22.	$y = \frac{\sqrt{\ln x + x}}{2 + ax}$	$x_1 = 1; x_n = 3;$ $\Delta x = 0,2; a = 2,8$
23.	$y = \frac{\cos \frac{b}{1+x^2}}{\ln x + 4}$	$x_1 = 3; x_n = 9;$ $\Delta x = 0,3; b = 0,71$
24.	$y = \sin \frac{1}{\sqrt{ax^2 + 2}} + e^x$	$x_1 = 0; x_n = 3;$ $\Delta x = 0,2; a = 3,9$
25.	$y = \frac{\ln(ax + 5)}{\sqrt{x + 2} + e^x}$	$x_1 = 5; x_n = 9;$ $\Delta x = 0,4; a = 2,4$
26.	$y = \sqrt{\frac{bx + 1}{a + \cos^2 bx}}$	$x_1 = 1; x_n = 3;$ $\Delta x = 0,2; a = 4,1; b = 4,7$
27.	$y = \frac{a^2 \sin x^2}{1 + a \sin^2 x}$	$x_1 = 0; x_n = 2;$ $\Delta x = 0,2; a = 1,92$
28.	$y = \frac{1}{\sin ax + 2} + \ln x$	$x_1 = 1; x_n = 4;$ $\Delta x = 0,3; a = 1,8$
29.	$y = ax^2 \left(1 + e^{-bx}\right)$	$x_1 = 1; x_n = 3;$ $\Delta x = 0,2; a = 0,8; b = 4,2$
30.	$y = \frac{e^{2x} - e^{bx}}{x^6 \left(1 + \sqrt{x}\right)}$	$x_1 = 1; x_n = 5;$ $\Delta x = 0,4; b = 0,37$

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;

- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Какой алгоритм называется циклическим?
2. Какими параметрами характеризуется переменная цикла?
3. Что такое тело цикла?
4. Где может располагаться условие окончания цикла?
5. Какие операторы используются для организации цикла?
6. Формат и блок-схема оператора WHILE, как он работает?
7. Формат и блок-схема оператора REPEAT, как он работает?
8. Каковы отличительные особенности оператора цикла с постусловием?
9. Различие между операторами REPEAT и WHILE.
10. В каких случаях получается бесконечный цикл?
11. В каком случае используется оператор FOR?
12. Какие ограничения накладываются на оператор FOR?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

7. ПРОГРАММИРОВАНИЕ ЗАДАЧ С ДАННЫМИ ТИПА ВЕКТОР И МАТРИЦА

Цель: освоить написание и отладку программ задач с данными типа вектор и матрица в интегрированной среде разработки ПО Lazarus.

Содержание:

- | | |
|----|---------------------------------------|
| 1. | Краткая теория. |
| 2. | Методика и порядок выполнения работы. |
| 3. | Индивидуальные задания. |
| 4. | Содержание отчета и его форма. |
| 5. | Контрольные вопросы и защита работы. |

1. Краткая теория

Наряду с простыми типами (*byte*, *integer*, *real*) в языке Pascal используются также структурированные данные, самыми простыми из которых является массивы.

Массивы удобно использовать для работы с множеством однотипных данных (целочисленными значениями, строками и датами). Например, можно создать массив для хранения списка студентов, обучающихся в одной группе. Вместо того, чтобы задавать переменные для каждого студента, например *Студент1*, *Студент2* и т. д., достаточно создать один массив, где каждой фамилии из списка будет присвоен порядковый номер.

Массив – структурированный тип данных, состоящий из фиксированного числа элементов одного типа, упорядоченных по номерам. Такого рода массив, представляющий собой список данных одного и того же типа, называют простым или одномерным.

Элементы массива располагаются в памяти ПК, т.е. они упорядочены по возрастанию порядковых номеров (индексов).

Для доступа к данным, хранящимся в определенном массиве, необходимо указать имя массива (идентификатор) и порядковый номер этого элемента.

Описание одномерного массива и обращение к элементам массива. Одномерные массивы имеют аналогию с таким понятием в математике, как вектор. Описание переменной типа массив задается следующим образом:

*имя_массива: ARRAY [тип индекса] OF
тип_элемента_массива;*

где: *array* и *of* – ключевые слова (перевод с англ. *массив, из*).

Для типа индексов обычно используют тип-диапазон, который определяет границы изменения индексов. Так как память выделяется под массив во время компиляции программы, то границы изменения индексов должны быть константами, или константными выражениями, т.е. они должны быть явно определены в программе. Чаще всего в реальных задачах индексы изменяются от 1, и в этом случае индекс элемента совпадает с его порядковым номером.

Элементами массива могут быть как простые переменные любых типов, так и переменные составных типов (массивов, строк, записей).

Ниже приведены примеры описания одномерных массивов:

var

f: array [1..10] of integer; массив *f* из 10 целых чисел

mas: array[0..100] of char; массив *mas* из 101 символов

b: array[-5..5] of real; массив *b* из 11 вещественных чисел

Размерность массива – величина произвольная, однако суммарная длина внутреннего представления любого массива не может быть больше 64 Кбайт.

Следует учесть, что резервирование объема памяти для массива выполняется при компиляции программы. Поэтому, если количество элементов массива заранее неизвестно или оно может изменяться, то необходимо объявлять массив максимально необходимой размерности.

Описать массив можно и в разделе *type*. Процедура объявления состоит в следующем: ввести новый тип данных и описать переменные нового типа:

type

имя_типа = array[тип_индекса] of тип_элемента_массива;

Тип_элемента_массива – это любой ранее определенный тип данных,

например:

type massiv = array[0..12] of real;

abc = array[-3..6] of integer;

var x, y: massiv; z: abc;

Для описания массива можно использовать предварительно определенные константы:

const n=10; m=12;

var a: array[1..n] of real; b: array[0..m] of byte;

Константы должны быть определены до использования, так как массив не может быть переменной длины.

Массив можно объявить и в разделе констант, явно перечислив его элементы:

Const

mas3: array [1.. 3] of integer = (2, 4, 8)

Обращение к массиву в целом осуществляется по его имени.

Для обращения к отдельному элементу массива указывается имя массива и индекс – целое число, следующее за именем массива в квадратных скобках:

f[1]:=0 1-й элемент массива *f*

mas[100]:='a' 100-й элемент массива *mas*

b[3]:=1.13 3-й элемент массива *b*

При обработке массивов возникают такие задачи, как ввод элементов массива, нахождение суммы, произведения, среднего и т. д., поиск некоторого элемента в массиве, сортировка элементов массива, вывод элементов массива.

Ввод-вывод элементов массива. Паскаль не имеет специальных средств ввода-вывода всего массива, поэтому необходимо последовательно вводить 1-й, 2-й, 3-й и т. д. его элементы и аналогичным образом поступить при выводе. Следовательно, для ввода-вывода необходимо организовать цикл, в котором практически все операции с массивами необходимо проводить поэлементно. Для формирования и обработки элементов массива удобно использовать циклы *for* и *while*. Кроме этого, массивы можно формировать с помощью датчика случайных чисел, используя процедуру *random*.

2. Методика и порядок выполнения работы

Задача 2.1. Программирование задач для обработки векторов.

Дан массив А, состоящий из элементов целого типа. Найти количество положительных элементов этого массива.

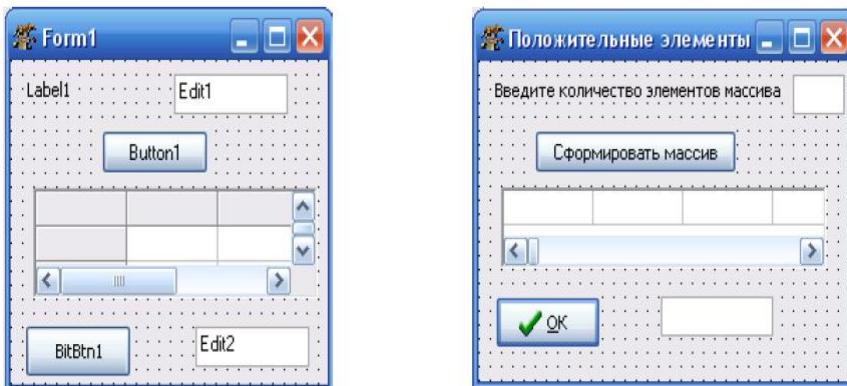
Определим переменные задачи:

Входные данные: а[i,0]: integer – элементы массива; n: integer – количество элементов массива.

Промежуточные переменные: i: integer – параметр цикла.

Результат: k: integer – количество положительных элементов массива.

Расположим следующие компоненты на форме.



Свойства выбранных компонент:

- **Form1** – Caption – Положительные элементы;
- **Label1** – Caption – Введите количество элементов массива;
- **Button1** – Caption – Сформировать массив;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.
- **BitBtn1** – Kind → bkOk
- **StringGrid1** – ColCount – 100, RowCount – 1, Visible – False, FixedCols – 0, FixedRow – 0;
- **Option** → GoEditing → True.

Процедура для кнопки «Сформировать массив»:

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

Процедура для кнопки «OK»:

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

В задачах на массивы часто используется генератор случайных чисел. Для этого необходимо запустить процедуру *Randomize*, которая инициализирует (запускает) генератор случайных чисел.

Чтобы получить случайное число, нужно воспользоваться функцией *Random*.

Пример.

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

Таким образом, диапазон значений массива будет (-10; 9).

Задача 2.2. Программирование задач для обработки матриц.

В матрице, размерность которой $n \times n$, отрицательные элементы заменить на их квадраты, а положительные элементы уменьшить на 10. Вывести полученный массив. Заполнение исходного массива осуществить с использованием генератора случайных чисел.

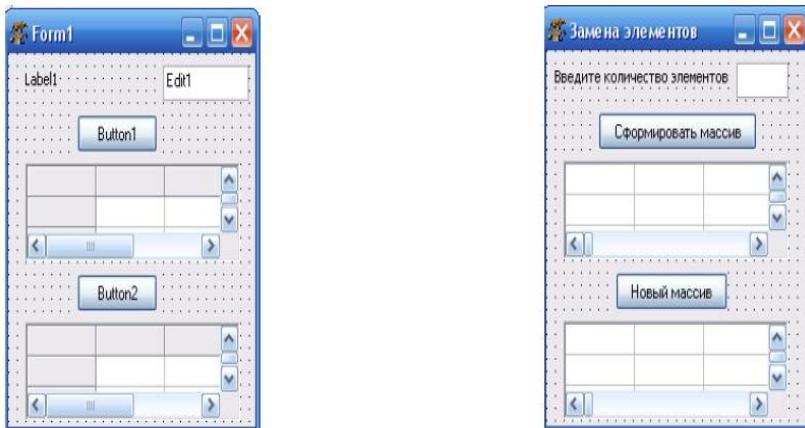
Определим переменные задачи:

Входные данные: $a[i,j]$: integer – элементы массива; n : integer – количество столбцов и строк.

Промежуточные переменные: i, j : integer – параметры элемента.

Результат: $a[i,j]$: integer – элементы массива.

Расположим следующие компоненты на форме.



Свойства выбранных компонент:

- **Form1** – Caption – Замена элементов;
- **Label1** – Caption – Введите количество элементов;
- **StringGrid2** – ColCount – 100, RowCount – 100, Visible – False, FixedCols – 0, FixedRow – 0, Option → GoEditing → True;
- **Button1** – Caption – Сформировать массив;
- **Button2** – Caption – Новый массив;
- **Edit1** – Text – пусто.
- **Edit2** – Text – пусто.
- **BitBtn** – Kind → bkOk
- **StringGrid1** – ColCount – 100, RowCount – 100, Visible – False, FixedCols – 0, FixedRow – 0, Option → GoEditing → True;
- **Option** → GoEditing → True.

Процедура для кнопки «Сформировать массив»:

```
procedure TForm1.Button1Click(Sender: TObject);
var n,i,j:integer;
a:array[0..100,0..100] of integer;
begin
n:=strtoint(edit1.text);
stringgrid1.ColCount:=n;
stringgrid1.RowCount:=n;
```

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

Процедура для кнопки «*Новый массив*»:

```
procedure TForm1.Button2Click();  
var i,j,n:integer;  
a:array[0..100,0..100] of integer;  
begin  
n:=strToInt(edit1.text);  
for i:=0 to n-1 do  
for j:=0 to n-1 do  
a[i,j]:=strToInt(stringGrid1.cells[i,j]);  
for i:=0 to n-1 do  
for j:=0 to n-1 do  
if a[i,j]<0  
then a[i,j]:=sqr(a[i,j])  
else  
if a[i,j]>0  
then a[i,j]:=a[i,j]-10;  
stringgrid2.Visible:=true;  
stringGrid2.ColCount:=n;  
for i:=0 to n-1 do  
for j:=0 to n-1 do  
StringGrid2.Cells[i,j]:=inttostr(a[i,j]);  
end;
```

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Дан массив размера N. Вывести его элементы в обратном порядке.
2.	Дан массив размера N. Вывести вначале его элементы с четными1 нечетными2 индексами, а затем – с нечетными1 четными2.
3.	Дан целочисленный массив A размера 10. Вывести номер первого1 последнего2 из тех его элементов A[i], которые удовлетворяют двойному неравенству: A[1] < A[i] < A[10]. Если таких элементов нет, то вывести 0.
4.	Дан целочисленный массив размера N. Преобразовать его, прибавив к четным1 нечетным2 числам первый3 последний4 элемент. Первый и последний элементы массива не изменять.
5.	Дан целочисленный массив размера N. Вывести вначале все его четные1 нечетные2 элементы, а затем – нечетные1 четные2.
6.	Поменять местами минимальный и максимальный элементы массива размера 10.
7.	Заменить все положительные1 отрицательные2 элементы целочисленного массива размера 10 на значение минимального3 максимального4.
8.	Дан массив размера 10. Переставить в обратном порядке элементы массива, расположенные между его минимальным и максимальным элементами.
9.	Дан массив размера N. Осуществить циклический сдвиг элементов массива влево1 вправо2 на одну позицию.
10.	Дан массив размера N и число k ($0 < k < 5$, $k < N$). Осуществить циклический сдвиг элементов массива влево1 вправо2 на k позиций.
11.	Проверить, образуют ли элементы целочисленного массива размера N арифметическую1 геометрическую2 прогрессию. Если да, то вывести разность1 знаменатель2 прогрессии, если нет – вывести 0.
12.	Дан массив ненулевых целых чисел размера N. Проверить, чередуются ли в нем [четные и нечетные1 положительные и отрицательные2] числа. Если чередуются, то вывести 0, если нет, то вывести номер первого элемента, нарушающего закономерность.

13.	Дан массив размера N. Найти количество его локальных минимумов1 максимумов2.
14.	Дан массив размера N. Найти максимальный1 минимальный2 из его локальных минимумов1 максимумов2.
15.	Дан массив размера N. Определить количество участков, на которых его элементы монотонно возрастают1 убывают2.
16.	Дан массив размера N. Определить количество его промежутков монотонности (то есть участков, на которых его элементы возрастают или убывают).
17.	Дано вещественное число R и массив размера N. Найти элемент массива, который наиболее1 наименее2 близок к данному числу.
18.	Дано вещественное число R и массив размера N. Найти два элемента массива, сумма которых наиболее1 наименее2 близка к данному числу.
19.	Дан массив размера N. Найти номера двух ближайших чисел из этого массива.
20.	Дан целочисленный массив размера N. Определить максимальное количество его одинаковых элементов.
21.	Дан целочисленный массив размера N. Удалить из массива все элементы, встречающиеся [менее двух раз]1 [более двух раз]2 [ровно два раза]3 [ровно три раза]4.
22.	Дан целочисленный массив размера N. Если он является перестановкой, то есть содержит все числа от 1 до N, то вывести 0, в противном случае вывести номер первого недопустимого элемента.
23.	Дан массив размера N. Преобразовать его, вставив перед1 после2 каждого положительного 3 отрицательного4 элемента нулевой элемент.
24.	Дан целочисленный массив размера N. Назовем серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов (длина серии может быть равна 1). Вывести массив, содержащий длины всех серий исходного массива.
25.	Дан целочисленный массив размера N. Преобразовать массив, увеличив1 уменьшив2 каждую его серию на один элемент.

26.	Дан целочисленный массив размера N. Преобразовать массив, увеличив первую1 последнюю2 всес3 серии наибольшей длины на один элемент.
27.	Дан целочисленный массив размера N. Вставить перед1 после2 каждой серии нулевой элемент.
28.	Дано число k и целочисленный массив размера N. Поменять местами первую1 последнюю2 и k-ю серии массива. Если серий в массиве меньше k, то вывести массив без изменений.
29.	Дано число k и целочисленный массив размера N. Удалить из массива все серии, длина которых меньше1 равна2 больше3 k.
30.	Дано число k и целочисленный массив размера N. Заменить серию, длина которой меньше1 равна2 больше3 k, на один нулевой элемент.
31.	Даны два массива А и В размера 5, элементы которых упорядочены по возрастанию1 убыванию2. Объединить эти массивы так, чтобы результирующий массив остался упорядоченным.
32.	Упорядочить массив размера N по возрастанию1 убыванию2.
33.	Дан массив размера N. Вывести индексы массива в том порядке, в котором соответствующие им элементы образуют возрастающую1 убывающую2 последовательность.
34.	Дана точка А и множество В из N точек. Найти номер точки из множества В, наиболее близкой1 удаленной2 от точки А.
35.	Дано множество А из N точек. Среди всех точек этого множества, лежащих в первой1 второй2 третьей3 четвертой4 четверти, найти точку, наиболее близкую5 удаленную6 от начала координат. Если таких точек нет, то вывести точку с нулевыми координатами.
36.	Дано множество А из N точек. Найти пару различных точек этого множества с минимальным1 максимальным2 расстоянием между ними и само это расстояние (точки выводятся в том же порядке, в котором они перечислены при задании множества А).

37.	Дано множество А из N точек. Найти такую точку из данного множества, сумма расстояний от которой до остальных его точек минимальна1 максимальна2, и саму эту сумму.
38.	Даны множества А и В, состоящие соответственно из N1 и N2 точек. Найти минимальное1 максимальное2 расстояние между точками этих множеств и сами точки, расположенные на этом расстоянии.
39.	Дано множество А из N точек. Найти наименьший1 наибольший2 периметр треугольника, вершины которого принадлежат различным точкам множества А, и сами эти точки (точки выводятся в том же порядке, в котором они перечислены при задании множества А).
40.	Дано множество А из N точек с целочисленными координатами. Порядок на координатной плоскости определим следующим образом: $(x_1, y_1) < (x_2, y_2)$, если либо $x_1 < x_2$, либо $x_1 = x_2$ и $y_1 < y_2$. Расположить точки данного множества по возрастанию1 убыванию2 в соответствии с указанным порядком.

Задание 3.2. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Вычислить сумму и количество элементов массива X(100) $0 \leq X \leq 1$
2.	Вычислить среднее арифметическое значение элемента массива A(80) $A_i < 0$
3.	Подсчитать количество элементов массива X(70), удовлетворяющих условию $-1 \leq X_i \leq 1$
4.	Определить максимальный элемент массива B(50) и его порядковый номер $X_i > 0$
5.	Вычислить минимальный элемент массива C(40) и его порядковый номер $X_i < 0$
6.	Вычислить сумму положительных элементов массива D(80)
7.	Вычислить среднее геометрическое элементов массива Y(20) $Y_i > 0$

8.	Подсчитать количество положительных элементов массива Z(30)
9.	Определить сумму элементов массива N(50), кратных трем
10.	Вычислить сумму элементов массива X(N), удовлетворяющих условию $X_i > 0, N \leq 30$
11.	Переписать в массив Y положительные элементы массива X(N) $X_i > 0, N \leq 30$
12.	Переписать в массив Y отрицательные элементы массива X(N) $N \leq 40$
13.	Дан массив a, состоящий из n-элементов. Вычислить $a_1, a_1+a_2, a_1+a_2+a_3, \dots, a_1+a_2+a_3+\dots+a_n$.
14.	Дан массив a, состоящий из n-элементов. Вычислить $a_1, -a_2, a_3, \dots, (-1)^{n-1} a_n$.
15.	Дан массив a, состоящий из n-элементов. Получить массив b, где $b_k = a_k + k!$.
16.	Дан массив a, состоящий из n-элементов. Получить массив b, где $b_k = 2a_k + k$.
17.	Даны массивы a и b, состоящие из n-элементов каждый. Получить массив c, где $c_k = a_k + b_k$.
18.	Даны массивы a и b, состоящие из n-элементов каждый. Получить массив c, где $c_k = a_k * b_k$.
19.	Дан массив a, состоящий из n-элементов. Найти сумму элементов массива, стоящих на нечетных местах.
20.	Дан массив a, состоящий из n-элементов и число x. Вычислить значение полинома $P = x^n + a_1x^{n-1} + \dots + a_{n-1}x + a_n$.

Задание 3.3. Матрицы.

Вариант	Условие задачи
1.	Дано число k ($0 < k < 11$) и матрица размера 4 x 10. Найти сумму и произведение элементов k-го столбца данной матрицы.
2.	Дана матрица размера 5 x 9. Найти суммы элементов всех ее четных1 нечетных2 строк3 столбцов4.
3.	Дана матрица размера 5 x 10. Найти минимальное1 максимальное2 значение в каждой строке3 столбце4.
4.	Дана матрица размера 5 x 10. В каждой строке1 столбце2 найти количество элементов, больших3 меньших4 среднего арифметического всех элементов этой строки1 столбца2.

5.	Дана матрица размера 5 x 10. Преобразовать матрицу, поменяв местами минимальный и максимальный элемент в каждой строке1 столбце2.
6.	Дана матрица размера 5 x 10. Найти минимальное1 максимальное2 значение среди сумм элементов всех ее строк3 столбцов4 и номер строки3 столбца4 с этим минимальным1 максимальным2 значением.
7.	Дана матрица размера 5 x 10. Найти минимальный1 максимальный2 среди максимальных1 минимальных2 элементов каждой строки3 столбца4.
8.	Дана целочисленная матрица размера 5 x 10. Вывести номер ее оerbn1 последней2 строки3 столбца4, содержащего равное количество положительных и отрицательных элементов (нулевые элементы не учитываются). Если таких строк3 столбцов4 нет, то вывести 0.
9.	Дана матрица размера 5 x 10. Вывести номер ее первой1 последней2 строки3 столбца4, содержащего только положительные элементы. Если таких строк3 столбцов4 нет, то вывести 0.
10.	Дана целочисленная матрица размера M x N. Различные строки (столбцы) матрицы назовем похожими, если совпадают множества чисел, встречающихся в этих строках (столбцах). Найти количество строк1 столбцов2, похожих на первую3 последнюю4 строку1 столбец2.
11.	Дана целочисленная матрица размера M x N. Найти количество ее строк1 столбцов2, все элементы которых различны.
12.	Дана целочисленная матрица размера M x N. Вывести номер ее первой1 последней2 строки3 столбца4, содержащего максимальное количество одинаковых элементов.
13.	Дана квадратная матрица порядка M. Найти сумму элементов ее главной1 побочной2 диагонали.
14.	Дана квадратная матрица порядка M. Найти суммы элементов ее диагоналей, параллельных главной1 побочной2 (начиная с одноэлементной диагонали A[1,M]1 A[1,1]2).

15.	Дана квадратная матрица порядка М. Вывести минимальные1 максимальные2 из элементов каждой ее диагонали, параллельной главной3 побочной4 (начиная с одноэлементной диагонали A[1,M]3 A[1,1]4).
16.	Дана квадратная матрица порядка М. Заменить нулями элементы, лежащие одновременно выше1 ниже2 главной диагонали (включая эту диагональ) и выше3 ниже4 побочной диагонали (также включая эту диагональ).
17.	Дана квадратная матрица порядка М. Заменить нулями элементы матрицы, лежащие ниже1 выше2 главной3 побочной4 диагонали.
18.	Дана квадратная матрица порядка М. Зеркально отразить ее элементы относительно [горизонтальной оси симметрии]1 [вертикальной оси симметрии]2 [главной диагонали]3 [побочной диагонали]4 матрицы.
19.	Дана квадратная матрица порядка М. Повернуть ее на 901 1802 2703 градусов в положительном направлении.
20.	Дана матрица размера 5 x 10. Вывести количество строк1 столбцов2, элементы которых монотонно возрастают3 убывают4.
21.	Дана матрица размера 5 x 10. Найти минимальный1 максимальный2 среди элементов тех строк3 столбцов4, которые упорядочены либо по возрастанию, либо по убыванию. Если такие строки3 столбцы4 отсутствуют, то вывести 0.
22.	Даны два числа k1 и k2 и матрица размера 4 x 10. Поменять местами строки1 столбцы2 матрицы с номерами k1 и k2.
23.	Дана матрица размера 5 x 10. Поменять местами строки1 столбцы2, содержащие минимальный и максимальный элементы матрицы.
24.	Дана матрица размера 5 x 10. Поменять местами столбец с номером 11 102 и первый3 последний4 из столбцов, содержащих только положительные элементы.
25.	Дано число k и матрица размера 4 x 10. Удалить строку1 столбец2 матрицы с номером k. Matrix26. Данна матрица размера 5 x 10. Удалить строку1 столбец2, содержащий минимальный3 максимальный4 элемент матрицы.

26.	Дана матрица размера 5 x 10. Удалить первый1 последний2 все3 столбцы, содержащие только положительные элементы.
27.	Дано число k и матрица размера 4 x 9. Перед1 после2 qgrpjh3 столбца4 матрицы с номером k вставить строку3 столбец4 из нулей.
28.	Дана матрица размера 4 x 9. Продублировать строку1 столбец2 матрицы, содержащий ее минимальный3 максимальный4 элемент.
29.	Дана матрица размера 5 x 9. Перед1 после2 первого3 последнего4 столбца, содержащего только положительные элементы, добавить столбец, состоящий из единиц.
30.	Дана целочисленная матрица размера M x N. Найти элемент, являющийся максимальным в своей строке и минимальным в своем столбце. Если такой элемент отсутствует, то вывести 0.
31.	Дана матрица размера M x N. Элемент называется локальным минимумом (максимумом), если он меньше (больше) всех окружающих его элементов. Заменить все локальные минимумы1 максимумы2 данной матрицы на 0.
32.	Дана матрица размера M x N. Поменять местами ее строки1 столбцы2 так, чтобы их минимальные3 максимальные4 элементы образовывали возрастающую5 убывающую последовательность.

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;

- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Что представляет собой массив как структура данных?
2. Что такое индекс?
3. Как определяется положение элемента в одномерном массиве?
4. Формат объявления массива.
5. Формат обращения к массиву и элементам массива.
6. Виды циклических алгоритмов ввода-вывода массива
7. Блок-схемы ввода-вывода одномерного массива
8. Какие типы данных используются для индекса массива?
9. Какие типы данных могут быть элементами массива?
- 10.Как элементы массива располагаются в памяти ПК?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

8. ПРОГРАММИРОВАНИЕ ЗАДАЧ СО СТРОКОВЫМИ ДАННЫМИ

Цель: освоить написание и отладку программ задач со строковыми данными в интегрированной среде разработки ПО Lazarus.

Содержание:

- | | |
|----|---------------------------------------|
| 1. | Краткая теория. |
| 2. | Методика и порядок выполнения работы. |
| 3. | Индивидуальные задания. |
| 4. | Содержание отчета и его форма. |
| 5. | Контрольные вопросы и защита работы. |

1. Краткая теория

Строка называется последовательность символов определенной длины. Элементы строки хранятся по 2 в двух байтах памяти ПК.

Переменные типа *string* могут быть описаны: Var st1:string[30]; st2:string;

где *st1*, *st2* – переменные, *string* – строковый тип данных, в квадратных скобках указывается максимальная длина строки (пример 1); если максимальный размер не указан, то он автоматически принимается равным 255, т. к. максимально возможная строка состоит из 255 символов.

Строка имеет определенную длину (не больше некоторого числа), к каждому символу можно обратиться по его номеру (как в массиве), например,

St1[i] – это обращение к *i*-му элементу строки *st1*.

Если ввести больше символов, чем *n* указанных или возможное максимальное значение, то строка будет равна первым *n* символам, а остальные будут игнорироваться. Стандартные функции обработки строковых величин:

1. СКЛЕИВАНИЕ СТРОК. Под склеиванием понимается последовательное объединение нескольких строк. Для склеивания используется процедура *Concat(str1,str2,...,strn)* – результат – строка, образованная склеиванием строк *str1*, *str2*, ..., *strn*.

Пример.

Str1:='Free';

```
Str2:='pascal';
Str3:=Concat(str1,' ',str2);
```

Значение str3=*Free pascal*.

Функция Pos аналогична операции ‘+’.

Пример.

```
Var str1,str2,str3:string[10];
begin Str1:='Free';
Str2:='pascal';
Str3:=str1+' '+str2;
```

Значение str3=*Free pasca*.

Итоговая строка может состоять максимум из 10 символов. Если в строке будет стоять больше 10 символов, то будут взяты только первые 10, а остальные будут игнорироваться (см. пример).

2. ДЛИНА СТРОКИ. Под длиной строки понимается количество введенных символов. Но она не может превышать максимальной возможной длины. Это значение можно определять при помощи функции *Length(str)*, результат которой целое число, равное количеству символов.

Пример.

```
str1:='март';
str2:='мама мыла раму';
k1:=Length(str1);
k2:=Length(str2);
```

В результате значения целых переменных будут равны k1=4, k2=15.

3. ПОИСК ПОДСТРОКИ В СТРОКЕ. Для этого используется функция *Pos(str1,str)* – позволяет определить положение подстроки str1 в строке str. Результат – целое число, равное номеру позиции, с которой в строке str начинается строка str1. Если подстрока str1 не найдена, то значение функции равно 0.

Пример.

```
Str1:='cal';
Str:='object pascal';
Str2:=Pos(str1,str);
Значение str2=11.
```

4. КОПИРОВАНИЕ. Для этого используется функция *Copy(str,n,m)* – копирует *m* символов строки *str*, начиная с *n*-го символа; при этом исходная строка не меняется. Можно результат этой функции присваивать другой строке или сразу выводить на экран.

Пример.

```
str1:='ABCDEFGHI';  
str2:='abcdefghijklmnopqrstuvwxyz';  
str3:=Copy(str1,4,3);
```

Значение переменной *str3*='DEF'.

Стандартные процедуры обработки строковых величин:

1. УДАЛЕНИЕ Для этого используется процедура *Delete(str,n,m)*, которая вырезает из строки *str* *m* символов, начиная с *n*-го; таким образом, строка изменяется.

Пример.

```
Str1:='ABCDEFGHI';  
Delete(str1,3,4);
```

После выполнения этих операторов из строки будут удалены 4 символа, начиная с 3-его, т. е. *Str1*='ABGH'.

2. ВСТАВКА. Для этого используется процедура *Insert(str1,str2,n)* – вставка строки *str1* в строку *str2*, начиная с *n*-го; при этом строка *str1* остается без изменения, а строка *str2* получает новое значение.

Пример.

```
Str1:='ABCDEFGHI';  
Str2:='abcdefghijklmnopqrstuvwxyz';  
Insert(str1,str2,3);
```

После выполнения этих операторов строка *str2* будет иметь вид: *Str2*='abABCDEFGHIcdefgh'.

Такой же результат будет после выполнения следующей последовательности операторов:

```
str2:='abcdefghijklmnopqrstuvwxyz';  
Insert('ABCDEFGHI',str2,3);
```

2. Методика и порядок выполнения работы

Задача 2.1. Создание текстовых данных

Дана строка символов. Удалить из строки заданный символ.

Определим переменные задачи:

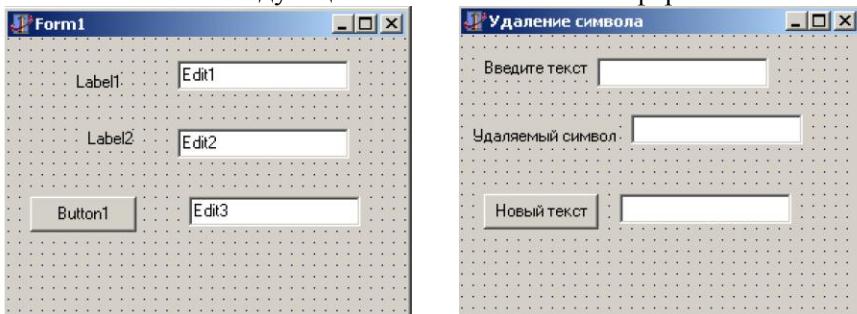
Входные данные: str: строкового типа – вводимый текст;
c: строкового типа – удаляемый элемент.

Промежуточные переменные: i: целого типа – параметр элемента, номер элемента строки, изменяющийся с шагом 1;

Результат: str: строкового типа – преобразованный текст.

Решение данной задачи рассмотрим для цикла с постусловием.

Расположим следующие компоненты на окне формы.



Свойства выбранных компонент:

- **Form1** – Caption – Удаление символа;
- **Label1** – Caption – Введите текст;
- **Label2** – Caption – Удаляемый символ;
- **Button1** – Caption – Новый текст;
- **Edit1** – Text – пусто;
- **Edit2** – Text – пусто;
- **Edit3** – Text – пусто.

Программа решения задачи с помощью цикла с постусловием следующие:

```
procedure TForm1.Button1Click(Sender: TObject);
var str,c:string;
i:integer;
begin
str:=edit1.text;
c:=edit2.text;
i:=0;
repeat if str[i]=c
```

```
then delete(str,i,1)
else i:=i+1;
until i>length(str);
edit3.text:=str;
end;
```

Задача 2.2. Поиск заданных символов в тексте

Составить программу, заменяющую в тексте данный слог на введенный, и подсчитывающую количество замен.

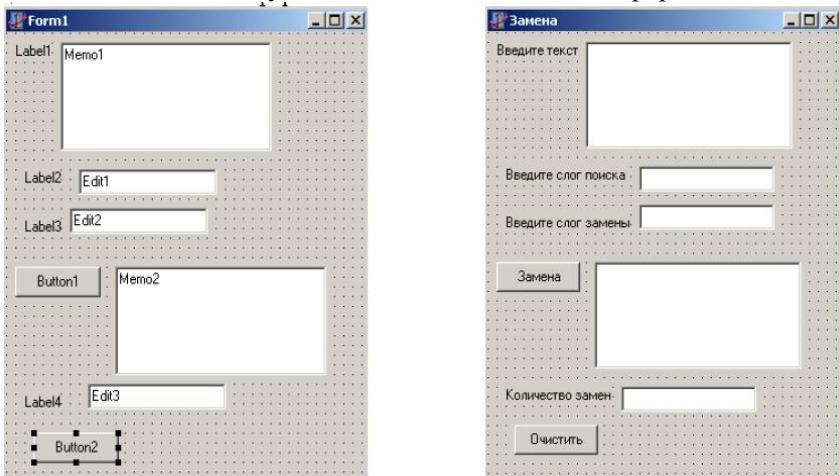
В задаче необходимо ввести текст, искомый слог и слог замены.

Для этого будем использовать входные переменные: S – текст; a – искомый слог; b – слог замены.

Результатом будет являться: новый текст (уже заданная переменная S); k – количество замен. Промежуточные переменные будут описываться в ходе написания программы.

Решение данной задачи рассмотрим для цикла с предусловием.

Расположим следующие компоненты на окне формы.



Свойства выбранных компонент:

- **Form1** – Caption – Замена;
- **Label1** – Caption – Введите текст;
- **Label2** – Caption – Введите слог поиска;
- **Label3** – Caption – Введите слог замены;

- **Label4** – Caption – Произведено замен;
- **Memo1** – Lines – пусто;
- **Memo2** – Lines – пусто;
- **Edit1** – Text – пусто;
- **Edit2** – Text – пусто;
- **Edit3** – Text – пусто.

В рассматриваемой программе будет происходить два события: замена и очистка.

```
procedure TForm1.Button1Click();
var Str,a,b:string;
i,k:integer;
begin
Str:=memo1.text;
a:=edit1.text;
b:=edit2.text;
k:=0; i:=Pos(a,str);
while i<>0 do begin delete(Str,i,length(a));
Insert (b,Str,i);
k:=k+1;
i:=Pos(a,str);
end;
memo2.text:=Str;
Edit3.text:=IntToStr(k);
Memo2.visible:=true;
Edit3.visible:=true;
label4.visible:=true;
end;
```

Примечание. При присваивании текстовой компоненте значения строковой переменной; строковой компоненте значения компоненты функции перевода не используются.

Например, Memo2.text:=S; S:=Memo1.text.

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Удалить из строки буквы 'b', 'c', 'd'
2.	Если какой-либо символ встречается в строке более одного раза, то первое вхождение этого символа оставить без изменения, второе и последующие удалить.
3.	Строка содержит слова, разделенные несколькими пробелами. Оставить по одному пробелу между словами.
4.	Подсчитать, сколько раз в данной строке встречается символ 's'.
5.	Написать строку задом наперед.
6.	Определить, содержит ли данная строка цифры, если да, то вывести их на экран
7.	В строке найти цифры и удалить их
8.	Строка состоит из цифр и букв. Строчные латинские буквы заменить на прописные.
9.	Вывести на экран только первые символы слов.
10.	Заменить все вхождения подстроки 'del' на 'Insert'.
11.	Подсчитать сумму цифр, входящих в данную строку.
12.	Удалить из строки цифры и вывести на экран строку без цифр.
13.	Дан текст. Подсчитать количество слов в данной строке.
14.	Дан текст. Подсчитать количество букв а в последнем слове данной строки.
15.	Дан текст. Найти количество слов, начинающихся с буквы б.
16.	Дан текст. Найти количество слов, у которых первый и последний символы совпадают между собой.
17.	Дан текст. Найти длину самого короткого слова.
18.	Составить программу циклической перестановки букв в словах текста так, что i-я буква слова становится i+1-ой, а последняя – первой.

19.	В каждом слове текста замените "а" на букву "е", если "а" стоит на четном месте, и заменить букву "б" на сочетание "ак", если "б" стоит на нечетном месте.
20.	Отредактировать заданное предложение текста, удаляя из него все слова с нечетными номерами и переворачивая слова с четными номерами. Например, HOW DO YOU DO \Rightarrow OD OD

Задание 3.2. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи
1.	Вывести строку длины N (N — четное), которая состоит из чередующихся символов C1 и C2, начиная с C1.
2.	Дана строка. Вывести строку, содержащую те же символы, но расположенные в обратном порядке.
3.	Дана строка. Вывести коды ее первого и последнего символа.
4.	Дана строка. Подсчитать количество содержащихся в ней цифр1 прописных букв]2 [строчных букв]3.
5.	Дана строка. Преобразовать все строчные1 прописные2 латинские3 русские4 буквы в прописные1 строчные2.
6.	Дана строка. Если она представляет собой запись целого числа, то вывести 1; если вещественного (с дробной частью), то вывести 2; если строку нельзя преобразовать в число, то вывести 0.
7.	Дано целое число. Вывести набор символов, содержащий цифры этого числа в исходном1 обратном2 порядке.
8.	Дана строка S, изображающая вещественное число в формате с плавающей точкой, и целое число N (> 0). Вывести набор символов, изображающих первые N цифр дробной части этого вещественного числа (без округления).
9.	Дана строка, изображающая двоичную1 десятичную2 запись целого числа. Вывести строку, изображающую десятичную1 двоичную2 запись этого же числа.
10.	Дана строка, изображающая целое число. Вывести сумму цифр этого числа.

11.	Дана строка S и число N. Преобразовать строку S в строку длины N следующим образом: если длина строки S больше N, то отбросить первые символы, если длина строки S меньше N, то в ее начало добавить символы "." (точка).
12.	Даны два числа: N1 и N2, и две строки: S1 и S2. Получить из этих строк новую строку, объединив N1 первых символов строки S1 и N2 последних символов строки S2.
13.	Даны две строки: S1 и S2. Проверить, содержится ли строка S2 в строке S1. Если да, то вывести номер позиции, начиная с которой S2 содержится в S1, если нет, то вывести 0.
14.	Даны две строки: S1 и S2. Определить количество вхождений строки S2 в строку S1.
15.	Дана строка S и символ C. Удвоить каждое вхождение qhlbnk` C в строку S.
16.	Даны строки S1, S2 и символ C. Перед1 после2 каждого вхождения символа C в строку S1 вставить строку S2.
17.	Даны две строки: S1 и S2. Удалить из строки S1 первую1 последнюю2 все3 подстроки, совпадающие с S2. Если таких подстрок нет, то вывести S1 без изменений.
18.	Даны три строки: S1, S2, S3. Заменить в строке S1 первое1 последнее2 все3 вхождения строки S2 на S3.
19.	Дана строка. Вывести подстроку, расположенную между первой и второй1 последней2 точками исходной строки. Если в строке менее двух точек, то вывести всю исходную строку.
20.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Определить количество слов в строке.
21.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Определить количество слов, которые [начинаются и заканчиваются одной и той же буквой]1 [содержат хотя бы одну букву "A"]2.
22.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Определить количество слов, которые содержат ровно три буквы "A".
23.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Определить длину самого короткого1 длинного2 слова.

24.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, но разделенные одним символом "." (точка). В конце точку не ставить.
25.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова (разделенные одним пробелом), но расположенные в обратном порядке.
26.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, удалив из него все последующие ₁ предыдущие ₂ вхождения первой ₁ последней ₂ буквы этого слова (количество пробелов между словами не изменять).
27.	Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова (разделенные одним пробелом), но расположенные в алфавитном порядке.
28.	Дана строка-предложение на русском языке. Преобразовать строку так, чтобы каждое слово начиналось с заглавной буквы.
29.	Дана строка-предложение на русском языке. Подсчитать количество содержащихся в строке [знаков препинания] ₁ [гласных букв] ₂ .
30.	Дана строка-предложение на русском языке. Вывести самое короткое ₁ длинное ₂ слово в предложении (если таких слов несколько, то вывести первое ₃ последнее ₄ из них).
31.	Дана строка-предложение, содержащая избыточные пробелы. Преобразовать ее так, чтобы между словами был ровно один пробел.
32.	Дана строка, содержащая полное имя файла, то есть имя диска, список каталогов (путь), собственно имя и расширение. Выделить из этой строки имя ₁ расширение ₂ файла.
33.	Дана строка, содержащая полное имя файла. Выделить из строки название последнего каталога (без символов "\""). Если файл содержится в корневом каталоге, то вывести символ "\".

34.	Дана строка-предложение на русском языке. Зашифровать ее, выполнив циклическую замену каждой буквы на следующую за ней в алфавите и сохраняя при этом регистр букв ("А" перейдет в "Б", "а" – в "б", "Б" – в "В", "я" – в "а" и т.д.). Букву "ё" в алфавите не учитывать ("е" должна переходить в "ж"). Знаки препинания и пробелы не изменять.
35.	Дана строка-предложение на русском языке и число k ($0 < k < 10$). Зашифровать строку, выполнив циклическую замену каждой буквы на букву того же регистра, расположенную в алфавите на k -й позиции после шифруемой буквы (например, для $k = 2$ "А" перейдет в "В", "а" – в "в", "Б" – в "Г", "я" – в "б" и т.д.). Букву "ё" в алфавите не учитывать, знаки препинания и пробелы не изменять.
36.	Дано зашифрованное предложение на русском языке (способ шифрования описан в задании String35) и кодовое смещение k ($0 < k < 10$). Расшифровать предложение.
37.	Дано зашифрованное предложение на русском языке (способ шифрования описан в задании String35) и его расшифрованный первый символ С. Определить кодовое смещение k и расшифровать предложение.
38.	Дана строка-предложение. Зашифровать ее, поместив вначале все символы, расположенные на четных местах, а затем, в обратном порядке, все символы, расположенные на нечетных местах (например, строка "Программа" превратится в "ргамамроП").
39.	Дано предложение, зашифрованное по правилу, описанному в задании. Расшифровать это предложение.
40.	Дана строка, содержащая несколько круглых скобок. Если скобки расставлены правильно (то есть каждой открывающей соответствует одна закрывающая), то вывести число 0. В противном случае вывести или номер позиции, в которой расположена первая ошибочная закрывающая скобка, или, если закрывающих скобок не хватает, число -1.

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Как объявить величину символьного типа, строкового типа?
2. К каким типам данных относятся строки?
3. Какова максимально возможная длина строки?
4. С величиной какого типа данных совместим по присваиванию отдельный символ строки?
5. Какие операции можно выполнять над строковыми данными?
6. Какие существуют стандартные функции для величин строкового типа?
7. Какие существуют стандартные процедуры для величин строкового типа?
8. Как осуществляется доступ к отдельному символу строки?
9. Почему значение отношения 'IBM'<>'ibm' равно TRUE?
10. Какая функция (процедура) является аналогом операции склеивания (+) при работе со строками?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

9. ПРОГРАММЫ ПОСТРОЕНИЯ ГРАФИКОВ

Цель: освоить написание и отладку программ построения графиков в интегрированной среде разработки программного обеспечения Lazarus.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория

Существуют классы-надстройки, созданные для использования в графических инструментах Windows: для контекста – класс *TCanvas*, для шрифта – *TFont*, для пера – *TPen*, для кисти – *TBrush*.

Класс TFont. С помощью класса *TFont* создаются объект-шрифт для любого графического устройства (в частности, объекты на экране).

Для данного класса характерны свойства:

Color: TColor;	Цвет шрифта.
Height: Integer;	Высота шрифта в пикселях экрана.
Name: TfontName;	Имя шрифта.
Size: Integer;	Высота шрифта в пунктах (1/72 дюйма).
Style = [fsbold, fsItalic, fsUnder- Line, fsStrikeOut]	Стиль шрифта. Может принимать комбинацию элементов: <i>fsbold</i> (жирный), <i>fsItalic</i> (курсив), <i>fsUnderLine</i> (подчеркнутый), <i>fsStrikeOut</i> (перечеркнутый).

Почти каждый компонент имеет свойство *Font:TFont*. Слева от свойства стоит знак «+», который указывает, что свойство вложенное. Щелчок мышью на знаке плюс открывает вложенные свойства класса *Tfont*.

Классы TPen, TBrush. С помощью класса *TPen* (*Перо*) создается объект *Перо*, служащий для вычерчивания линий, контуров графических рисунков.

Свойства класса:

Color: TColor;	Цвет вычерчиваемых пером линий.
Style: TPenStyle;	Определяет стиль линий: сплошная, пунктирная и т. д.
Width: Integer;	Толщина линий в пикселях экрана.
Mode: TPenMode;	Определяет способ взаимодействия линий с фоном.

Объект класса *TBrush* (*Кисть*) служит для заполнения внутреннего пространства замкнутых фигур.

Свойства класса:

Bitmap: TBitMap;	Содержит растовое изображение, которое будет использоваться кистью для заполнения. Если это свойство определено, то свойства Color, Style игнорируются.
Color: TColor;	Цвет вычерчиваемых пером линий.
Style: TBrushStyle;	Определяет стиль линий заполнения: сплошная закраска, диагональные линии, горизонтальные линии и т. д.

Класс TCanvas. Данный класс создает «канву», на которой можно рисовать. Объекты класса *TCanvas* автоматически создаются для всех видимых неоконных компонентов.

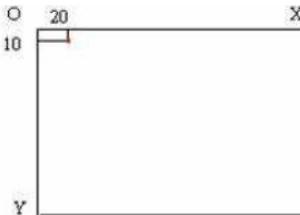
Класс имеет свойства *Pen: TPen, Brush: TBrush, Font: TFont*.

Задание цвета пера и кисти можно осуществить несколькими способами. Например, непосредственно указав стандартные цвета Windows (*clred, clblue, clgreen* и т.д.).

Наиболее распространенным способом задания цвета в Windows является цветовая модель *RGB* (по первым буквам названий цветов: *Red* – красный, *Green* – зеленый, *Blue* – голубой). Комбинируя эти три цвета, можно получить любой цвет. Для этого интенсивность каждого из трех составляющих цветов задается значением от 0 до 255.

Рассмотрим основные свойства и методы класса *TCanvas*. *Pixels[x,y: Integer]: TColor* – это двумерный массив, который отвечает за цвет пикселя.

Координаты пикселя в массиве определяются согласно рисунка.



Например, для размещения красной точки в объекте *Label1* с координатами (10,20) следуют записать:

Label1.Canvas.Pixel(10,20):=clred;

PenPos: TPoint – определяет текущее положение пера в пикселях относительно левого верхнего угла канвы.

MoveTo(X, Y: integer) – перемещение пера, курсор устанавливается, не рисуя, в позицию (X, Y).

LineTo(X, Y) – рисуется отрезок от позиции, в которой установлено перо, до позиции (X, Y).

PolyLine(Points: array of TPoints) – рисование контура многоугольника. Для этого используется массив *POINT:TPOINT*, передающий последовательность точек, которые функция *PolyLine* соединяет линиями. Чтобы многоугольник получился замкнутым, необходимо соединить позиции последней и первой точек.

Пример рисования треугольника:

label1.Canvas.Pen.Color:=clgreen;

Label1.Canvas.PolyLine (Points(10,10), Points(50,75), Points(100,10), Points(10,10));

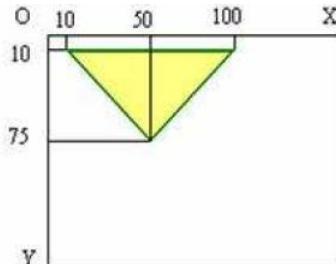
Polygon (Points: array of TPoints) – рисование закрашенных многоугольников. Контур рисуется установленными свойствами пера, при закрашивании используются текущие свойства кисти. Пример рисования закрашенного треугольника:

Labell.Canvas.Pen.Color := clgreen;

Labell.Canvas.Pen.Width := 4;

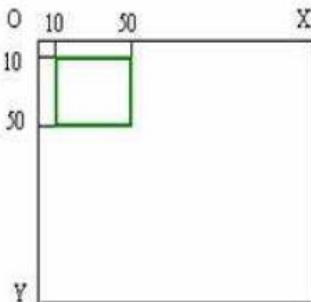
label1.Canvas.Brush.Color:=clYellow;

Label1.Canvas.Polygon (Points(10,10), Points(50,75), Points(100,10), Points(10,10));



Rectangle(x1,y1,x2,y2: integer) – рисование контура прямоугольника. Рисуется контур прямоугольника, где $(x1,y1)$ – координаты верхнего левого угла, $(x2,y2)$ – координаты нижнего правого угла. При рисовании используются текущие свойства пера, которые можно задать перед применением процедуры рисования. Например:

```
Label1.Canvas.Pen.Color := clgreen;
Label1.Canvas.Pen.Width := 4;
Label1.Canvas.Rectangle(10,10,50,50);
```

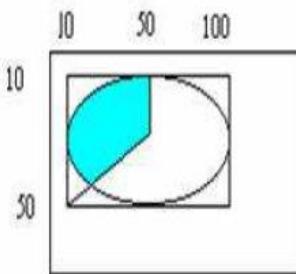


FillRect(Rect(x1,y1,x2,y2:integer)) – рисуется закрашенный прямоугольник, где функция *Rect(x1,y1,x2,y2)* возвращает размер прямоугольной области аналогично функции *Rectangle*, в которой $(x1,y1)$ – координаты верхнего левого угла, $(x2,y2)$ – координаты нижнего правого угла. Контур рисуется установленными свойствами пера, при закрашивании используются текущие свойства кисти.

Ellipse(X1,Y1,X2,Y2: integer) – рисование закрашенных окружностей и эллипсов. Чертит эллипс в охватывающей прямоугольной области. При рисовании окружностей указываются размеры квадратной области.

Pie(x1,y1,x2,y2,x3,y3,x4,y4:integer) – рисование секторов эллипса. При этом первые четыре параметра указывают размер прямоугольной области. Параметры $X3, Y3, X4, Y4$ определяют часть эллипса, которая будет показана. Начало дуги лежит на пересечении эллипса и луча, проведенного из его центра в точку $(x3,y3)$, а конец – на пересечении с лучом из центра в точку $(x4,y4)$. Дуга чертится против часовой стрелки. Например:

```
Label1.Brush.Color := clblue;
Label1.Canvas.Pie (10,10,100,50,50,10,10,50);
```



TextOut(x,y:integer; S:string) – выводит текстовую строку S, начиная с позиции (x,y). Перед применением функции можно задать параметры шрифта.

Например:

```
Label1.canvas.font.Style:=[fsbold];
Label1.canvas.TextOut (1,1,'чертеж');
```

Компоненты *TShape*, *TImage*, *TPaintBox*, *TChart*.

TShape – стандартная фигура. Компонент рисует одну из простейших геометрических фигур: прямоугольник, квадрат, скругленный прямоугольник, скругленный квадрат, эллипс, окружность.

TImage – отображение картинок. Этот компонент служит для размещения на форме трех видов изображения: растровой картинки, пиктограммы или метафайла. Изображение содержится в свойстве компонента *Picture*. Свойство *Canvas* позволяет отредактировать растровое изображение.

Свойства компонента:

Shape = (stRectangle, stSquare, stRoundRect, stRoundSquare, stEllipse, stCircle);	Определяет вид фигуры: stRectangle – прямоугольник, stCircle – окружность, stSquare – квадрат, stEllipse – эллипс, stRoundRect – скругленный прямоугольник, stRoundSquare – скругленный квадрат.
+ Brush: Tbrush;	Кисть, служит для заполнения внутреннего пространства фигур. Имеет вложенные свойства.
+ Pen: Tpen;	Перо, служит для вычерчивания линий, контуров графических рисунков.

TPaintBox – окно для рисования. Компонент размещает на форме прямоугольную область (канву) для рисования. Если при рисовании фигуры ее размеры окажутся больше компонента *TPaintBox*, то фигура окажется усеченной – не выйдет за пределы области рисования. Создав обработчик события *OnPaint* для *TPaintBox*, можно передвигать изображение по форме, изменения у *TPaintBox* свойства *Left* и *Top*. *TPaintBox* использует свойство *ALIGN* для удержания изображения сверху, снизу, сверху, внизу формы или заполнять всю область формы.

2. Методика и порядок выполнения работы

Задача 2.1. Компонент построения графиков *TChart*

TChart – построитель графиков. Компонент предназначен для графического представления числовых данных.

После размещения компонента на форме нужно щелкнуть по нему правой кнопкой мыши для вызова контекстного меню и выбрать команду *Edit Chart*, после чего появляется окно редактирования компонента.

В окне редактирования нужно нажать кнопку *Add* и выбрать подходящий тип графика. Каждый выбранный график называется серией, которые нумеруются, начиная с нуля. Доступ к сериям осуществляется с помощью закладки *Series*. Окно редактирования имеет закладки для задания заголовка графика *Title*, редактирования осей – *Axis*.

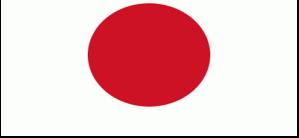
После закрытия окна редактора компонент будет содержать примерный вид графика. Однако его реальный вид зависит от фактических данных, которые создаются в работающей программе. В программе обращаются к свойству *SeriesList[<номер серии>]* – список серий; первая созданная серия имеет индекс 0, вторая 1 и т.д. Свойству *SeriesList* доступны методы: *AddX* – добавить данные по *X*, *AddY* – добавить данные по *Y*, *AddXY* – добавить данные по *X* и *Y*. Например, обработчик события *OnCreate* формы создает график $y=5-\sin(x)$ (график будет показан при открытии окна программы):

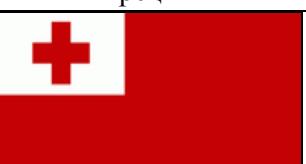
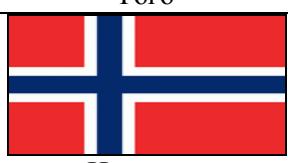
```
procedure TForm1.FormCreate(Sender: TObject);
const PI=3.14;
var x:integer;
y:real;
begin
for x:=-100 to 100 do
begin
y:= 5-sin(x/pi);
Chart1.SeriesList[0].AddXY(x,y);
end;
end;
```

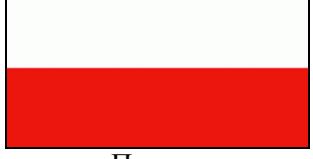
3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

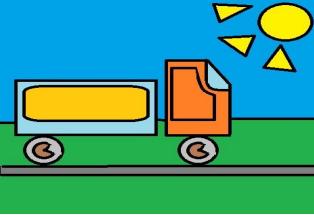
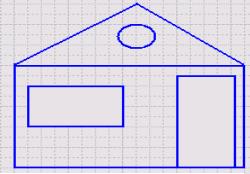
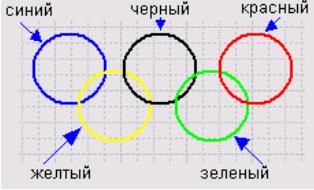
Задание 3.1. Разработать программу, рисующую флаги государств.

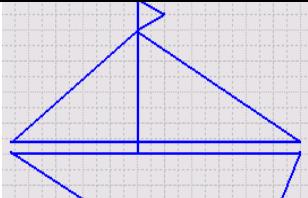
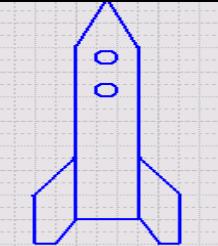
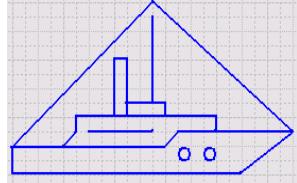
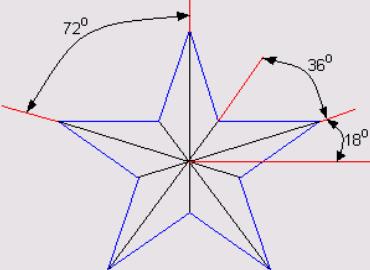
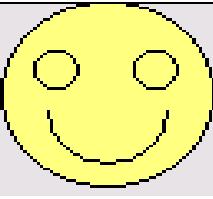
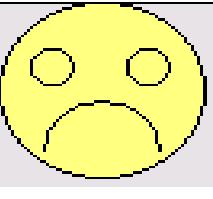
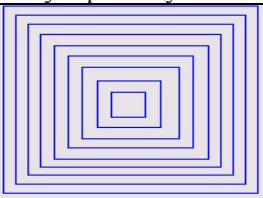
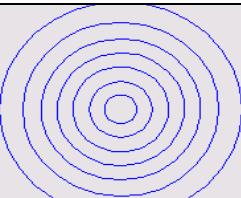
Вариант	Условие задачи	Вариант	Условие задачи
1.		2.	
	Австрия		Япония
3.		4.	
	Армения		Багамы
5.		6.	
	Бангладеш		Бельгия
7.		8.	
	Бенин		Болгария
9.		10.	
	Боливия		Ботсвана
11.		12.	
	Ямайка		Венгрия

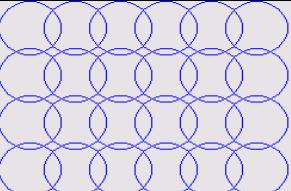
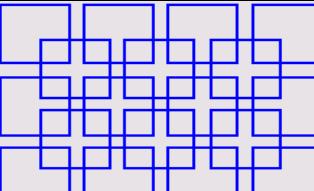
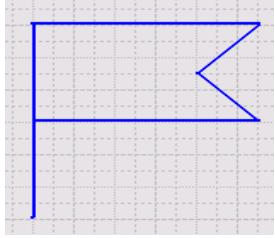
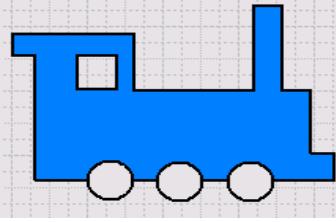
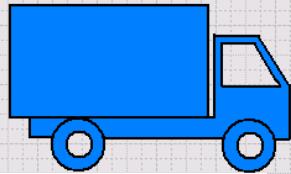
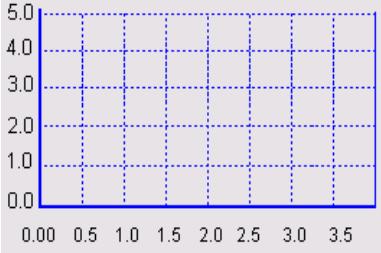
13.		14.	
	Эстония		Швеция
15.		16.	
	Швейцария		Чили
17.		18.	
	Чехия		Вьетнам
19.		20.	
	Финляндия		Греция
21.		22.	
	Того		Тонга
23.		24.	
	Норвегия		Судан

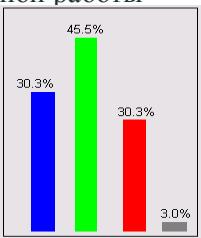
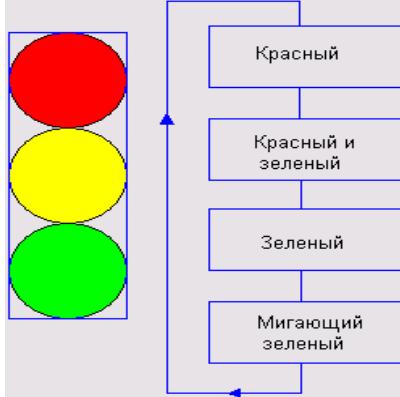
25.		26.	
	Франция		Россия
27.		28.	
	Чад		Гамбия
29.		30.	
	Украина		Суринам
31.		32.	
	Сьерра-Леоне		Таиланд
33.		34.	
	Сейшельы		Северная Корея
35.		36.	
	Сомали		Польша

Задание 3.2. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Условие задачи		Условие задачи	
1.		2.	
	Написать программу, которая вычерчивает на экране домик.		Используя методы рисования написать программу, результатом выполнения которой является рисунок.
3.	Написать программу, которая вычерчивает на экране узор из 100 окружностей случайного диаметра и цвета.	4.	Написать программу, которая вычерчивает на экране узор из 50 прямоугольников случайного размера и цвета.
5.	Написать программу, которая вычерчивает на экране узор – ломаную линию из 100 звеньев со случайными координатами, случайного цвета.	6.	Написать программу, которая выводит на экран изображение идущих часов, у которых есть секундная и минутная стрелки.
7.		8.	
	Написать программу, которая вычерчивает на экране домик.		Написать программу, которая выводит на экран флаг Олимпийских игр. (одной клетке соответствует пять пикселов).

9.	 <p>Написать программу, которая вычерчивает на экране кораблик.</p>	10.	 <p>Написать программу, которая вычерчивает на экране ракету.</p>
11.	 <p>Написать программу, которая с использованием метода базовой точки выводит на экран изображение кораблика.</p>	12.	 <p>Написать программу, которая выводит на экран контур пятиконечной звезды.</p>
13.	 <p>Написать программу, которая рисует на экране веселую рожицу.</p>	14.	 <p>Написать программу, которая рисует на экране грустную рожицу.</p>
15.	 <p>Написать программу, которая выводит на экран изображенный ниже узор.</p>	16.	 <p>Написать программу, которая выводит на экран изображенный ниже узор.</p>

17.	Написать программу, которая вычерчивает на экране шестиугольник.	18.	Написать программу, которая выводит на экран изображение шахматной доски.
19.	 Написать программу, которая выводит на экран изображенный ниже узор.	20.	 Написать программу, которая выводит на экран изображенный ниже узор.
21.	 Написать программу, которая рисует на экране флагок красного цвета.	22.	 Написать программу, которая рисует на экране паровоз. Используйте метод базовой точки.
23.	 Написать программу, которая рисует на экране автомобиль. Инструкции, обеспечивающие вычерчивание колеса автомобиля, оформите как функцию.	24.	 Написать программу, которая выводит на экран оцифрованную координатную сетку.

25.	<p>Написать программу, которая выводит на экран гистограмму успеваемости класса, например по итогам контрольной работы. Обработка результатов контрольной работы. Введите исходные данные:</p> <p>пятерок -> 10; четверок -> 15; троек -> 7; двоек -> 1;</p> <p>Результаты контрольной работы</p>  <table border="1"> <thead> <tr> <th>Оценка</th> <th>Процент</th> </tr> </thead> <tbody> <tr> <td>пятерок</td> <td>30.3%</td> </tr> <tr> <td>четверок</td> <td>45.5%</td> </tr> <tr> <td>троек</td> <td>30.3%</td> </tr> <tr> <td>двоек</td> <td>3.0%</td> </tr> </tbody> </table>	Оценка	Процент	пятерок	30.3%	четверок	45.5%	троек	30.3%	двоек	3.0%	26.	<p>Написать программу, которая выводит на экран изображение работающего светофора. Рекомендуемый вид светофора и алгоритм его работы приведены ниже.</p>  <pre> graph TD A[Красный] --> B[Красный и зеленый] B --> C[Зеленый] C --> D[Мигающий зеленый] D --> A </pre>
Оценка	Процент												
пятерок	30.3%												
четверок	45.5%												
троек	30.3%												
двоек	3.0%												
27.	<p>Написать программу, которая выводит на экран точечный график функции $y=0.5x^2+4x-3$. Диапазон изменения аргумента - от -15 до 5; шаг аргумента- 0.1.</p> <p>График вывести на фоне координатных осей, точка пересечения которых должна находиться в центре экрана.</p>	28.	<p>Написать программу, которая выводит круговую диаграмму, отражающую товарооборот (в процентах) книжного магазина. Исходные данные (объем продаж в рублях по категориям: книги, журналы, открытки и канцтовары) вводятся во время работы программы.</p> <p>Пример диаграммы приведен ниже.</p>  <table border="1"> <thead> <tr> <th>Категория</th> <th>Процент</th> </tr> </thead> <tbody> <tr> <td>Книги</td> <td>34.4%</td> </tr> <tr> <td>Журналы</td> <td>31.4%</td> </tr> <tr> <td>Канцтовары</td> <td>22.9%</td> </tr> <tr> <td>Прочее</td> <td>11.4%</td> </tr> </tbody> </table>	Категория	Процент	Книги	34.4%	Журналы	31.4%	Канцтовары	22.9%	Прочее	11.4%
Категория	Процент												
Книги	34.4%												
Журналы	31.4%												
Канцтовары	22.9%												
Прочее	11.4%												

29.	Написать программу, которая выводит на экран пятиконечную звезду красного цвета с белой окантовкой.	30.	Написать программу, которая рисует движущуюся по экрану окружность.
-----	---	-----	---

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Основные средства рисования – свойства *Canvas*, *Pen*, *Brash*.
2. Основные свойства кисти *Brash*.
3. Основные свойства карандаша *Pen*.
4. Основные свойства холста *Canvas*.
5. Рисование линий, ломанных линий.
6. Рисование эллипсов и дуг окружностей
7. Рисование прямоугольников и многоугольников
8. Вывод гистограмм и графиков
9. Какие графические функции вы изучили в данной работе?
10. Что такое кисть, перо, зачем они нужны и их отличия друг от друга?
11. Какое свойство объекта *Canvas* определяет цвет, стиль, толщину линий при использовании методов *Line*, *Ellipse*, *Rectangle*?

12. Какая функция обработки события используется при выводе графики?
13. Каким образом создаётся на форме главное меню?
14. Что такое кисть, перо, зачем они нужны и их отличия друг от друга?
15. Напишите инструкцию для вывода на поверхность графического объекта текста. Как изменить его цвет и шрифт?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

10. СОЗДАНИЕ ГРАФИЧЕСКИХ ПРИМИТИВОВ В СРЕДЕ LAZARUS

Цель: приобретение навыков создания приложения с использованием графических примитивов в интегрированной среде разработки ПО Lazarus.

Содержание:

- | | |
|----|---------------------------------------|
| 1. | Краткая теория. |
| 2. | Методика и порядок выполнения работы. |
| 3. | Индивидуальные задания. |
| 4. | Содержание отчета и его форма. |
| 5. | Контрольные вопросы и защита работы. |

1. Краткая теория

Многие компоненты в Lazarus имеют свойство *Canvas* (канва, холст), представляющие собой область компонента, на которой можно рисовать или отображать готовые изображения. Свойство *Canvas* используется для рисования пером (свойство *Pen*) и (кистью (свойство *Brush*).

Канва обеспечивает:

- загрузку и хранение графических изображений;
- создание новых и изменение имеющихся изображений с помощью пера, кисти, шрифта;
- рисование и закраску различных фигур, линий, текстов;
- комбинирование различных изображений.

Каждая точка канвы имеет координаты X и Y. Система координат канвы имеет началом левый верхний угол канвы. Координата X возрастает при перемещении слева направо, а координата Y – при перемещении сверху вниз. Координаты измеряются в пикселях.

Пиксель – наименьший элемент поверхности рисунка. Важнейшее свойство пикселя – его цвет. Для описания цвета используется тип *TColor*.

Для того, чтобы вывести на поверхность объекта графический элемент, необходимо применить к свойству *Canvas* этого объекта соответствующий метод. Например, для вычерчивания на форме прямоугольника, можно записать:

Form1.Canvas.Rectangle(10,10,100,100);

Карандаш используется для вычерчивания точек, линий, контуров геометрических фигур: прямоугольников, окружностей, эллипсов, дуг и др. Вид линии, которую оставляет карандаш на поверхности холста, определяют свойства объекта *Tpen*, представленные в таблице 10.1.

Таблица 10.1
Свойства объекта *Tpen*

Color	Цвет линии
Width	Толщина линии
Style	Вид линии
Mode	Режим отображения

Например, указание

Form1.Canvas.Pen.Width:=2; – устанавливает толщину линии 2 пикселя.

Form1.Canvas.Pen.Color:=clRed; – устанавливает красный цвет линии.

Style определяет стиль линии, который можно задать именованной константой. Список констант представлен в таблице 10.2.

Таблица 10.2
Список констант

psSolid	Сплошная линия
psDash	Пунктирная линия, длинные штрихи
psDot	Пунктирная линия, короткие штрихи
psDashDot	Пунктирная линия, чередование длинного и короткого штрихов
psDashDotDot	Пунктирная линия, чередование одного длинного и двух коротких штрихов
psClear	Линия не отображается

Кисть используется методами, обеспечивающими вычерчивание замкнутых областей и обладает двумя свойствами таблица 10.3.

Таблица 10.3
Свойства кисти

Color	Цвет закрашивания замкнутой области
Style	Стиль заполнения области

Константы, позволяющие задать стиль заполнения области, приведены в таблице 10.4.

Таблица 10.4

Константы задания стиля заполнения области

bsSolid	Сплошная заливка
bsClear	Область не закрашивается
bsHorizontal	Горизонтальная штриховка
bsVertical	Вертикальная штриховка
bsFdiagonal	Диагональная штриховка с наклоном линий вперед
bsBDiagonal	Диагональная штриховка с наклоном линий назад
bsCross	Горизонтально-вертикальная штриховка в клетку
bsDiagCross	Диагональная штриховка, в клетку

Для вывода текста на поверхности графического объекта используется метод *TextOut*.

Например, *Form1.Canvas.TextOut(x,y,Текст)*, где x,y – координаты точки графической поверхности, от которой выполняется вывод текста.

Шрифт, который используется для вывода текста, определяется значением свойства *Font*.

Методы вычерчивания графических примитивов

Линия. Вычерчивание прямой линии осуществляется методом *LineTo*, инструкция вызова которого в общем виде выглядит следующим образом:

Компонент.*Canvas.LineTo(x,y);*

Метод *LineTo* вычерчивает прямую линию от текущей позиции карандаша в точку с координатами, указанными при вызове метода.

Начальную точку линии можно задать, переместив карандаш в нужную точку графической поверхности при помощи метода *MoveTo(x,y)*.

Окружность, эллипс можно начертить, используя метод *Ellipse*. Вызов метода выглядит: Компонент.*Canvas.Ellipse(x1,y1, x2,y2)*, где x1,y1, x2,y2 – координаты прямоугольника, внутри которого вычерчивается эллипс или, если прямоугольник является квадратом, окружность.

Дуга. Вычерчивание дуги выполняет метод *Arc*, инструкция вызова которого имеет вид: Компонент.*Canvas.Ellipse(x1,y1, x2,y2, x3,y3, x4,y4)*, где x1,y1,x2,y2, – параметры, определяющие эллипс (окружность), частью которого является вычерчиваемая дуга;

$x3, y3$ – параметры, определяющие начальную точку дуги;
 $x4, y4$ – параметры, определяющие конечную точку дуги.

Прямоугольник вычерчивается методом *Rectangle*, вызов которого имеет вид: Компонент.Canvas.Rectangle($x1, y1, x2, y2$), где $x1, y1, x2, y2$ – координаты левого верхнего и правого нижнего углов прямоугольника. Прямоугольник со скругленными углами вычерчивается методом: *RounRect($x1, y1, x2, y2$)*.

2. Методика и порядок выполнения работы

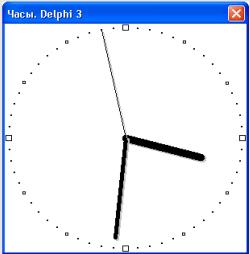
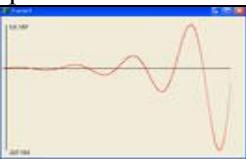
Задача 2.1. Начертить графические примитивы: линию, прямоугольник, эллипс.

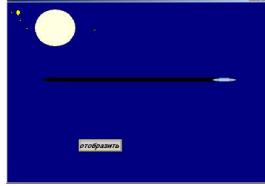
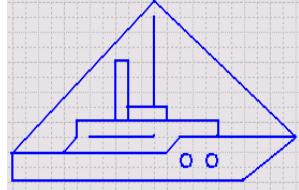
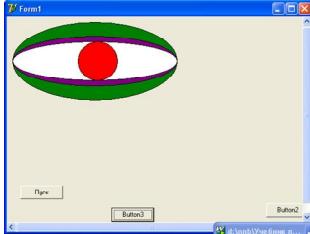
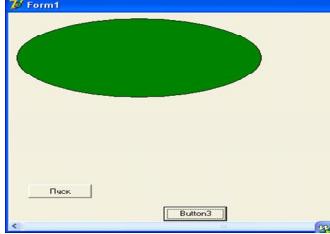
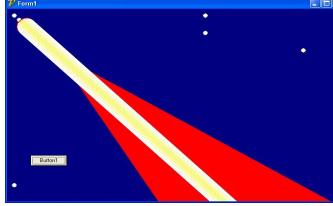
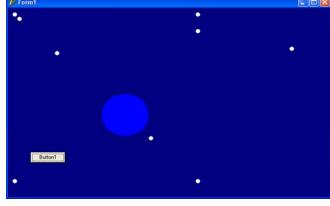
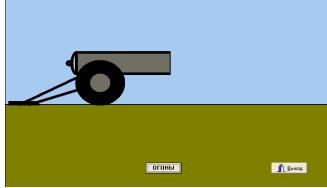
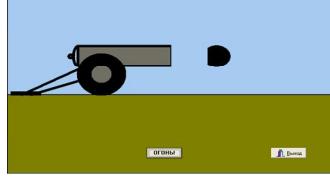
Задача 2.2. Создать приложение, реализующее отображение геометрических объектов, изображенных на рисунке.

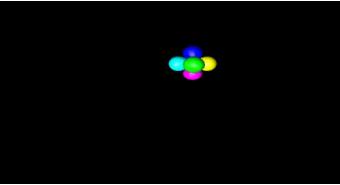
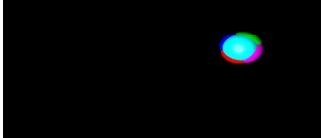
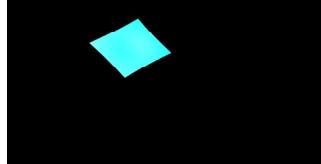
3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	Условие задачи	Вариант	Условие задачи
1.	 Написать программу стрелочные часы.	2.	 Отобразить на фоне звездного неба солнце с вращающейся вокруг него планетой.
3.	 Построить график затухающей функции.	4.	 Написать программу, которая будет рисовать на форме звезду по щелчку мыши. В месте щелчка должен быть центр звезды.

5.	 <p>Отобразить на фоне звездного неба летящую летающую тарелку.</p>	6.	 <p>Написать программу, моделирующую движущийся корабль.</p>
7.	Изобразить на форме космический глаз, периодически открывающийся и закрывающийся.		
			
8.	Отобразить на фоне звездного неба периодически пролетающую комету.		
			
9.	Отобразить орудие, из которого через каждые 10 сек. вылетает снаряд		
			

10.	Создать заставку «Метаморфоза»
	
	 

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Какие графические функции вы изучили в данной работе?
2. Что такое кисть, перо, зачем они нужны и их отличия друг от друга?
3. С помощью каких свойств можно определить толщину и цвет вычерченой линии?

4. Какое свойство используется для вывода графики на поверхность объекта?
5. Какое свойство объекта Canvas определяет цвет, стиль, толщину линий при использовании методов Line, Ellipse, Rectangle?
6. Какая функция обработки события используется при выводе графики?
7. Что такое кисть, перо, зачем они нужны и их отличия друг от друга?
8. Напишите и расшифруйте код для беспорядочного рисования различных геометрических фигур.
9. Напишите инструкцию для вывода на поверхность графического объекта текста. Как изменить его цвет и шрифт?
- 10.Какие свойства Canvas используются для рисования?
- 11.С какого угла канвы начинается система координат?
- 12.Что представляет собой наименьший элемент поверхности рисунка?
- 13.Как называется свойство пикселя, обозначающее цвет?
- 14.Какие свойства имеет кисть?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

11. СОЗДАНИЕ ПРИЛОЖЕНИЙ С МУЛЬТИПЛИКАЦИЕЙ В СРЕДЕ LAZARUS

Цель: приобретение навыков создания приложения с элементами мультипликации в интегрированной среде разработки ПО Lazarus.

Содержание:

- 1. Краткая теория.**
- 2. Методика и порядок выполнения работы.**
- 3. Индивидуальные задания.**
- 4. Содержание отчета и его форма.**
- 5. Контрольные вопросы и защита работы.**

1. Краткая теория

Под мультипликацией понимается движущийся и меняющийся рисунок. Рисунок может быть сформирован из графических примитивов (линий, окружностей, дуг, многоугольников и т. д.). Обеспечить перемещение рисунка можно следующим образом: вывести рисунок на экран, затем через некоторое время стереть его или нарисовать цветом фона, а затем снова вывести этот же рисунок, но уже на некотором расстоянии от его первоначального расположения. Впечатление равномерного движения достигается путем подбора времени между выводом и удалением рисунка.

Для задания некоторого интервала времени можно использовать невизуальный компонент *Timer* (таймер) на вкладке *System*. Свойства компонента *Timer* перечислены в таблице 11.1.

Таблица 11.1
Свойства компонента *Timer*

Свойство	Определяет
Name	Имя компонента.
Interval	Период генерации события OnTimer. Задается в миллисек.
Enabled	Разрешает (True) или запрещает(False) генерацию события OnTimer.

Компонент *Timer* является невизуальным и во время работы приложения не виден. Действия, которые необходимо выполнить по истечению интервала таймера, помещают в обработчик события *OnTimer* компонента *Timer*.

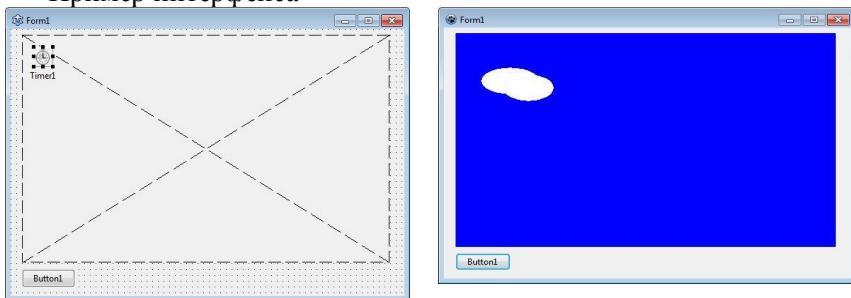
2. Методика и порядок выполнения работы

Задача 2.1. Линейное движение по однородному фону (движение тучки). Линейное движение по однородному фону является довольно простым в плане программной реализации. Достаточно за-крашивать объект цветом фона, изменять его координаты и прорисовывать в новом месте, повторяя эти действия через определенный интервал времени.

Для реализации анимации, помимо двух уже известных компонентов *TPaintBox* (поле для рисования) и *TButton* (кнопка запуска), понадобится компонент *TTimer* со вкладки *System*. Компонент *Timer* имеет единственное событие *OnTimer*, которое выполняется пока *Timer* включен с интервалом по времени, установленным в свойстве *Interval*.

Расположите компонент *Timer1* на форме. Установите его свойства *Timer1.Interval := 100* и *Timer1.Enabled := false*

Пример интерфейса



Пример программы:

```
unit Unit1;  
{$mode objfpc} {$H+}  
interface  
uses
```

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
StdCtrls,

```

ExtCtrls;
type
  { TForm1 }
TForm1 = class(TForm)
  Button1: TButton;
  PaintBox1: TPaintBox;
  Timer1: TTimer;
  procedure Button1Click(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
private
  { private declarations }
  x1, y1 : Integer;
public
  { public declarations }
  procedure Cloud (x,y:Integer; ColorCloud:TColor);
end;

var
  Form1: TForm1;
implementation
{$R *.lfm}
{ TForm1 }
procedure TForm1.Cloud(x,y:Integer;ColorCloud: TColor);
begin
  with PaintBox1.Canvas do begin
    Pen.style := psClear;
    Brush.Color := ColorCloud;
    Ellipse(x,y,x+80,y+40);
    Ellipse(x+30,y+10,x+100,y+50);
  end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Cloud(x1,y1,clBlue);
  x1:=x1+1;
  Cloud(x1,y1,clWhite);
end;

```

```

procedure TForm1.Button1Click(Sender: TObject);
begin
  x1:=0;
  y1:=50;
  Timer1.Interval:=100;
  PaintBox1.Canvas.Brush.Color:=clBlue;
  PaintBox1.Canvas.Rectangle(0,0,PaintBox1.Width, Paint-
Box1.Height);
  Timer1.Enabled := true;
end;
end.
```

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Создать фон и перемещаемый объект. Реализовать перемещение объекта относительно фона.

Вариант	Условие задачи
1.	Отобразить в приложении работу светофора.
2.	Фон – луг, объект – трактор.
3.	Фон – небо, объект – вертолет.
4.	Фон – небо, объект – парашютист.
5.	Фон – горы, объект – альпинист.
6.	Фон – луг с кромкой леса, объект – велосипедист.
7.	Фон – подводная часть моря, объект – подводная лодка.

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами задач приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Для чего служит компонент таймер?
2. С помощью какого свойства можно начать отсчет времени?
3. С помощью какого свойства можно задать период?
4. С помощью какого свойства можно задать период?
5. С помощью какого свойства картинка может выводиться на канве?
6. Какие объекты используются для хранения битовых образов фона?
7. За счет чего достигается эффект передвижения картинки?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

12. ИСПОЛЬЗОВАНИЕ БИТОВЫХ ОБРАЗОВ В СРЕДЕ LAZARUS. МУЛЬТИПЛИКАЦИЯ

Цель: Приобрести навыки создания и использования битовых образов, реализация перемещения одного сложного изображения на фоне другого в интегрированной среде разработки ПО Lazarus.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория

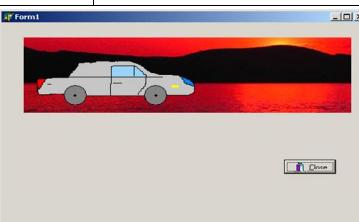
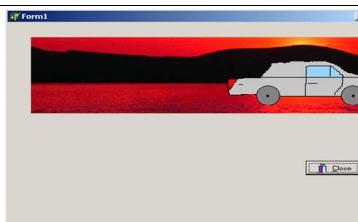
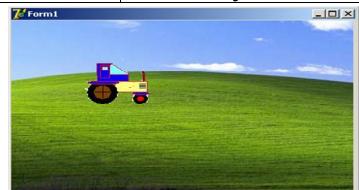
Эффект перемещения картинки может быть создан путем периодической перерисовки картинки с некоторым смещением относительно ее прежнего положения. При этом предполагается, что перед выводом картинки в новой точке сначала удаляется предыдущее изображение. Удаление картинки может быть выполнено путем перерисовки всей фоновой картинки или только той ее части, которая перекрыта битовым образом движущегося объекта. Картина может выводиться на канве применением метода *Draw* к свойству *Canvas* компонента (например, *Image*), а стирается путем копирования методом *CopyRect* нужной части фона из буфера на поверхность компонента (например, *Image*).

Для хранения битовых образов (картинок) фона и перемещающегося объекта, а также копии области фона, перекрываемой изображением объекта, используются объекты типа *TBitMap*. Программные объекты, используемые для копирования в буфер области фона, и на которую будет наложено изображение движущегося объекта и которая в последствии должна быть восстановлена из буфера, имеют тип *Trect* (прямоугольник). Для заполнения программных объектов типа *Trect* используется функция *Bounds(x,y,W,H)*, где *x,y* – координата начальной точки, *W,H* – ширина и высота прямоугольной **области**.

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Создать фон и перемещаемый объект. Реализовать перемещение объекта относительно фона.

Вариант	Условие задачи
1.	Фон – дорожный перекресток, объект – средство передвижения
	
	
2.	Фон – луг, объект – трактор
	
	
3.	Фон – небо, объект – вертолет
	
	
4.	Фон – небо, объект – парашютист.
5.	Фон – горы, объект – альпинист.
6.	Фон – луг с кромкой леса, объект – велосипедист.
7.	Фон – подводная часть моря, объект – подводная лодка.

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б):

название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Для чего служит компонент таймер?
2. С помощью какого свойства можно начать отсчет времени?
3. С помощью какого свойства можно задать период?
4. С помощью какого свойства картинка может выводиться на канве?
5. Какие объекты используются для хранения битовых образов фона?
6. За счет чего достигается эффект передвижения картинки?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

13. СТРАНИЦА ДИАЛОГИ

Цель: Приобрести навыки создания и использования диалогов в интегрированной среде разработки программного обеспечения Lazarus.

Содержание:

- | | |
|----|---------------------------------------|
| 1. | Краткая теория. |
| 2. | Методика и порядок выполнения работы. |
| 3. | Индивидуальные задания. |
| 4. | Содержание отчета и его форма. |
| 5. | Контрольные вопросы и защита работы. |

1. Краткая теория

Знакомство со стандартными диалогами Windows.

На странице Dialogs представлены компоненты для вызова стандартных диалогов Windows: *TOpenDialog* – открыть файл, *TSaveDialog* – сохранить файл, *TFontDialog* – настройка шрифта, *TColorDialog* – выбор цвета, *TPrintDialog* – печать, *TPrinterSetupDialog* – настройки принтера, *TFindDialog* – поиск строки, *ReplaceDialog* – поиск с заменой.

Работа со стандартными диалоговыми окнами осуществляется в 3 этапа:

1. На форму помещается соответствующий компонент и осуществляется настройка его свойств. Сам компонент не виден в момент работы программы (невизуальный компонент), видно лишь создаваемое им стандартное окно.

2. Осуществляется вызов стандартного для диалогов метода *Execute*, который создает и показывает на экране диалоговое окно. *Execute* – логическая функция, возвращает в программу *True*, если результат диалога с пользователем был успешным.

3. Анализ результата метода *Execute*, использование введенных с помощью диалогового окна данных – имени файла, настроек принтера, выбранного шрифта и т. д.

Основные компоненты страницы *Dialogs*:

1. *TColorDialog*. Компонент создает и обслуживает стандартное окно выбора цвета. Свойство *Color* содержит стандартный цвет в окне диалога.

Пример использования диалога в программе может быть следующим:

If ColorDialog1.Execute (вызов окна диалога)

Then Memo1.Color:=ColorDialog1.Color; (memo1 залить цветом, выбранным в окне диалога «Цвет»)

2. *TFontDialog*. Компонент создает и обслуживает стандартное окно выбора шрифта. Свойство компонента *Font* содержит выбранные параметры шрифта в окне диалога.

Пример использования диалога в программе может быть следующим:

If FontDialog1.Execute (вызов окна диалога)

Then Memo1.Font:=FontDialog1.Font; (задать стили шрифта в memo1, выбранных в окне диалога «Шрифт»)

3. *TOpenDialog* и *TSaveDialog*. Диалоги открытия и сохранения файла.

При открытии или сохранении файла пользователь в окне диалога указывает путь и имя файла, которое возвращается свойством *FileName:string*.

При открытии файла пользователь может ввести произвольное имя и, следовательно, указать несуществующий файл. Чтобы этого избежать, можно проверить существующий файл функцией *FileExists:boolean*.

Свойство *Filter:string* используется для фильтрации (отбора) файлов, показываемых в диалоговом окне. Это свойство можно устанавливать в программе, например,

OpenDialog.Filter:='текстовые файлы/.txt/файлыObject Pascal/*.pas';*

(задает две маски отбора файлов с расширением *.txt* и *.pas*)

или в окне *Инспектора Объектов* при выборе свойства *Filter*, открывающего окно специального редактора.

Свойство *Title:string* задает заголовок диалогового окна.

Установить начальный каталог окна диалога можно с помощью свойства *InitialDir: string* в программе или в окне *Инспектора Объектов*.

В свойстве *DefaultExt:string* указывается расширение, присваиваемое сохраняемому файлу по умолчанию (если пользователь не указал другое).

Для некоторых компонент характерны методы:

- *SavetoFile(имя файла:string)* – сохраняет информацию в виде файла с указанным именем;
- *LoadFromFile(имя файла:string)* – считывает содержимое файла.

Пример использования компоненты *OpenDialog* для чтения информации с файла и записи ее в *Memo1*.

Begin

OpenDialog1.InitialDir:='C:\Мои документы'; (устанавливаем начальный каталог в окне диалога «Открытие файла»)

OpenDialog1.Filter:='текстовые файлы|.txt|файлы Object Pascal|*.pas';* (настраиваем диалог на отбор текстовых файлов)

If OpenDialog1.Execute and FileExists(OpenDialog1.FileName)
(выполняем диалог и проверяем существование выбранного диалогом файла)

Then Memo1.Lines.LoadFromFile(OpenDialog1.FileName); (считываем содержимое файла в *Memo1*)

End;

Пример использования компоненты *SaveDialog* для записи содержимого компоненты *Memo1* в файл.

Begin

SaveDialog1.InitialDir:='C:\Мои документы';

SaveDialog1.Filter:='текстовые файлы|.txt|файлы Object Pascal|*.pas';*

SaveDialog1.DefaultExt:='txt'; (указываем расширение файла по умолчанию)

If SaveDialog1.Execute

Then Memo1.Lines.SaveToFile(SaveDialog1.FileName);

(записываем содержимое *Memo1* в файл)

End;

4. *TOpenPictureDialog* и *TSavePictureDialog* – диалоги открытия и сохранения изображений. Специализированные диалоги для открытия и сохранения графических файлов отличаются от *TOpenDialog* и *TSaveDialog* двумя обстоятельствами:

- В них предусмотрены стандартные фильтры для выбора графических файлов (с расширением *.bmp*, *.ico*, *.wmt*, *.emf* и др.), т. е. отсутствует свойство *Filter*.

- В окна диалога включена панель для предварительного просмотра выбиравшегося файла. Считывать содержимое графического файла можно только в компоненты, работающие с графикой, рисунками (*Image* и др.).

Вызов диалогов происходит программно, например:

```
OpenPictureDialog1.InitialDir:='C:\Мои документы';
```

```
If OpenPictureDialog1.Execute and FileExists (OpenPictureDialog1.FileName)
```

```
Then     Image1.Picture.LoadFromFile      (OpenPictureDialog1.FileName);
```

Дополнительные сведения:

Image(Additional) – изображение картинок. Изображения содержатся в свойстве *Picture*.

MainMenu – горизонтальное меню. Свойство *Items* позволяет определить пункты меню.

OpenDialog – компонент представляет собой стандартное диалоговое окно выбора и открытия файлов.

OpenPictureDialog – предназначен для открытия графических файлов.

Основные свойства:

- *FileName* – содержит имя выбранного файла;
- *Files* – содержит список имен выделенных файлов;
- *Filter* – позволяет задавать фильтр для файлов, которые будут отображаться в диалоговом окне;
- *InitialDir* – позволяет установить начальный каталог поиска в *Инспекторе объектов*.

FontDialog – предназначен для настройки шрифта и его характеристик.

Основные свойства:

- *Font* – результат изменения характеристик шрифта;
- *MaxFontSize* – установка максимального размера шрифта;
- *MinFontSize* – установка минимального размера шрифта.

ColorDialog – предоставляет возможность выбора определенного цвета из палитры.

Основным свойством является свойство *Color* – содержит данные о выбранном пользователем цвете.

2. Методика и порядок выполнения работы

1. Разместить компоненты *Memo* (текстовая область) *Image* (графика).

2. Создать горизонтальное меню с пунктами:

- Текст: цвет, шрифт, открыть, сохранить (для работы с текстовой областью).
- Графика: открыть (для размещения файла с графическими рисунками).
- Окно: выход, о программе (*ShowMessege*).

3. Написать код для каждого пункта *Memo* (реакцию на событие *OnClick*). Не забыть разместить на форме соответствующие компоненты.

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

На форме расположить следующие компоненты: *Panel*, *Image*, *RichEdit*, , *OpenPictureDialog*, *FontDialog*, *Open Dialog*, *ColorDialog*, 3 *PopupMenu*, *BitBtn*.

PopupMenu компонента *Image* должно содержать пункт вызова окна *OpenPictureDialog*.

PopupMenu компонента *RichEdit* должно содержать пункты вызова окна *FontDialog* и *Open Dialog*.

PopupMenu формы должно содержать пункт вызова окна *ColorDialog*.

При выборе изображения с помощью компонента *OpenPictureDialog* в компоненте *Image* должно отображаться выбранное изображение, одновременно с этим в компоненте *RichEdit* должна отображаться информация, соответствующая выбранному изображению. И наоборот, при выборе информационного файла с помощью компонента *OpenDialog* в компоненте *RichEdit* должна выводится информация из выбранного файла и одновременно в *Image* загружаться изображение, соответствующее выбранному текстовому файлу.

Кроме того, с помощью ColorDialog должно быть предусмотрено смена цвета формы, а с помощью FontDialog, изменение настроек RichEdit. Каждый визуальный компонент должен быть снабжен всплывающей подсказкой.

Напишите программу, использующую 8 изображений.

Вариант	Условие задачи (задание)
1.	Картинная галерея
2.	Автомобили
3.	Компьютерные игры
4.	Телефоны
5.	Семь чудес света
6.	Семь чудес России
7.	Великие ученые
8.	Столицы мира

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Стандартные диалоги Windows.
2. Основные этапы работы со стандартными диалоговыми окнами.
3. Основные компоненты страницы *Dialogs*.

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

14. СОСТАВНЫЕ ДАННЫЕ НЕОДНОРОДНОЙ СТРУКТУРЫ. ФИКСИРОВАННЫЕ ЗАПИСИ

Цель: Приобрести навыки создания алгоритмизации и программирования задач ввода и вывода записей, их обработки, использования записей как структурных элементов баз данных.

Содержание:

1. Краткая теория.
2. Методика и порядок выполнения работы.
3. Индивидуальные задания.
4. Содержание отчета и его форма.
5. Контрольные вопросы и защита работы.

1. Краткая теория

Компонент TStringGrid в Lazarus

Компонент TStringGrid – на вкладке компонентов **Additional**.

Настройка таблицы StringGrid1 в окне Инспектор объектов

1.	Убрать заголовки строк	FixedCols = 0
2.	Убрать заголовки колонок	FixedRows = 0
3.	Установить количество колонок	ColCount = 5
4.	Установить количество строк	RowCount = 1
5.	Установить ширину всех колонок	DefaultColWidth = 30
6.	Установить высоту всех строк	DefaultRowHeight = 30
7.	Разрешить редактирование ячеек	Options/goEditing = True

Отдельно взятые массивы, множество или файл всегда включают элементы одинакового типа. Если необходимо логически объединить компоненты разных типов, то применяется тип *запись*.

Запись Паскаля – структурированный комбинированный тип данных, состоящий из фиксированного числа компонент (полей) разного типа.

Например, анкетные данные о студенте вуза могут быть представлены в виде информационной структуры.

Анкета студента					
ФИО	Пол	Дата рождения	Адрес	Курс	Группа

Такая структура называется двухуровневым деревом. В Паскале эта информация может храниться в одной переменной типа *record* (запись).

Запись может быть объявлена в разделе *type*:

```
type <имя_типа>=record
  <имя_поля1>: тип;
  <имя_поля2>: тип;
  .....
  <имя_поля К>: тип
end;
```

где *record* – служебное слово, а *<имя_типа>* и *<имя_поля>* – правильные идентификаторы языка. Поля записи могут иметь любой тип, кроме файла, в частности, сами могут быть записями.

Описание анкеты студента в Паскале будет выглядеть так:

```
type anketa=record
  fio: string[45];
  pol: char;
  dat_r: string[8];
  adres: string[50];
  curs: byte;
  grupp: string[3];
end;
```

Такая запись Паскаля, так же как и соответствующее ей дерево, называется двухуровневой.

После того, как определен тип записи Паскаля, можно определять переменную этого типа. Переменная определяется путем задания ее идентификатора и указания типа.

```
var student: anketa;
```

Обращение к записи и к элементам записи. Обращение к записи в целом допускается только в операторах присваивания, где слева и справа от знака присваивания используются имена записей одинакового типа. Элементы записи называются *полями*, а обращение к ним производится через использование их имен – *идентификаторов полей*. Практически, поля записи обрабатываются точно

так же, как и любые другие переменные. Но в отличие от обычной переменной имена полей должны предваряться ссылкой на идентификатор записи Паскаля и отделяться от него точкой:

<имя_записи>.<имя_поля>

Такая запись называется уточняющий или составной идентификатор, а <имя_записи> – префикс. Например, чтобы обратиться к полю *curs* переменной *student*, необходимо указать следующее составное имя: *student.curs:=3*;

Использование полей записи Паскаля в выражениях и условиях идентично использованию обычных переменных.

Операции над записями. Единственная операция, которую можно произвести над однотипными записями Паскаля – это присваивание. Все другие операции, в том числе ввод и вывод, производятся над отдельными полями записи.

Префикс – обязательная предшествующая часть составного идентификатора для имен полей в структуре типа запись Паскаля. Если в программе необходимо последовательно обращаться к полям одной и той же записи, то префикс можно не указывать. Такая возможность реализуется при помощи *оператора присоединения*, который в общем виде выглядит так: **with** <имя_записи> **do** <оператор>.

Он позволяет, один раз указав имя переменной типа "запись" после слова *with*, работать в пределах одного оператора (простого или составного) с именами полей как с обычными переменными, т.е. не писать громоздких составных имен.

Оператор представляет собой область действия оператора присоединения, в пределах которой можно не использовать составные имена. Например, ввод записей с использованием оператора присоединения будет выглядеть так:

```
for I:=1 to 100 do
  with student[I] do
    begin
      write('введите сведения о', I, '-м студенте');
      write('введите фамилию, имя и отчество');readln (fio);
      write('введите дату рождения'); readln (dat_r);
      write('введите курс'); readln(curs);
      write('введите группу');readln (grupp);
    end;
```

2. Методика и порядок выполнения работы

Задача 2.1. Классный журнал 1. Вывести в таблицу сведения из классного журнала: номер записи, фамилию ученика, оценки по информатике, физике и математике.

Требования к выполнению: использовать тип запись.

Пример интерфейса

N	Фамилия	Информатика	Физика	Математика
1	Бабурин	3	4	5
2	Васильев	5	4	3
3	Дынкинъя	4	3	5

Пример программы

```
var  
  Form1: TForm1;  
  
// Глобальные константы.  
const  
  N_MAX = 3;  
  INDEX_COL = 0;  
  TITLE_ROW = 0;  
  
procedure TForm1.Button1Click(Sender: TObject);  
type  
  t_student = record  
    name : string[25];  
    inf : 0..5;  
    phys : 0..5;  
    math : 0..5;  
  end;  
var  
  stud1,  
  stud2,  
  stud3 : t_student;
```

```
i : Byte;
begin
stud1.name := 'Бабурин';
stud1.inf := 3;
stud1.phys := 4;
stud1.math := 5;
with stud2 do begin
name := 'Васильев';
inf := 5;
phys := 4;
math := 3;
end;
with stud3 do begin
name := 'Дьяконова';
inf := 4;
phys := 3;
math := 5;
end;
i := 1;
StringGrid1.Cells[1, i] := stud1.name;
StringGrid1.Cells[2, i] := IntToStr(stud1.inf);
StringGrid1.Cells[3, i] := IntToStr(stud1.phys);
StringGrid1.Cells[4, i] := IntToStr(stud1.math);
i := 2;
with stud2 do begin
StringGrid1.Cells[1, i] := name;
StringGrid1.Cells[2, i] := IntToStr(inf);
StringGrid1.Cells[3, i] := IntToStr(phys);
StringGrid1.Cells[4, i] := IntToStr(math);
end;
i := 3;
with stud3 do begin
StringGrid1.Cells[1, i] := name;
StringGrid1.Cells[2, i] := IntToStr(inf);
StringGrid1.Cells[3, i] := IntToStr(phys);
StringGrid1.Cells[4, i] := IntToStr(math);
end;
end;
```

```

procedure TForm1.FormCreate(Sender: TObject);
var
  i : 0..N_MAX;
begin
  for i := 1 to N_MAX do begin
    StringGrid1.Cells[INDEX_COL, i] := IntToStr(i);
  end;
  StringGrid1.Cells[0, TITLE_ROW] := 'N';
  StringGrid1.Cells[1, TITLE_ROW] := 'Фамилия';
  StringGrid1.Cells[2, TITLE_ROW] := 'Информатика';
  StringGrid1.Cells[3, TITLE_ROW] := 'Физика';
  StringGrid1.Cells[4, TITLE_ROW] := 'Математика';
  StringGrid1.ColWidths[1]:=110;
  StringGrid1.ColWidths[2]:=110;
  StringGrid1.ColWidths[3]:=60;
  StringGrid1.ColWidths[4]:=110;
end;

```

Задача 2.2. Классный журнал 2. Для каждого ученика в предыдущей задаче найти среднее арифметическое и сумму оценок, вывести их в таблицу.

Требования к выполнению. Использовать массив записей, для среднего арифметического использовать вещественный тип данных и выводить с точностью до десятых.

Пример интерфейса

N	Фамилия	Информатика	Физика	Математика	Сумма	Среднее
1	Бабурин	3	4	5	12	4,0
2	Васильев	5	4	3	12	4,0
3	Дьяконова	4	3	5	12	4,0

Пример программы

var

Form1: TForm1;

```

// Глобальные константы.
const
  N_MAX = 3;
  N_SCORES = 3;
  INDEX_COL = 0;
  TITLE_ROW = 0;
  SUM_COL = 5;
  AVG_COL = 6;
type
  t_student = record
    name : string[25];
    inf : 0..5;
    phys : 0..5;
    math : 0..5;
  end;
var
  stud : array [1..N_MAX] of t_student;

procedure TForm1.Button1Click(Sender: TObject);
var
  i : Byte;
begin
  for i := 1 to N_MAX do begin
    StringGrid1.Cells[1, i] := stud[i].name;
    StringGrid1.Cells[2, i] := IntToStr(stud[i].inf);
    StringGrid1.Cells[3, i] := IntToStr(stud[i].phys);
    StringGrid1.Cells[4, i] := IntToStr(stud[i].math);
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
const
  N = 3;
  M = 1;
var
  i : 1..N_MAX;
  sum : Byte;
  avg : double;
begin

```

```

for i := 1 to N_MAX do begin
with stud[i] do begin
sum := inf + phys + math;
avg := sum/N_SCORES;
end;
StringGrid1.Cells[SUM_COL, i] := IntToStr(sum);
StringGrid1.Cells[AVG_COL, i] := FloatToStrF(avg, ffFixed, N, M);
end;
end;
procedure TForm1.FormCreate(Sender: TObject);
var
i : 0..N_MAX;
begin
for i := 1 to N_MAX do begin
StringGrid1.Cells[INDEX_COL, i] := IntToStr(i);
end;
StringGrid1.Cells[0, TITLE_ROW] := 'N';
StringGrid1.Cells[1, TITLE_ROW] := 'Фамилия';
StringGrid1.Cells[2, TITLE_ROW] := 'Информатика';
StringGrid1.Cells[3, TITLE_ROW] := 'Физика';
StringGrid1.Cells[4, TITLE_ROW] := 'Математика';
StringGrid1.Cells[5, TITLE_ROW] := 'Сумма';
StringGrid1.Cells[6, TITLE_ROW] := 'Среднее';
StringGrid1.ColWidths[1]:=110;
StringGrid1.ColWidths[2]:=110;
StringGrid1.ColWidths[3]:=60;
StringGrid1.ColWidths[4]:=110;
i := 1;
stud[i].name := 'Бабурин';
stud[i].inf := 3;
stud[i].phys := 4;
stud[i].math := 5;
i := 2;
with stud[i] do begin
name := 'Васильев';
inf := 5;
phys := 4;
math := 3;
end;

```

```
i := 3;
with stud[i] do begin
  name := 'Дьяконова';
  inf := 4;
  phys := 3;
  math := 5;
end;
end;
```

Задача 2.3. Классный журнал 3. Найти ученика с наибольшим и наименьшим средним баллом.

Требования к выполнению: Использовать массив записей.

Пример интерфейса

N	Фамилия	Информатика	Физика	Математика	Сумма	Среднее
1	Бабурин	3	4	3	10	3,3
2	Васильев	3	2	3	8	2,7
3	Дьяконова	5	4	5	14	4,7
4	Иванов	2	2	3	7	2,3
5	Миронов	3	5	3	11	3,7

Показать Вычислить Найти

Пример программы

```
var
  Form1: TForm1;
  // Глобальные константы.
  const
    N_MAX = 5;
    N_SCORES = 3;
    INDEX_COL = 0;
    TITLE_ROW = 0;
    SUM_COL = 5;
    AVG_COL = 6;
    MIN_COLOR = clAqua;
    MAX_COLOR = clYellow;
    SHIFT_TEXT_X = 3;
    SHIFT_TEXT_Y = 3;
  // Глобальный тип.
```

```

type
  t_student = record
    name : string[25];
    inf : 0..5;
    phys : 0..5;
    math : 0..5;
    sum : Byte;
    avg : double;
  end;
// Глобальные массивы.
var
  stud : array [1..N_MAX] of t_student;

procedure TForm1.Button1Click(Sender: TObject);
var
  i : Byte;
begin
  for i := 1 to N_MAX do begin
    StringGrid1.Cells[1, i] := stud[i].name;
    StringGrid1.Cells[2, i] := IntToStr(stud[i].inf);
    StringGrid1.Cells[3, i] := IntToStr(stud[i].phys);
    StringGrid1.Cells[4, i] := IntToStr(stud[i].math);
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
const
  N = 3;
  M = 1;
var
  i : 1..N_MAX;
begin
  for i := 1 to N_MAX do begin
    with stud[i] do begin
      sum := inf + phys + math;
      avg := sum/N_SCORES;
      StringGrid1.Cells[SUM_COL, i] := IntToStr(sum);
      StringGrid1.Cells[AVG_COL, i] := FloatToStrF(avg, ffFixed, N, M);
    end;
  end;
end;

```

```

end;
end;
end;
procedure TForm1.Button3Click(Sender: TObject);
var
  i, min, k_min, max, k_max : Byte;
  ts : string;
  rect : TRect;

begin
  k_min := 1;
  min := stud[k_min].sum;
  k_max := 1;
  max := stud[k_max].sum;
  for i := 2 to N_MAX do begin
    if min > stud[i].sum then begin
      k_min := i;
      min := stud[i].sum;
    end;
    if max < stud[i].sum then begin
      k_max := i;
      max := stud[i].sum;
    end;
  end;
// ....:: MIN MIN MIN ::....
  ts := StringGrid1.Cells[INDEX_COL, k_min];
  rect := StringGrid1.CellRect(INDEX_COL, k_min);
  StringGrid1.Canvas.Brush.Color := MIN_COLOR;
  StringGrid1.Canvas.FillRect(rect);
  StringGrid1.Canvas.TextOut(rect.Left+SHIFT_TEXT_X,
rect.Top+SHIFT_TEXT_Y, ts);

// ....:: MAX MAX MAX ::....
  ts := StringGrid1.Cells[INDEX_COL, k_max];
  rect := StringGrid1.CellRect(INDEX_COL, k_max);
  StringGrid1.Canvas.Brush.Color := MAX_COLOR;
  StringGrid1.Canvas.FillRect(rect);

```

```
StringGrid1.Canvas.TextOut(rect.Left+SHIFT_TEXT_X,  
rect.Top+SHIFT_TEXT_Y, ts);  
end;  
  
procedure TForm1.FormCreate(Sender: TObject);  
var  
  i : 0..N_MAX;  
begin  
  for i := 1 to N_MAX do begin  
    StringGrid1.Cells[INDEX_COL, i] := IntToStr(i);  
  end;  
  StringGrid1.Cells[0, TITLE_ROW] := 'N';  
  StringGrid1.Cells[1, TITLE_ROW] := 'Фамилия';  
  StringGrid1.Cells[2, TITLE_ROW] := 'Информатика';  
  StringGrid1.Cells[3, TITLE_ROW] := 'Физика';  
  StringGrid1.Cells[4, TITLE_ROW] := 'Математика';  
  StringGrid1.Cells[5, TITLE_ROW] := 'Сумма';  
  StringGrid1.Cells[6, TITLE_ROW] := 'Среднее';  
  StringGrid1.ColWidths[1]:=110;  
  StringGrid1.ColWidths[2]:=110;  
  StringGrid1.ColWidths[3]:=60;  
  StringGrid1.ColWidths[4]:=110;  
  i := 1;  
  stud[i].name := 'Бабурин';  
  stud[i].inf := 3;  
  stud[i].phys := 4;  
  stud[i].math := 3;  
  i := 2;  
  with stud[i] do begin  
    name := 'Васильев';  
    inf := 3;  
    phys := 2;  
    math := 3;  
  end;  
  i := 3;  
  with stud[i] do begin  
    name := 'Дьяконова';  
    inf := 5;
```

```

phys := 4;
math := 5;
end;
i := 4;
with stud[i] do begin
name := 'Иванов';
inf := 2;
phys := 2;
math := 3;
end;
i := 5;
with stud[i] do begin
name := 'Миронов';
inf := 3;
phys := 5;
math := 3;
end;
end;

```

3. Индивидуальные задания

Каждый студент выполняет задания по варианту. №варианта = № студента в Журнале лабораторных работ.

Задание 3.1. Разработать программу в среде программирования Lazarus. Для каждой задачи создать интерфейс, соответствующий условию.

Вариант	База данных	Критерий отбора
1.	«Абитуриенты» (оценки по 3-м экзаменам, средний балл)	Средний балл не ниже 4,5
2.	«Отдел кадров университета» (фамилия, имя, отчество, стаж педагогической деятельности)	Стаж педагогической деятельности более 10 лет
3.	«Научно-техническая библиотека» (фамилия автора, название книги, город, издательство, год выпуска, тематика)	Книги по программированию
4.	«Легковые автомобили» (марка, цвет, год выпуска, цена)	Стоимость менее 500 тыс. руб.

5.	«Администратор железнодорожной кассы» (№ поезда, пункты отправления и прибытия, время отправления и прибытия)	Поезда, следующие до Ставрополя
6.	«Магазин по продаже ПК» (процессор, ОЗУ, винчестер, цена)	Цена менее 10 тыс. руб
7.	«Товары на складе» (название, количество, стоимость 1 единицы товара)	Товары, суммарная стоимость которых более 20 тыс. руб.
8.	«Спортсмены» (фамилия, имя, разряд, рост, вес)	Спортсмены, рост которых превышает 175 см
9.	«Телефонный справочник» (фамилия, имя, отчество, адрес, № телефона)	Абоненты, № телефона которых начинается на «22»
10.	«Сведения о родственниках» (фамилия, имя, отчество, дата рождения, № телефона)	Родственники, родившиеся в январе
11.	«Домашняя библиотека» (фамилия, имя, отчество автора книги, название, год издания, тематика)	Книги автора «Толстой»
12.	«Кондитерская» (название торта, вид (бисквит, слоеный, песочный, калорийность, срок годности, цена)	Вид торта – бисквитный

4. Содержание отчета и его форма

Отчет должен иметь форму согласно оформлению простого реферата. Титульный лист должен включать (Приложение Б): название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;

- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами приведенными в лабораторной работе.

5. Контрольные вопросы и защита работы

Контрольные вопросы

1. Чем отличается тип «запись» от других структурированных типов?
2. Как определяется тип записи?
3. Что такое поле записи?
4. Какие типы может иметь поле записи?
5. Могут ли поля записи быть разных типов?
6. Как обратиться к отдельному полю записи?
7. Какие операции возможны над данными типа «запись»?
8. Что такое «оператор присоединения»? В каких целях он используется?
9. Как заполнить массив записей?

Защита работы

Защита работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Аппаратура и материалы. Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 1800 МГц и выше, оперативная память – не менее 512 Мбайт, свободное дисковое пространство – не менее 500 Мбайт, монитор (число цветов – 256) с диагональю не менее 17". Программное обеспечение – операционная система **WINDOWS NT/XP/2007**, пакет **Lasarus (Free Pascal)**.

Указания по технике безопасности. Техника безопасности при выполнении лабораторных работ совпадает с общепринятой для пользователей ПК, в частности: самостоятельно не производить ремонт ПК, установку и удаление программного обеспечения; в случае неисправности ПК сообщить об этом обслуживающему персоналу лаборатории; соблюдать правила техники безопасности при работе с электрооборудованием; содержать рабочее место в чистоте.

Содержание отчета и его форма. Отчет должен иметь форму простого реферата. Титульный лист должен включать: название дисциплины, название лабораторной работы, фамилию и инициалы сдающего студента, номер группы, фамилию и инициалы принимающего преподавателя.

Основная часть лабораторной работы должна содержать:

- блок-схему алгоритма задачи;
- листинг разработанного программного приложения;
- скриншоты программного приложения;
- ответы на вопросы, предложенные в каждом варианте;
- выводы по проделанной работе.

Задание должно быть выполнено в соответствии с примерами лабораторной работы. Отчет о выполнении лабораторной работы в письменном и электронном (отчет + файлы проекта) видах сдается преподавателю.

Защита лабораторной работы заключается:

- в ознакомлении с теоретической частью, приведенной в лабораторной работе, и оформлении отчета в соответствии с подпунктом «Содержание отчета и его форма»;
- в ответах на контрольные вопросы по лабораторной работе;
- в ответах на дополнительные вопросы по лабораторной работе и дисциплине «Объектно-ориентированное программирование».

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Федоренко Ю. П., Алгоритмы и программы на C++Builder. Издательство: ДМК Пресс, 2010. – 544 с. Электронное издание: режим доступа: www.knigafund.ru/books/54473.
2. Кауфман В. Ш. Языки программирования. Концепции и принципы. ДМК Пресс, 2010. – 450 с. Электронное издание: режим доступа: www.knigafund.ru/books/54502.
3. Бескоровайный И. В. Азбука Delphi: программирование с нуля. Сибирское университетское издательство, 2009. – 112 с. Электронное издание: режим доступа: www.knigafund.ru/books/84433.
4. Сигал И. Х., Иванова А. П. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы: учебное пособие. Издательство: ФИЗМАТЛИТ, 2009. – 304 с. Электронное издание: режим доступа: www.knigafund.ru/books/84455.
5. Гаврилова И. В. Разработка приложений: учебное пособие. Издательство: ФЛИНТА, 2012. – 242 с. Электронное издание: режим доступа: www.knigafund.ru/books/84459.
6. Журавлев Т. Ю. Системное и прикладное программное обеспечение: учебное пособие. Издательство Московского государственного открытого университета, 2010. – 144 с. Электронное издание: режим доступа: www.knigafund.ru/books/84444.
7. Иванова Н. Ю., Маняхина В.Г. Системное и прикладное программное обеспечение: учебное пособие. Издательство: МПГУ, 2011. – 201 с.
8. Электронное издание: режим доступа: www.knigafund.ru/books/84489.
9. Нагинаев В. Н. Основы алгоритмизации и программирования на языке C++: Учебное пособие. Издательство: МИИТ, 2010. – 173 с. Электронное издание: режим доступа: www.knigafund.ru/books/84461.
10. Зеленяк О. П. Практикум программирования на Turbo Pascal. Задачи, алгоритмы и решения. Издательство: ДиаСофтЮП; ДМК Пресс, 2009. – 311 с. Электронное издание: режим доступа: www.knigafund.ru/books/84473.
11. Авдеев В. А. Интерактивный практикум по компьютерной схемотехнике на Delphi. Издательство: ДМК Пресс, 2011. – 360 с. Электронное издание: режим доступа: www.knigafund.ru/books/84489.

12. Потопахин В. В. Искусство алгоритмизации. Издательство: ДМК Пресс, 2011. – 320 с. Электронное издание: режим доступа: www.knigafund.ru/books/84544.
13. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Издательство: ДМК Пресс, 2009. – 365 с. Электронное издание: режим доступа: www.knigafund.ru/books/84568.
14. Free Pascal и Lazarus: учебник по программированию / Е. Р. Алексеев, О. В. Чеснокова, Т. В. Кучер. М.: ALT Linux, 2010. – 438 с. Электронное издание: режим доступа: www.knigafund.ru/books/84537.
15. Алексеев Е. Р., Чеснокова О. В. Турбо Паскаль 7.0. – М.:НТПресс, 2008. – 320 с. Электронное издание: режим доступа: www.knigafund.ru/books/84577.
16. Костюкова Н. И. Программирование на языке Си. Методические рекомендации и задачи по программированию. Сибирское университетское издательство, 2009. –178 с. Электронное издание: режим доступа: www.knigafund.ru/books/84479.
17. Белладжио Д., Миллиган Т. Стратегия управления конфигурацией программного обеспечения с использованием IBM Rational ClearCase. Издательство: ДМК Пресс, 2010. –382 с. Электронное издание: режим доступа: www.knigafund.ru/books/106209.

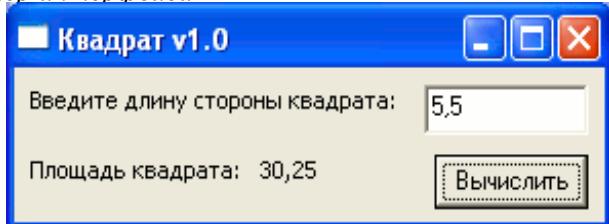
ПРИЛОЖЕНИЯ

Приложение А

Примеры выполнения заданий (упражнения)

Упражнение 1. Вычислить площадь квадрата $S=a^2$

Пример интерфейса



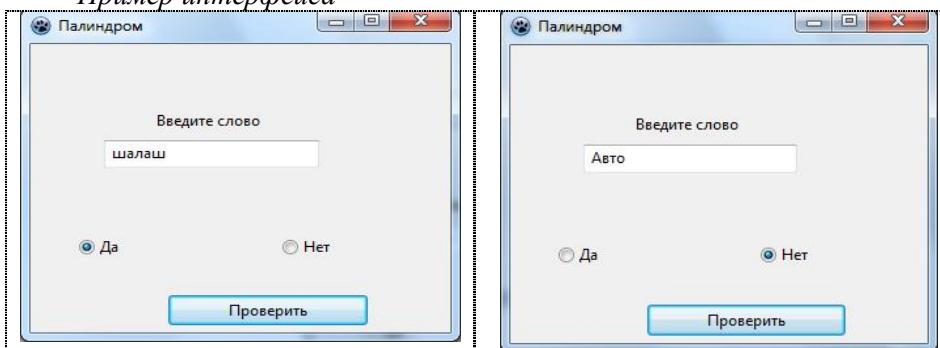
Код программы

```
procedure TForm1.Button1Click(Sender: TObject);
const
  W = 0;
  D = 2;
var
  a, s : double;
begin
  a := StrToFloat(Edit1.Text);
  s := a * a;
  Label3.Caption := FloatToStrF(s, ffFixed, W, D);
end;
```

Упражнение 2. Определить является ли симметричным данное

СЛОВО

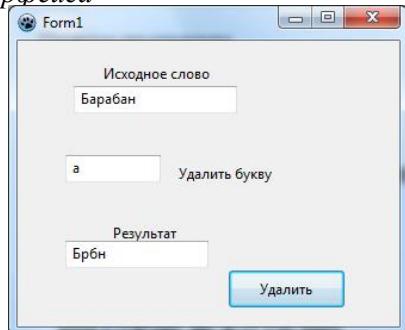
Пример интерфейса



Пример программы

```
uses
// ...
LCLProc;
//...
procedure TForm1.Button1Click(Sender: TObject);
var
  s : string;
  c1, c2 : string;
  i, j, len, len2 : Byte;
  symmetric : Boolean;
begin
  symmetric := True;
  s := Edit1.Text;
  len := UTF8Length(s);
  len2 := len div 2; // Половина длины слова.
  j := len;
  for i := 1 to len2 do begin
    c1 := UTF8Copy(s, i, 1); // С начала слова.
    c2 := UTF8Copy(s, j, 1); // С конца слова.
    if c1 <> c2 then begin
      symmetric := False;
    end;
    Dec(j);
  end;
  RadioButton1.Checked := symmetric;
  RadioButton2.Checked := not(symmetric);
end;
```

Упражнение 3. Удалить из текста все буквы «а»
Пример интерфейса



Код программы

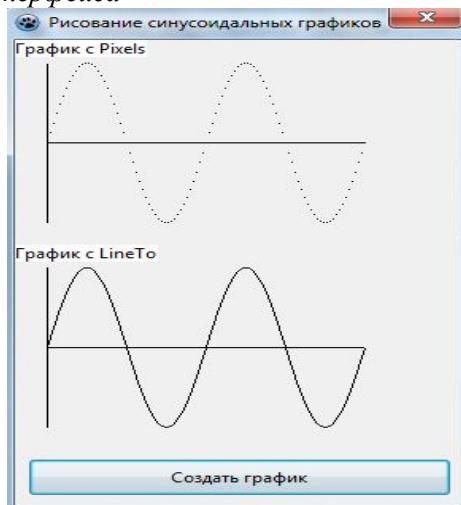
```

uses
// ...
LCLProc;
//...

procedure TForm1.Button1Click(Sender: TObject);
var
  c_del, s : string;
  i : Byte;
begin
  s := Edit1.Text; // Исходный текст.
  c_del := Edit3.Text; // Удаляемая буква.
  while UTF8Pos(c_del, s) <> 0 do begin
    // Пока в s осталась хотя бы одна искомая буква определяем её
    // номер и удаляем.
    i := UTF8Pos(c_del, s);
    UTF8Delete(s, i, 1);
  end;
  Edit2.Text := s;
end;
```

Упражнение 4. Построение графика функции

Пример интерфейса



Код программы

```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs,
StdCtrls;
type
  { TForm1 }
TForm1 = class(TForm)
  Button1: TButton;
  procedure Button1Click(Sender: TObject);
end;
var
  Form1: TForm1;
implementation
procedure TForm1.Button1Click(Sender: TObject);
var i, x01, y01, x02, y02, imax, dt1, dt2, func: integer;
begin
  imax := 100; // число точек в периоде
  dt1 := 2; // цена деления X
  dt2 := 2; // шаг во времени
  amp := 70; // амплитуда
  x01 := 20; // начала координат
  x02 := 20;
  y01 := 20 + amp;
  y02 := y01 + 2 * amp + 40;
  // Рисуем график с Pixels
  Canvas.TextOut(0, 0, 'График с Pixels');
  Canvas.MoveTo(x01, y01); // Рисуем ось X
  Canvas.LineTo(x01 + imax * dt1, y01);
  Canvas.MoveTo(x01, y01 + amp); // Рисуем ось Y
  Canvas.LineTo(x01, y01 - amp);
  for i := 0 to imax do // Рисуем график
    begin
      func := -round(amp * sin(2 * pi / imax * i * dt2));
      Canvas.Pixels[x01 + i * dt1, y01 + func] := clBlack;
    end;
end;
```

```

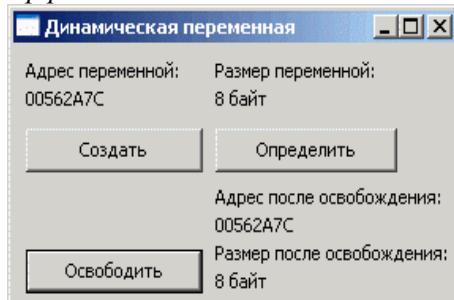
// Рисуем график с LineTo
Canvas.TextOut(0,y02-amp-20,'График с LineTo');
Canvas.MoveTo(x02,y02); //Рисуем ось X
Canvas.LineTo(x02+imax*dt1,y02);
Canvas.MoveTo(x02,y02+amp); //Рисуем ось Y
Canvas.LineTo(x02,y02-amp);
for i:=0 to imax do //Рисуем график
begin
func:=-round(amp*sin(2*pi/imax*i*dt2));
Canvas.LineTo(x02+i*dt1,y02+func);
end;
end;
initialization
{$R *.lfm}
end.

```

Упражнение 5. Динамическая переменная

В программе по нажатию кнопки должна создаваться динамическая переменная, определяться ее размер и адрес. Если переменная существует, то вывести сообщение об этом. Аналогичные действия должна выполнять программа при удалении.

Требования к выполнению. Использовать указатели и тип запись.
Пример интерфейса



Код программы

```

// Глобальные константы.
const
IS_EXIST = 'Переменная уже существует!';
NO_EXIST = 'Переменная НЕ существует!';

```

```
MY_DATA = 333;
INFO_SIZE = ' байт';
// Глобальные типы.
type
  p_node = ^t_node;
  t_node = record
    next : p_node;
    data : Integer;
  end;
// Глобальные переменные.
var
  p : p_node;
procedure TForm1.Button1Click(Sender: TObject);
begin
  if p = nil then begin
    New(p);
    p^.next := nil;
    p^.data := MY_DATA;
    Label2.Caption := IntToHex(Integer(Addr(p)), 8);
  end
  else ShowMessage(IS_EXIST);
end;

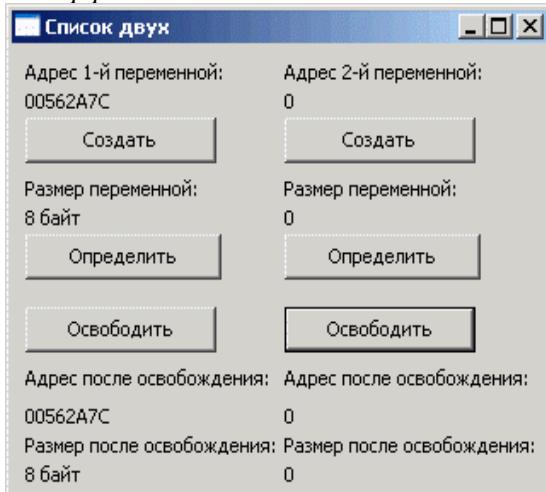
procedure TForm1.Button2Click(Sender: TObject);
begin
  if p <> nil then begin
    Dispose(p);
    p := nil;
    Label6.Caption := IntToHex(Integer(Addr(p)), 8);
    Label8.Caption := IntToStr(SizeOf(p^)) + INFO_SIZE;
  end
  else ShowMessage(NO_EXIST);
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  if p <> nil then begin
    Label4.Caption := IntToStr(SizeOf(p^)) + INFO_SIZE;
  end
  else ShowMessage(NO_EXIST);
end;
```

Упражнение 6. Создать две связанные динамические переменные
Аналогично предыдущей задаче создать две связанные динамические переменные.

Требования к выполнению. Использовать указатели и тип записи. Для визуализации процесса добавления использовать компоненты TShape.

Пример интерфейса



Код программы

```
// Глобальные константы.  
const  
  IS_EXIST = 'Переменная уже существует!';  
  NO_EXIST = 'Переменная НЕ существует!';  
  MY_DATA = 333;  
  INFO_SIZE = ' байт';  
// Глобальные типы.  
type  
  p_node = ^t_node;  
  t_node = record  
    next : p_node;  
    data : Integer;  
  end;  
// Глобальные переменные.  
var  
  p : p_node;
```

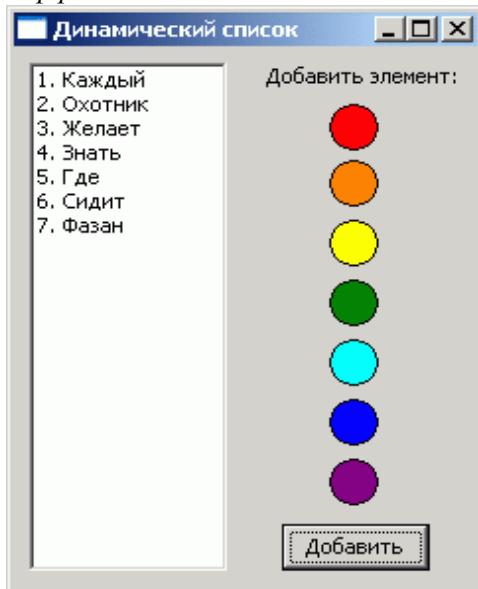
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if p = nil then begin
    New(p);
    p^.next := nil;
    p^.data := MY_DATA;
    Label2.Caption := IntToHex(Integer(Addr(p)), 8);
  end
  else ShowMessage(IS_EXIST);
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
  if p <> nil then begin
    Dispose(p);
    p := nil;
    Label6.Caption := IntToHex(Integer(Addr(p)), 8);
    Label8.Caption := IntToStr(SizeOf(p^)) + INFOSIZE;
  end
  else ShowMessage(NO_EXIST);
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
  if p <> nil then begin
    Label4.Caption := IntToStr(SizeOf(p^)) + INFOSIZE;
  end
  else ShowMessage(NO_EXIST);
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
  if p = nil then begin
    New(p);
    p^.next := nil;
    p^.data := MY_DATA;
    Label2.Caption := IntToHex(Integer(Addr(p)), 8);
  end
  else ShowMessage(IS_EXIST);
end;
```

Упражнение 7. Динамический список

По нажатию кнопки в динамический список должен добавляться новый элемент.

Требования к выполнению. Для визуализации процессов извлечения и добавления использовать компоненты *TShape*.

Пример интерфейса



Код программы

```
// Глобальные константы.  
type  
  p_my_list = ^t_my_list;  
  t_my_list = record  
    next : p_my_list;  
    s : string;  
  end;  
// Глобальные переменные.  
var  
  p_head : p_my_list;  
  p_current : p_my_list;  
  p : p_my_list;  
// Глобальные константы.
```

```

const
  N_MAX = 7;
  RANGE : array [1..N_MAX] of string = ('Каждый',
  'Охотник',
  'Желает',
  'Знать',
  'Где',
  'Сидит',
  'Фазан');
// Глобальные переменные.
var
  n : 1..N_MAX;
procedure TForm1.FormCreate(Sender: TObject);
begin
  n := 0;
  p_head := nil;
  p_current := nil;
  Shape1.Visible := False;
  Shape2.Visible := False;
  Shape3.Visible := False;
  Shape4.Visible := False;
  Shape5.Visible := False;
  Shape6.Visible := False;
  Shape7.Visible := False;
end;
procedure TForm1.FormDestroy(Sender: TObject);
var
  p : p_my_list;
  prev : p_my_list;
  find : Boolean;
begin
  if p_head <> nil then
    p := p_head;
    find := false;
  end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  if n < N_MAX then Inc(n);

```

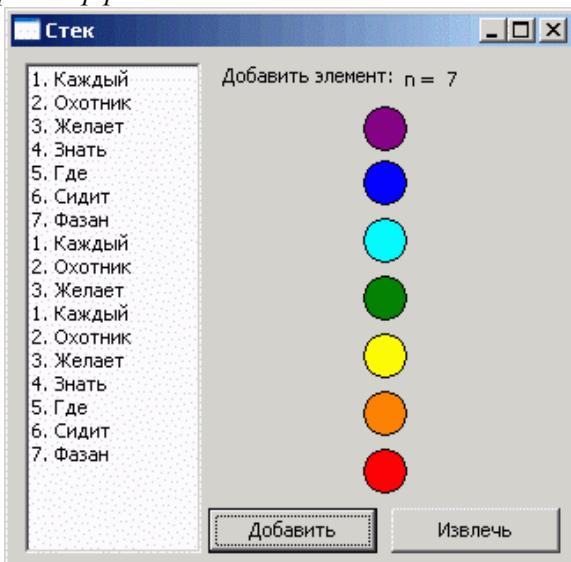
```
New(p);
p^.s := RANGE[n];
ListBox1.Items.Add(IntToStr(n) + ' ' + p^.s);
if p_head = nil then begin
  p^.next := nil;
  p_head := p;
end
else begin
  p^.next := p_current^.next;
  p_current^.next := p;
end;
p_current := p;
case n of
  1 : begin
    Shape1.Visible := True;
  end;
  2 : begin
    Shape2.Visible := True;
  end;
  3 : begin
    Shape3.Visible := True;
  end;
  4 : begin
    Shape4.Visible := True;
  end;
  5 : begin
    Shape5.Visible := True;
  end;
  6 : begin
    Shape6.Visible := True;
  end;
  7 : begin
    Shape7.Visible := True;
  end;
end;
```

Упражнение 8. Стек

В программе по нажатию кнопки должен добавляться элемент в стек, а по нажатию другой кнопки – извлекаться из стека.

Требования к выполнению. Для визуализации процессов извлечения и добавления использовать компоненты *TShape*.

Пример интерфейса



Код программы

```
// Глобальные константы.  
const  
NO_EXIST = 'Элемент не существует!';  
DEEP_STACK = 'Глубина стека ограничена: ';  
N_MAX = 7;  
RANGE : array [1..N_MAX] of string = ('Каждый',  
'Охотник',  
'Желает',  
'Знать',  
'Где',  
'Сидит',  
'Фазан');  
// Глобальные типы.  
type
```

```

p_my_stack = ^t_my_stack;
t_my_stack = record
  next : p_my_stack;
  s : string;
end;

// Глобальные переменные.
var
  p_head : p_my_stack;
  p : p_my_stack;
  n : 1..N_MAX;

procedure TForm1.FormCreate(Sender: TObject);
begin
  n := 0;
  p_head := nil;
  Shape1.Visible := False;
  Shape2.Visible := False;
  Shape3.Visible := False;
  Shape4.Visible := False;
  Shape5.Visible := False;
  Shape6.Visible := False;
  Shape7.Visible := False;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  p : p_my_stack;
begin
  if p_head <> nil then
    p := p_head;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if n < N_MAX then begin
    Inc(n);
    Label3.Caption := IntToStr(n);
    New(p);
    p^.next := p_head; // новый указывает на "бывшую" верхушку стека
  end;
end;

```

```
p^.s := RANGE[n];
p_head := p; // вершиной стека становится новый элемент
ListBox1.Items.Add(IntToStr(n) + '.' + p^.s);
case n of
  1 : begin
    Shape1.Visible := True;
  end;
  2 : begin
    Shape2.Visible := True;
  end;
  3 : begin
    Shape3.Visible := True;
  end;
  4 : begin
    Shape4.Visible := True;
  end;
  5 : begin
    Shape5.Visible := True;
  end;
  6 : begin
    Shape6.Visible := True;
  end;
  7 : begin
    Shape7.Visible := True;
  end;
end;
else ShowMessage(DEEP_STACK + IntToStr(N_MAX));
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  case n of
    1 : begin
      Shape1.Visible := False;
    end;
    2 : begin
      Shape2.Visible := False;
    end;
    3 : begin
```

```

Shape3.Visible := False;
end;
4 : begin
Shape4.Visible := False;
end;
5 : begin
Shape5.Visible := False;
end;
6 : begin
Shape6.Visible := False;
end;
7 : begin
Shape7.Visible := False;
end;
end;
if p_head <> nil then begin
Dec(n);
p := p_head;
p_head := p_head^.next;
end
else ShowMessage(NO_EXIST);
Label3.Caption := IntToStr(n);
end;

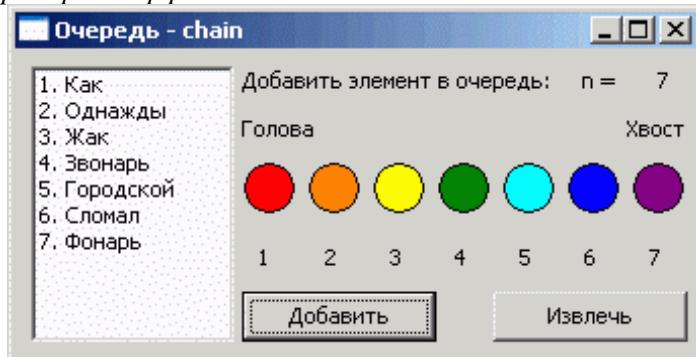
```

Упражнение 9. Очередь

В программе по нажатию кнопки должен добавляться элемент в очередь, а по нажатию другой кнопки – извлекаться из очереди.

Требования к выполнению. Для визуализации процессов извлечения и добавления использовать компоненты TShape.

Пример интерфейса



Код программы

```
// Глобальные константы.  
const  
  NO_EXIST = 'Элемент не существует!';  
  LEN_CHAIN = 'Длина очереди ограничена: ';  
  N_MAX = 7;  
  RANGE : array [1..N_MAX] of string = ('Как',  
  'Однажды',  
  'Как',  
  'Звонарь',  
  'Городской',  
  'Сломал',  
  'Фонарь');  
  
// Глобальные типы.  
type  
  p_my_chain = ^t_my_chain;  
  t_my_chain = record  
    next : p_my_chain;  
    s : string;  
  end;  
  
// Глобальные переменные.  
var  
  p_head : p_my_chain;  
  p : p_my_chain;  
  p_last : p_my_chain;  
  n : 0..N_MAX;  
  
procedure TForm1.FormCreate(Sender: TObject);  
begin  
  n := 0;  
  p_head := nil;  
  p := nil;  
  p_last := nil;  
  Shape1.Visible := False;  
  Shape2.Visible := False;  
  Shape3.Visible := False;  
  Shape4.Visible := False;  
  Shape5.Visible := False;
```

```

Shape6.Visible := False;
Shape7.Visible := False;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  p : p_my_chain;
  prev : p_my_chain;
begin
  if p_head <> nil then
    p := p_head;
  end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  if n < N_MAX then begin
    Inc(n);
    Label3.Caption := IntToStr(n);
    if p_last = nil then begin // если очередь пуста
      New(p);
      p_last := p;
      p_head := p_last;
      p^.next := nil; // за новым очередником никто не стоит
      // - Кто крайний? - Р - крайний!
    end
    else begin
      New(p);
      p^.next := nil; // за новым крайним никого нет
      p_last^.next := p; // "бывший" крайний указывает на нового крайне-
      го
      p_last := p; // указатель на крайнего смещается к новому
    end;
    p^.s := RANGE[n];
    ListBox1.Items.Add(IntToStr(n) + ' ' + p^.s);
    case n of
      1 : begin
        Shape1.Visible := True;
      end;
      2 : begin
        Shape2.Visible := True;
      end;
    end;
  end;
end;

```

```
3 : begin
Shape3.Visible := True;
end;
4 : begin
Shape4.Visible := True;
end;
5 : begin
Shape5.Visible := True;
end;
6 : begin
Shape6.Visible := True;
end;
7 : begin
Shape7.Visible := True;
end;
end;
else ShowMessage(LEN_CHAIN + IntToStr(N_MAX));
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
if n > 0 then Dec(n);
case (N_MAX - n) of
1 : begin
Shape1.Visible := False;
end;
2 : begin
Shape2.Visible := False;
end;
3 : begin
Shape3.Visible := False;
end;
4 : begin
Shape4.Visible := False;
end;
5 : begin
Shape5.Visible := False;
end;
6 : begin
Shape6.Visible := False;
```

```

end;
7 : begin
Shape7.Visible := False;
end;
end;
if p_head <> nil then begin
// Dec(n);
p := p_head;
p_head := p_head^.next;
Dispose(p);
end
else ShowMessage(NO_EXIST);
Label3.Caption := IntToStr(n);
end;

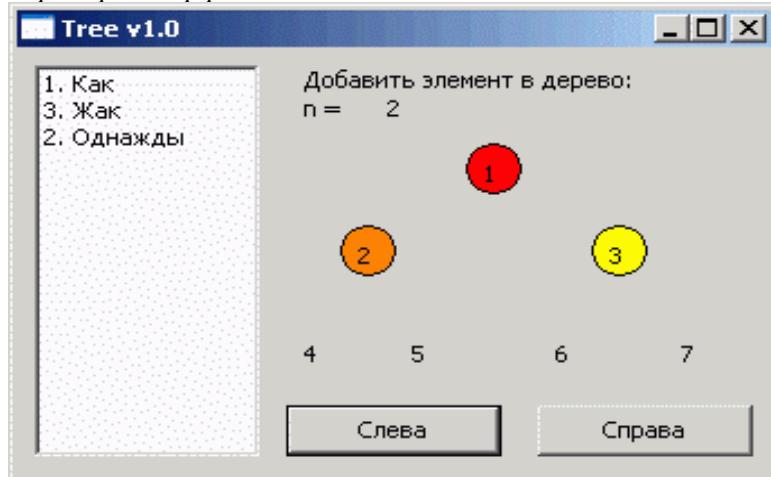
```

Упражнение 10. Дерево

Создать программу визуализирующую добавление элементов в динамическую структуру данных – дерево – с помощью компонента TShape. Доработать пример программы до второго уровня дерева с учетом того, что корень, root – нулевой уровень.

Требования к выполнению. Для визуализации построения дерева использовать компоненты TShape.

Пример интерфейса



Код программы

```
// Глобальные типы.
type
  p_my_tree = ^t_my_tree;
  t_my_tree = record
    left : p_my_tree;
    right : p_my_tree;
    s : string;
  end;

// Глобальные константы.
const
  NO_EXIST = 'Элемент не существует!';
  NUM_NODES = 'Узлы дерева ограничены: ';
  N_MAX = 7;
  RANGE : array [1..N_MAX] of string = ('Как',
  'Однажды',
  'Как',
  'Звонарь',
  'Городской',
  'Сломал',
  'Фонарь');

// Глобальные переменные.
var
  p_left : p_my_tree;
  p_root : p_my_tree;
  p_right : p_my_tree;
  p : p_my_tree;
  n : 0..N_MAX;
implementation
{ TForm1 }

procedure TForm1.FormCreate(Sender: TObject);
begin
  n := 0;
  p_left := nil;
  p_root := nil;
  p_right := nil;
```

```

Shape1.Visible := False;
Shape2.Visible := False;
Shape3.Visible := False;
Shape4.Visible := False;
Shape5.Visible := False;
Shape6.Visible := False;
Shape7.Visible := False;
end;

procedure TForm1.FormDestroy(Sender: TObject);
var
  p : p_my_tree;
begin
  if p_root <> nil then
    p := p_root;
  end;
// Добавить слева.
procedure TForm1.Button1Click(Sender: TObject);
begin
  if n < N_MAX then begin
    Inc(n, 2);
    if p_root = nil then begin
      New(p);
      p_root := p;
      p^.left := nil;
      p^.right := nil;
      n := 1;
      p^.s := RANGE[n];
      Shape1.Visible := True;
    end
    else begin
      New(p);
      if p_root^.left = nil then n := 2;
      p_root^.left := p;
      p^.left := nil;
      p^.right := nil;
      p^.s := RANGE[n];
    end;
  end;
end;

```

```

case n of
1 : begin
Shape1.Visible := True;
end;
2 : begin
Shape2.Visible := True;
end;
4 : begin
Shape4.Visible := True;
end;
6 : begin
Shape6.Visible := True;
end;
end;
ListBox1.Items.Add(IntToStr(n) + '.' + p^.s);
end
else ShowMessage(NUM_NODES + IntToStr(N_MAX));
end;

// Добавить справа.
procedure TForm1.Button2Click(Sender: TObject);
begin
if n < N_MAX then begin
Inc(n, 2);
if p_root = nil then begin
New(p);
p_root := p;
p^.left := nil;
p^.right := nil;
n := 1;
p^.s := RANGE[n];
Shape1.Visible := True;
end
else begin
New(p);
if p_root^.right = nil then n := 3;
p_root^.right := p;
p^.left := nil;

```

```

p^.right := nil;
p^.s := RANGE[n];
p^.s := RANGE[n];
end;
case n of
1 : begin
Shape1.Visible := True;
end;
3 : begin
Shape3.Visible := True;
end;
5 : begin
Shape5.Visible := True;
end;
7 : begin
Shape7.Visible := True;
end;
end;
end;
ListBox1.Items.Add(IntToStr(n) + '. ' + p^.s);
end
else ShowMessage(NUM_NODES + IntToStr(N_MAX));
end;

```

Упражнение 11. Олимпийский флаг

Создать программу, которая на поверхности формы рисует олимпийский флаг.

Пример интерфейса



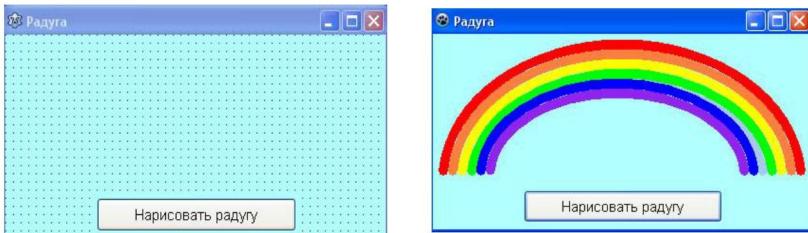
Код программы

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs;
TForm1 = class(TForm)
procedure FormPaint(Sender: TObject); private
{ Private declarations } public
{ Public declarations } end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
procedure TForm1.FormPaint(Sender: TObject); begin
  with Canvas do begin
    // полотнище Canvas.Pen.Width := 1; Canvas.Pen.Color := clBlack; Canvas.Brush.Color := clCream; Rectangle(30,30,150,115);
    // кольца
    Pen.Width := 2;
    Brush.Style := bsClear; // область внутри круга
    // не закрашивать Pen.Color := clBlue; Ellipse(40,40,80,80);
    Pen.Color := clBlack; Ellipse(70,40,110,80); Pen.Color := clRed; Ellipse(100,40,140,80); Pen.Color := clYellow; Ellipse(55,65,95,105);
    Pen.Color := clGreen; Ellipse(85,65,125,105); end; end;
end.
```

Упражнение 12. Радуга

Создать программу, которая на поверхности формы рисует радугу.

Пример интерфейса



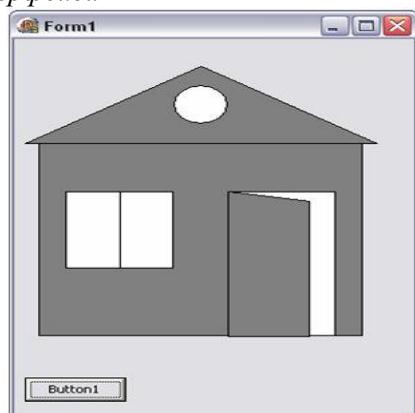
Код программы

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form1.Canvas.Pen.Width:=10; // Толщина пера
  Form1.Canvas.Pen.Color:=ClRed; // Цвет пера
  Form1.Canvas.Arc(10,10,390,290,390,140,10,140);
  Form1.Canvas.Pen.Color:=$004080FF; // Код цвета $00BBGGRR
  Form1.Canvas.Arc(20,20,380,280,380,140,20,140);
  Form1.Canvas.Pen.Color:=ClYellow;
  Form1.Canvas.Arc(30,30,370,270,370,140,30,140);
  Form1.Canvas.Pen.Color:=ClLime;
  Form1.Canvas.Arc(40,40,360,260,360,140,40,140);
  Form1.Canvas.Pen.Color:=ClSkyBlue;
  Form1.Canvas.Arc(50,50,350,250,350,140,50,140);
  Form1.Canvas.Pen.Color:=ClBlue;
  Form1.Canvas.Arc(50,50,340,240,340,140,60,140);
  Form1.Canvas.Pen.Color:=$00EE2590;
  Form1.Canvas.Arc(60,60,330,230,330,140,70,140);
end;
```

Упражнение 13. Дом

Создать программу, которая на поверхности формы рисует дом.

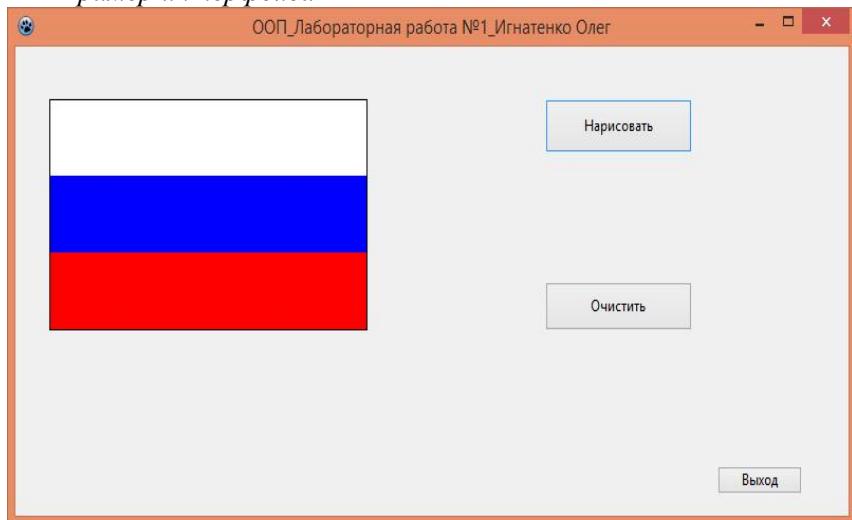
Пример интерфейса



Код программы

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Paintbox1.Canvas do begin
    Brush.Color := clGray;
    Rectangle(10,100,250,300); // корпус
    Polygon([Point(0,100),Point(130,20),Point(260,100)]); // крыша
    Brush.Color := clWhite;
    Ellipse(110,40,150,80); // чердак
    Rectangle(30,150,110,230); // окно
    MoveTo(70,150);
    LineTo(70,230);
    Rectangle(150,300,230,150); // дверь
    Brush.Color := clGray;
    Poly-
gon([Point(150,300),Point(150,150),Point(210,160),Point(210,300)]);
  end;
end;
```

Упражнение 14. Нарисовать в объекте TImage флаг России
Пример интерфейса



Код программы

```
unit Unit1;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
ExtCtrls,
  StdCtrls;
type
  { TForm1 }

TForm1 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  Button3: TButton;
  Image1: TImage;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure Label1Click(Sender: TObject);
  procedure Label4Click(Sender: TObject);
private
  { private declarations }
public
  { public declarations }
end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

{ TForm1 }

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

```
begin
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Color:=clBlack;
  Image1.Canvas.Brush.Color:=clBlack;
  Image1.Canvas.Rectangle(0,0,180,300);
  Image1.Canvas.Brush.Color:= clWhite;
  Image1.Canvas.Pen.Color:=clWhite;
  Image1.Canvas.Rectangle(1,1,300,60);
  Image1.Canvas.Brush.Color:= clBlue;
  Image1.Canvas.Pen.Color:= clBlue;
  Image1.Canvas.Rectangle (1,60,300,120);
  Image1.Canvas.Brush.Color:= clRed;
  Image1.Canvas.Pen.Color:= clRed;
  Image1.Canvas.Rectangle (1,120,300,180);

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Image1. Picture:= nil;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  close;
end;

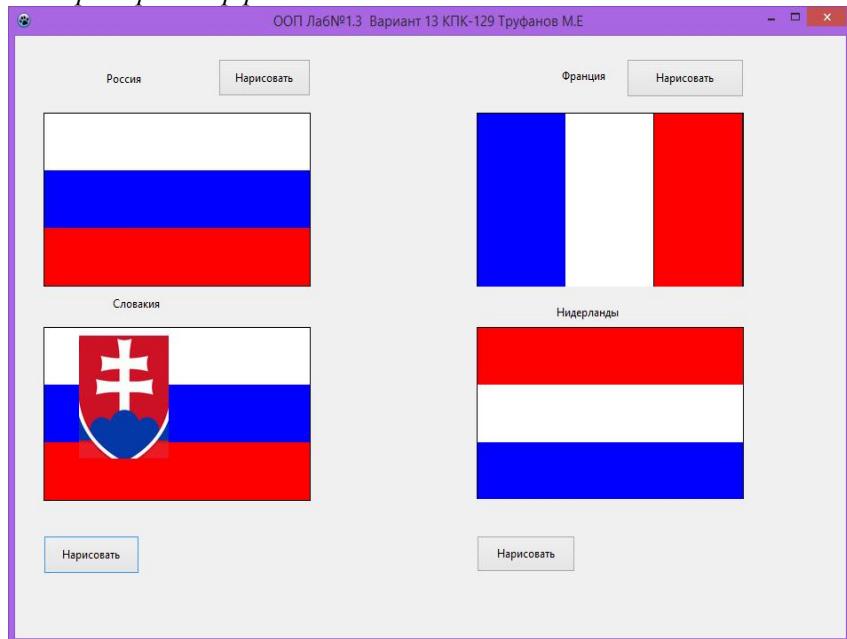
procedure TForm1.Label1Click(Sender: TObject);
begin
end;

procedure TForm1.Label4Click(Sender: TObject);
begin
end;

end.
```

Упражнение 15. Нарисовать в объекте TImage флаги стран мира

Пример интерфейса



Код программы

```
unit Unit1;
{$mode objfpc} {$H+}
interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
ExtCtrls,
  StdCtrls;

type
  { TForm1 }
TForm1 = class(TForm)
  Button1: TButton;
  Button2: TButton;
```

```
Button3: TButton;
Button4: TButton;
Image1: TImage;
Image2: TImage;
Image3: TImage;
Image4: TImage;
Image5: TImage;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Image1Click(Sender: TObject);
procedure Image2Click(Sender: TObject);
procedure Image4Click(Sender: TObject);
procedure Image5Click(Sender: TObject);
private
  { private declarations }
public
  { public declarations }
end;

var
  Form1: TForm1;
implementation
{$R *.lfm}
{ TForm1 }

procedure TForm1.Image1Click(Sender: TObject);
begin
end;

procedure TForm1.Image2Click(Sender: TObject);
begin
end;
procedure TForm1.Image4Click(Sender: TObject);
```

```
begin
end;

procedure TForm1.Image5Click(Sender: TObject);
begin
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Image1.Canvas.Pen.Color:=clBlack;
  Image1.Canvas.Brush.Color:=clBlack;
  Image1.Canvas.Rectangle(0,0,180,300);
  Image1.Canvas.Brush.Color:= clWhite;
  Image1.Canvas.Pen.Color:=clWhite;
  Image1.Canvas.Rectangle(1,1,300,60);
  Image1.Canvas.Brush.Color:= clBlue;
  Image1.Canvas.Pen.Color:= clBlue;
  Image1.Canvas.Rectangle (1,60,300,120);
  Image1.Canvas.Brush.Color:= clRed;
  Image1.Canvas.Pen.Color:= clRed;
  Image1.Canvas.Rectangle (1,120,300,180);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Image2.Canvas.Pen.Color:=clBlack;
  Image2.Canvas.Brush.Color:=clBlack;
  Image2.Canvas.Rectangle(0,0,180,300);
  Image2.Canvas.Brush.Color:= clWhite;
  Image2.Canvas.Pen.Color:=clWhite;
  Image2.Canvas.Rectangle(1,1,300,60);
  Image2.Canvas.Brush.Color:= clBlue;
  Image2.Canvas.Pen.Color:= clBlue;
  Image2.Canvas.Rectangle (1,60,300,120);
  Image2.Canvas.Brush.Color:= clRed;
  Image2.Canvas.Pen.Color:= clRed;
  Image2.Canvas.Rectangle (1,120,300,180);
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  Image5.Canvas.Pen.Color:=clBlack;
  Image5.Canvas.Brush.Color:=clBlack;
  Image5.Canvas.Rectangle(0,0,180,300);
  Image5.Canvas.Brush.Color:= clBlue;
  Image5.Canvas.Pen.Color:=clBlue;
  Image5.Canvas.Rectangle(1,1,100,299);
  Image5.Canvas.Brush.Color:= clWhite;
  Image5.Canvas.Pen.Color:= clWhite;
  Image5.Canvas.Rectangle (100,1,199,299);
  Image5.Canvas.Brush.Color:= clRed;
  Image5.Canvas.Pen.Color:= clRed;
  Image5.Canvas.Rectangle (200,1,299,299);
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  Image4.Canvas.Pen.Color:=clBlack;
  Image4.Canvas.Brush.Color:=clBlack;
  Image4.Canvas.Rectangle(0,0,180,300);
  Image4.Canvas.Brush.Color:= clRed;
  Image4.Canvas.Pen.Color:=clRed;
  Image4.Canvas.Rectangle(1,1,300,60);
  Image4.Canvas.Brush.Color:= clWhite;
  Image4.Canvas.Pen.Color:= clWhite;
  Image4.Canvas.Rectangle (1,60,300,120);
  Image4.Canvas.Brush.Color:= clBlue;
  Image4.Canvas.Pen.Color:= clBlue;
  Image4.Canvas.Rectangle (1,120,300,180);
end;
end.
```

Упражнение 16. Анимация (Сочи 2014)

Пример интерфейса



Код программы

```
unit Unit1;
{$mode objfpc} {$H+}
interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtCtrls,
  StdCtrls, Buttons;
const
  scr_width = 640; // ширина формы
  scr_height = 479; // высота формы
var
  x,y,y1:integer;
  type
```

```
{ TForm1 }

TForm1 = class(TForm)
Button1: TButton;
Image1: TImage;
Image2: TImage;
procedure Button1Click(Sender: TObject);
private
{ private declarations }
public
{ public declarations }
end;

var
Form1: TForm1;
implementation
{$R *.lfm}
{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
begin
X:=X+7;//ТЕКУЩАЯ КООРДИНАТА + ШАГ для фигуры
IF X>SCR_WIDTH+IMAGE2.WIDTH THEN X:=
IMAGE2.WIDTH;// ОГРАНИЧЕНИЕ СПРАВА
IF X<-IMAGE2.WIDTH THEN X:=SCR_WIDTH;// ОГРАНИЧЕНИЕ СЛЕВА
//РИСУЕМ
IMAGE2.LEFT:=X;
end;
end.
```

Упражнение 17. Анимация (Сочи 2014) + падающие медали

Пример интерфейса



Код программы

```
unit Unit3;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, Ex-
tCtrls,
  StdCtrls;
const
  scr_width = 640; // ширина формы
  scr_height = 479; // высота формы
var
  x,y,y1,i:integer;

type
  { TForm2 }
```

```

TForm2 = class(TForm)
  Button1: TButton;
  Image1: TImage;
  Image2: TImage;
  Image3: TImage;
  Image4: TImage;
  Image5: TImage;
  Timer1: TTimer;
  procedure Button1Click(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
private
  { private declarations }
public
  { public declarations }
end;

var
  Form2: TForm2;
implementation
{$R *.lfm}
{ TForm2 }

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  X:=X+7;//ТЕКУЩАЯ КООРДИНАТА + ШАГ для фигуры
  IF X>SCR_WIDTH+IMAGE2.WIDTH THEN X:=-IMAGE2.WIDTH;// ОГРАНИЧЕНИЕ СПРАВА
  IF X<-IMAGE2.WIDTH THEN X:=SCR_WIDTH;// ОГРАНИЧЕНИЕ СЛЕВА
  //РИСУЕМ
  IMAGE2.LEFT:=X;
  //Медаль 1
  y:=y+7;//ТЕКУЩАЯ КООРДИНАТА + ШАГ для фигуры
  IF y>SCR_WIDTH+IMAGE3.WIDTH THEN X:=-IMAGE3.WIDTH;// ОГРАНИЧЕНИЕ СПРАВА
  IF y<-IMAGE3.WIDTH THEN y:=SCR_WIDTH;// ОГРАНИЧЕНИЕ СЛЕВА

```

```

//РИСУЕМ
IMAGE3.Top:=y;
//Медаль 2
y:=y+7;//ТЕКУЩАЯ КООРДИНАТА + ШАГ для фигуры
IF y>SCR_WIDTH+IMAGE4.WIDTH THEN X:=-  

IMAGE4.WIDTH;// ОГРАНИЧЕНИЕ СПРАВА
IF y<-IMAGE4.WIDTH THEN y:=SCR_WIDTH;// ОГРАНИЧЕ-  

НИЕ СЛЕВА
//РИСУЕМ
IMAGE4.Top:=y;
//Медаль3
y:=y+7;//ТЕКУЩАЯ КООРДИНАТА + ШАГ для фигуры
IF y>SCR_WIDTH+IMAGE5.WIDTH THEN X:=-  

IMAGE5.WIDTH;// ОГРАНИЧЕНИЕ СПРАВА
IF y<-IMAGE5.WIDTH THEN y:=SCR_WIDTH;// ОГРАНИЧЕ-  

НИЕ СЛЕВА
//РИСУЕМ
IMAGE5.Top:=y;
end;

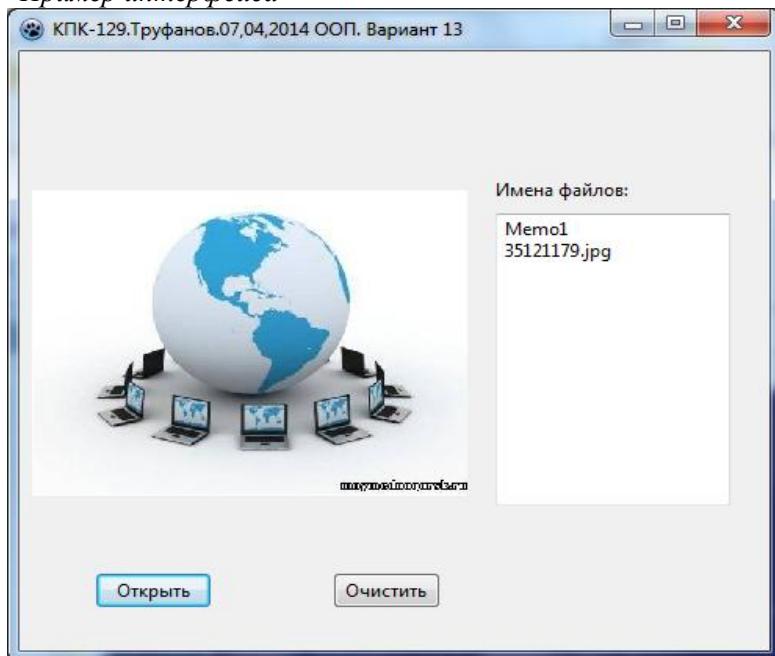
procedure TForm2.Button1Click(Sender: TObject);
begin
IF TIMER1.ENABLED=FALSE THEN
BEGIN
TIMER1.ENABLED:=TRUE;
BUTTON1.CAPTION:='СТОП';
I:=0;
END
ELSE
BEGIN
TIMER1.ENABLED:=FALSE;
BUTTON1.CAPTION:='ПУСК';
END;
END;
END.
end;

end.

```

Упражнение 18. Загрузить в компонент TImage один за другим несколько рисунков из файлов в формате JPEG с помощью диалога TOpenPictureDialog. Имя каждого файла записать в отдельную строку компонента TMemo

Пример интерфейса



Код программы

```
unit Unit1;
{$mode objfpc} {$H+}
interface
uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, ExtDlgs,
  ExtCtrls, StdCtrls;
type
  { TForm1 }
  TForm1 = class(TForm)
    Button1: TButton;
```

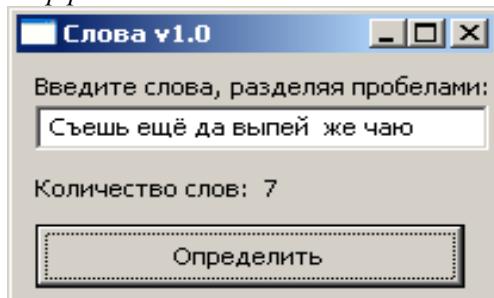
```
Button2: TButton;
Image1: TImage;
Label1: TLabel;
Memo1: TMemo;
OpenPictureDialog1: TOpenPictureDialog;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Image1Click(Sender: TObject);
private
  { private declarations }
public
  { public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.lfm}
{ TForm1 }

procedure TForm1.Image1Click(Sender: TObject);
begin
end;
procedure TForm1.Button1Click(Sender: TObject);
var
  s: string;
begin
  if OpenPictureDialog1.Execute then begin
    s:=OpenPictureDialog1.FileName;
    Image1.Picture.LoadFromFile(s);
    s:=ExtractFileName(s);
    Memo1.Append(s);
  end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Image1.Picture:=nil;
end;
end.
```

Упражнение 19. Определить сколько слов в тексте (слова отделены пробелами)

Пример интерфейса



Код программы

```
uses
// ...
LCLProc;
//...

procedure TForm1.Button1Click(Sender: TObject);
const
  SEPARATOR = ' ';
var
  s : string;
  c : string;
  i, len, count : Byte;
begin
  s := Edit1.Text;
  len := UTF8Length(s);
  count := 0;
  for i := 1 to len do begin
    c := UTF8Copy(s, i, 1);
    if c = SEPARATOR then Inc(count);
  end;
  if s <> " then Inc(count);
  Label3.Caption := IntToStr(count);
end;
```

Упражнение 20. Создать анимацию движущихся машины и мотоцикла

Пример интерфейса



Код программы

```
unit L16_Fedkov_unit1;

{$mode objfpc} {$H+}

interface
uses
  SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls;
const
  scr_width = 640; // ширина формы
  scr_height = 480; // высота формы

type
  { TForm1 }

TForm1 = class(TForm)
  Timer1: TTimer;
```

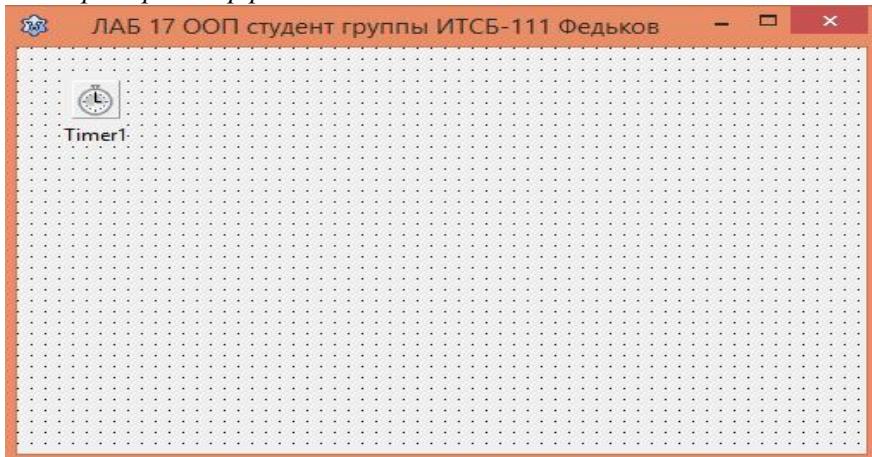
```
Image1: TImage;
Image2: TImage;
Image3: TImage;
procedure Image1Click(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form1: TForm1;
  x,y,x1,y1:integer;
implementation

{$R *.lfm}
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  x:=x+2;//текущая координата + шаг для автомобиля
  x1:=x1-2;//текущая координата + шаг для мотоцикла
  if x>scr_width+image2.Width then x:=-image2.Width;// ограничение справа
  if x1<-image3.Width then x1:=scr_width;// ограничение слева
  //рисуем
  image2.Left:=x;
  image3.Left:=x1;
end;

procedure TForm1.Image1Click(Sender: TObject);
begin
end;
```

Упражнение 21. Создать анимацию плывущего кораблика
Пример интерфейса



Код программы

```
unit L17_Fedkov_unit1;

{$mode objfpc} {$H+}
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Con-
trols, Forms,
  Dialogs, ExtCtrls;

type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```

var
  Form1: TForm1;
  x, y: integer; // координаты опорной точки
implementation

{$R *.lfm}
procedure Insert(x, y: integer);
begin
  // контук кораблика
  Form1.Canvas.MoveTo(x, y);
  Form1.Canvas.LineTo(x, y - 10);
  Form1.Canvas.LineTo(x + 50, y - 10);
  Form1.Canvas.LineTo(x + 55, y - 15);
  Form1.Canvas.LineTo(x + 85, y - 15);
  Form1.Canvas.LineTo(x + 70, y);
  Form1.Canvas.LineTo(x, y);
  // палубные надстройки
  Form1.Canvas.MoveTo(x + 15, y - 10);
  Form1.Canvas.LineTo(x + 20, y - 15);
  Form1.Canvas.LineTo(x + 20, y - 20);
  Form1.Canvas.LineTo(x + 65, y - 20);
  Form1.Canvas.LineTo(x + 65, y - 15);
  Form1.Canvas.MoveTo(x + 25, y - 15);
  Form1.Canvas.LineTo(x + 45, y - 15);
  Form1.Canvas.Rectangle(x + 40, y - 20, x + 55, y - 25);
  // труба
  Form1.Canvas.Rectangle(x + 35, y - 20, x + 40, y - 35);
  // иллюминаторы
  Form1.Canvas.Ellipse(x + 55, y - 10, x + 60, y - 5);
  Form1.Canvas.Ellipse(x + 65, y - 10, x + 70, y - 5);
  // мачта
  Form1.Canvas.MoveTo(x + 50, y - 25);
  Form1.Canvas.LineTo(x + 50, y - 60{50});
  // флагок на мачте
  Form1.Canvas.LineTo(x + 30, y - 60);
  Form1.Canvas.LineTo(x + 40, y - 58);
  Form1.Canvas.LineTo(x + 30, y - 55);
  Form1.Canvas.LineTo(x + 50, y - 55);

```

```

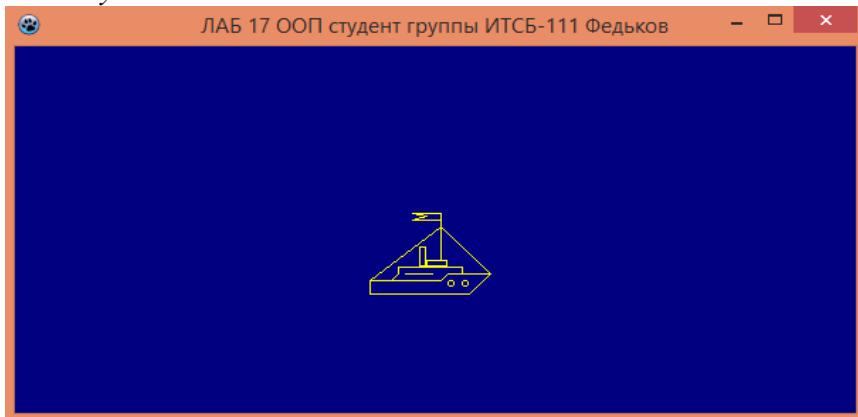
// тросы
Form1.Canvas.MoveTo(x + 85, y - 15);
Form1.Canvas.LineTo(x + 50, y - 50);
Form1.Canvas.LineTo(x, y - 10);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  // Стираем старый кораблик
  Form1.Canvas.Pen.Color := Form1.Color;
  Insert(x,y);

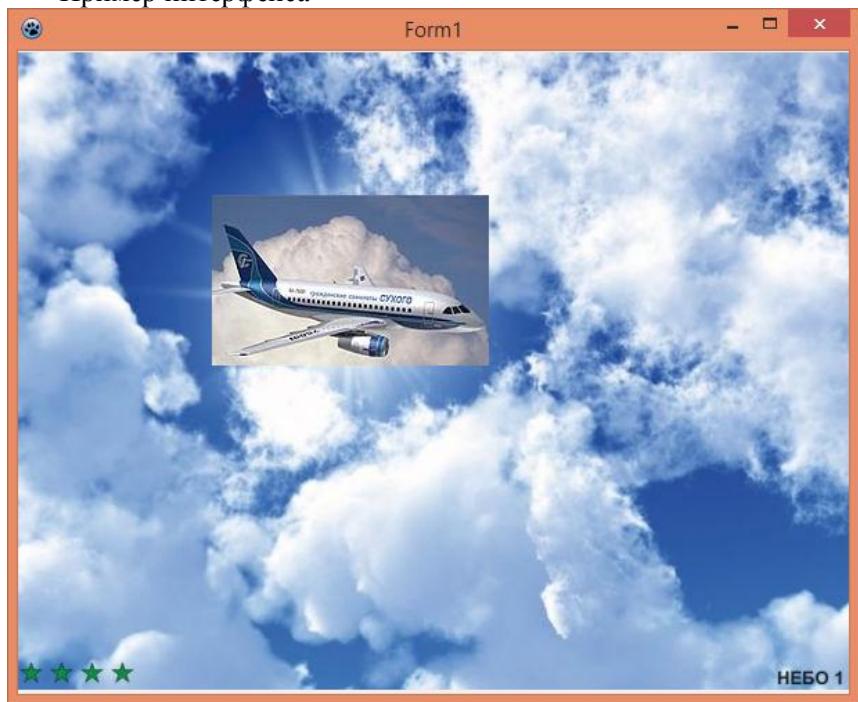
  // Смещаем координаты опорной точки
  if (Form1.ClientWidth > x) then
    Inc(x,5)
  else
    begin
      // Если вышли за границы экрана,
      // назначаем новые начальные координаты
      x := 0;
      y := Random(100) + 100;
    end;
  // Рисуем новый кораблик
  Form1.Canvas.Pen.Color := clYellow;
  Insert(x,y);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Form1.Width := 300;
  Form1.Height := 300;
  x := 0;
  y := Random(100) + 100;
  Form1.Color := clNavy;
  Timer1.Interval := 50;
end;
end.
```

Результат



Упражнение 22. Создать анимацию летящего самолета
Пример интерфейса



Код программы

```
unit Unit1;

{$mode objfpc} {$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
ExtCtrls;
const
  scr_width=969;// Ширина фона
  scr_height=506;// Высота фона
var
  x,y,y1:integer;
type
  { TForm1 }

TForm1 = class(TForm)
Image1: TImage;
Image2: TImage;
Timer1: TTimer;
procedure Timer1Timer(Sender: TObject);
private
  { private declarations }
public
  { public declarations }
end;

var
  Form1: TForm1;

implementation
{$R *.lfm}

{ TForm1 }
```

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  X:=X+70;//ТЕКУЩАЯ КООРДИНАТА + ШАГ ДЛЯ
  АВТОМОБИЛЯ
  IF X>SCR_WIDTH+IMAGE2.WIDTH THEN X:=-  

  IMAGE2.WIDTH;// ОГРАНИЧЕНИЕ СПРАВА
  IF X<-IMAGE2.WIDTH THEN X:=SCR_WIDTH;//  

  ОГРАНИЧЕНИЕ СЛЕВА
  //РИСУЕМ
  IMAGE2.LEFT:=X;
  END;
  END.

end;

end.
```

Приложение Б

Шаблон титульного листа к отчету по лабораторной работе

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

Кафедра инфокоммуникаций

ЛАБОРАТОРНАЯ РАБОТА № 2 по дисциплине «Объектно-ориентированное программирование»

Вариант № 7

на тему: «Программирование задач со строковыми данными»

Направление подготовки

210700.62 – Инфокоммуникационные технологии и системы связи

Обозначение лабораторной работы: **ЛР-СКФУ-210700.62-2014**

Группа: **ИТС-б-о-121**

Выполнил студент _____ **Иванов Иван Иванович**
дата, подпись

Проверил _____ **к. т. н., доц. Сорокин А. А.**
дата, подпись

Отчет защищен «__» ____ 2014 года

Оценка «_____»

Ставрополь, 2014

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ.....	3
ЛАБОРАТОРНЫЕ РАБОТЫ.....	5
1. Знакомство с интегрированной средой Lazarus	5
2. Реализация в IDE Lazarus простейших алгоритмов	18
3. Программирование задач линейной структуры	36
4. Программирование задач ветвящейся структуры	44
5. Программирование задач множественного выбора	56
6. Программирование задач циклической структуры	65
7. Программирование задач с данными типа вектор и матрица	83
8. Программирование задач со строковыми данными	99
9. Программы построения графиков.....	111
10. Создание графических примитивов в среде Lazarus	126
11. Создание приложений с мультиплексацией в среде Lazarus	133
12. Использование битовых образов в среде Lazarus.	
Мультиплексия	138
13. Страница Диалоги	141
14. Составные данные неоднородной структуры.	
Фиксированные записи.....	148
ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ	163
СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ.....	164
ПРИЛОЖЕНИЕ А. Примеры выполнения заданий (упражнения)	166
ПРИЛОЖЕНИЕ Б. Титульный лист отчета по лабораторной работе.....	214

Учебное издание

Сорокин Александр Анатольевич

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ
LAZARUS (FREE PASCAL)**

**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ
(ЛАБОРАТОРНЫЙ ПРАКТИКУМ)**



Издаётся в авторской редакции
Компьютерная верстка Н. П. Чивиджева

Подписано в печать 27.08.2014 г.

Формат 60 × 84 1/16

Усл. п. л. 12.96

Уч.-изд. л. 11.24

Бумага офсетная

Заказ 221

Тираж 15 экз.

Отпечатано в издательско-полиграфическом комплексе
ФГАОУ ВПО «Северо-Кавказский федеральный университет»
355029, Ставрополь, пр-т Кулакова, 2