ЗАКИРОВА ФЕРУЗА МАХМУДОВНА
НАБИУЛИНА ЛУИЗА МАХМУДОВНА

СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

ЗАКИРОВА ФЕРУЗА МАХМУДОВНА
НАБИУЛИНА ЛУИЗА МАХМУДОВНА

СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Данное учебное пособие предназначено для обучения студентов высших педагогических образовательных учреждений направления специальности 5110700 – Методика преподавания информатики по учебной дисциплине «Современные языки программирования».

НАЦИОНАЛЬНОЕ ОБЩЕСТВО
ФИЛОСОФОВ УЗБЕКИСТАНА
ТАШКЕНТ – 2012

UDK: 372.8(075)
BBK 32.973-018
З 23

Закирова, Феруза Махмудовна.

Современные языки программирования: учебное пособие / Ф. М. Закирова, Л. М. Набиулина; Мин-во высшего и среднего спец. образования РУз, Национальное общество философов Узбекистана. 2012. -192 с.

Набиулина, Луиза Махмудовна

UDK:372.8(075)
BBK 32.973-018
З 23

Данное учебное пособие предназначено для обучения студентов высших педагогических образовательных учреждений направления специальности 5110700 – Методика преподавания информатики по учебной дисциплине «Современные языки программирования». В нем представлены лабораторные работы по обучению основам программирования в среде Delphi.

Учебное пособие рекомендовано студентам педагогических университетов и институтов, а также учителям школ, академических лицеев и профессиональных колледжей, слушателям институтов и факультетов повышения квалификации.

Рецензенты:
Абдукадиров А. – доктор педагогических наук, профессор,
Мамаражабов М. – кандидат педагогических наук, доцент.

ISBN: 978-9943-391-44-4

ВВЕДЕНИЕ

С обретением независимости в Республике Узбекистан большое внимание уделяется модернизации и обновлению системы образования в соответствии с мировыми тенденциями. Президент И.А. Каримов отмечал: «Каждый из нас должен отдавать себе отчет в том, что Узбекистан сегодня – это составная часть мирового пространства...» [1, С.10]. Мировые тенденции развития образования направлены на всестороннее развитие человека путем вовлечения его в целесообразную самостоятельную учебно-познавательную деятельность.

Президент Республики Узбекистан И.А. Каримов в своих выступлениях неоднократно указывал на недостатки и проблемы в системе образования. Одной из проблем в обучении информатики является содержание и организация лабораторных занятий.

Лабораторные работы отнесены к одним из основных форм организации процесса обучения в высшей школе. Направленные на экспериментальное подтверждение теоретических положений и формирование учебных и профессиональных практических умений, они составляют важную часть профессиональной подготовки кадров.

Основная цель учебного пособия – дать представление о современном языке программирования Delphi, научить программировать в среде Delphi. Научиться программированию можно только программируя и решая конкретные задачи. Поэтому в данном учебном пособии представлены девять лабораторных работ по изучению основ программирования в среде Delphi. В них раскрываются:

– особенности программирования алгоритмов линейной структуры в среде Delphi;

– этапы решения задач программирования алгоритмов разветвляющейся структуры в среде Delphi на основе конструкций if ... then ... else и case;

– особенности программирования алгоритмов циклической структуры в среде Delphi, организация циклов с параметром, итерационных циклов, рекурсивных формул с использованием процедур и функций;

– специфика работы с одномерными массивами в среде Delphi, организация поиска и сортировок в линейных массивах;

– специфика работы с двумерными массивами в среде Delphi;

– принципы решения задач программирования с использованием символьных переменных в среде Delphi;

– особенности работы с файлами в среде Delphi;

– графические возможности среды Delphi;

– работа с мультимедиа в Delphi.

Выполнение этих лабораторных работ направлено на:

– обобщение, систематизацию, углубление, закрепление теоретических знаний;

– формирование умений применять теоретические знания на практике, реализацию единства интеллектуальной и практической деятельности;

– развитие интеллектуальных умений: аналитических, проектировочных, конструктивных и др.;

– выработку при решении поставленных задач таких профессионально значимых качеств специалиста, как самостоятельность, ответственность, точность, творческая инициатива.

Ведущей дидактической целью лабораторных работ по курсу «Современные языки программирования» является формирование практических умений – профессиональных (выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности) или учебных (решать задачи с использованием объектно-ориентированного программирования), необходимых в учебно-исследовательской деятельности.

При выборе содержания и объема лабораторных работ авторы исходили из сложности учебного материала, из внутрипредметных и межпредметных связей, из значимости изучаемых положений для предстоящей профессиональной деятельности, из того, какое место занимает конкретная лабораторная работа в совокупности лабораторных работ и их значимости для формирования целостного представления о содержании данной учебной дисциплины.

Все представленные лабораторные работы по курсу «Современные языки программирования» имеют следующую структуру:

1) тема и количество часов, отведенных на выполнение данной работы,

2) цель и результаты работы,

3) теоретическая часть с вопросами для самоконтроля,

4) основная часть с примерами решения задач в среде Delphi,

5) задания по вариантам трех уровней: первый – типовые задачи, второй – задачи частично-поискового характера, третий – задачи поискового характера для самостоятельного выполнения,

6) лист самооценки выполнения лабораторной работы.

Как уже было сказано, задания, предложенные в лабораторных работах, носят репродуктивный, частично-поисковый и поисковый характер. Задания первого – репродуктивного характера – отличаются тем, что при их выполнении студенты пользуются подробными инструкциями, в которых даны пояснения и порядок выполнения работы. Задания второго уровня, носящие частично-поисковый характер, отличаются тем, что при их проведении студенты не пользуются подробными инструкциями, им не дан порядок выполнения необходимых действий и от студентов требуют самостоятельного подбора выбора способов и действий выполнения работы. Задания третьего уровня, носящие поисковый характер, характеризуются тем, что студенты должны решить новую для них проблему, опираясь на имеющиеся у них теоретические знания, как из информатики, так и из смежных областей знаний.

При планировании лабораторных работ по курсу «Современные языки программирования» было найдено оптимальное соотношение репродуктивных, частично-поисковых и поисковых работ, чтобы обеспечить высокий уровень интеллектуальной деятельности студентов.

Лабораторная работа № 1

Программирование алгоритмов линейной структуры в Delphi

4 часа

Вы научитесь:

- запускать среду Delphi;
- визуально программировать, используя компоненты *Button, Edit,* Label, Form и устанавливая их свойства;
- создавать проект, сохранять и редактировать его;
- создавать код обработчика событий для решения задач линейной структуры в среде Delphi.

Теоретическая часть

Delphi – это среда разработки программ, ориентированных на работу в Windows. В основе идеологии Delphi лежат *методология объектно-ориентированного программирования и технология визуального проектирования*. Включать объекты в программу можно путем визуального программирования, используя заготовки – *компоненты*. Компоненты могут быть *визуальными*, видимыми при работе приложения, и *невизуальными*, выполняющими некоторые служебные функции; они отображаются в виде значка в процессе проектирования и не видны при работе приложения.

Программа, создаваемая в среде Delphi в процессе проектирования приложения, основана на *модульном принципе*. Головная программа состоит из объявления списка используемых модулей и нескольких операторов, создающих объекты для необходимых форм и запускающих приложение на выполнение. Все объекты компонентов размещаются в объектах – формах. Для каждой формы, проектируемой в приложении, Delphi автоматически создает отдельный модуль, в который пользователь может ввести собственный код, создавая обработчики различных событий. Именно в модулях и осуществляется программирование задачи.

После запуска Delphi на экране компьютера появляется основное окно интегрированной среды разработки.

В верхней части окна отображается полоса главного меню.

Ниже – две инструментальные панели:

• Левая панель содержит два ряда кнопок, дублирующих некоторые наиболее часто используемые команды меню (рис. 1.1).

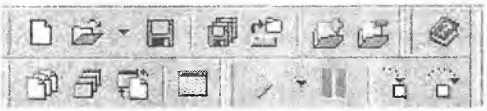


Рис. 1.1. Левая инструментальная панель среды Delphi.

• Правая панель содержит панель библиотеки визуальных компонентов (Visual Component Library – VCL), в дальнейшем просто *палитра компонентов* (рис. 1.2).

Рис. 1.2. Палитра компонентов среды Delphi.

Палитра компонентов позволяет выбрать с помощью иконок визуальные и другие компоненты, из которых, как из «строительных блоков», собирается разрабатываемое Delphi-приложение.

Правее полосы главного меню располагается небольшая инструментальная панель, которая служит для сохранения и выбора различных конфигураций окна (рис. 1.3).

Рис. 1.3. Инструментальная панель для сохранения и выбора конфигураций окна Delphi.

На основном окне интегрированной среды разработки расположены еще четыре окна:

Окно формы Form1 представляет собой заготовку (макет) окна разрабатываемого приложения.

Окно инспектора объектов Object Inspector позволяет изменять свойства (характеристики) объектов: формы командных кнопок, полей ввода и т.д.

Окно Object TreeView (дерево объектов) отображает иерархию компонентов приложения с точки зрения их принадлежности друг другу.

Окно Code Editor (редактор кода), в котором между `Begin` и `End` можно печатать инструкции Object Pascal, реализующие процедуру обработки событий (рис. 1.4).

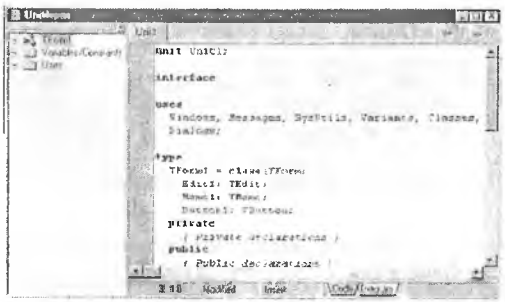


Рис. 1.4. Окно редактора кода Delphi.

Краткая характеристика основных компонентов Delphi

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

Основные свойства компонента Form:

Align	Задает режим выравнивания объектов внутри формы.
BorderStyle	Задает стиль обрамления формы, а также поведение формы (возможность менять размеры окна).
Caption	Задает заголовок окна формы.
Color	Задает цвет формы.
Font	Задает атрибуты шрифта формы.

Компонент **Label** (надпись или метка)  . Назначение – нести на себе надпись. Можно использовать для вывода ответа или пояснения вводимых данных. Относится к группе Standard.

Основные свойства компонента Label:

Caption	Задает заголовок надписи, выводимой на экран.
Alignment	Задает режим выравнивания текста метки.
AutoSize	Позволяет автоматически менять размеры метки, чтобы соответствовать размерам надписи (значение True).
Font	Задает шрифт, используемый для отображения текста.
Visible	Задает видимость надписи на экране. Имеет два значения. Если значение True, то надпись видна, False – нет.

Компонент **Edit** (поле редактирования)  . Используется для ввода/вывода чисел и текста в программу. Относится к группе Standard.

Основные свойства компонента Edit:

AutoSize	Задает необходимость изменения размера компонента при изменении размера шрифта (если True).
BorderStyle	Задает стиль обрамления поля.
Text	Задает содержимое строки редактирования.
MaxLength	Ограничивает число вводимых в поле символов.
ReadOnly	Запрещает редактировать отображаемый текст (если True).

Компонент **Button** (командная кнопка)  . Используется для задания реакции на событие. Относится к группе Standard.

Основные свойства Button:

Caption	Задает название кнопки.
Height	Задает высоту кнопки.
Width	Задает ширину кнопки.
Left	Задает расстояние от левой границы кнопки до левой границы формы

1. В чем заключается основное отличие процедурного и объектно-ориентированного программирования? Назовите основные принципы объектно-ориентированного программирования.
2. Что такое объект? Что такое класс?
3. В чем заключается наследование?
4. Что такое метод?
5. Что такое свойство объекта, каким образом его можно изменять?
6. Что такое события? Каково назначение обработчика событий?
7. В чем заключается преимущество визуального программирования интерфейса?
8. Опишите структуру и назначение отдельных элементов головной программы приложения Delphi.
9. Каково назначение модуля в проекте приложения Delphi? Опишите назначение отдельных разделов модуля.
10. Какие компоненты входят в интегрированную среду разработки приложений Delphi?

11. Перечислите основные компоненты окна среды Delphi и укажите их назначение.
12. Как разместить компонент на форме?
13. Какими способами можно изменять свойства компонента? Приведите примеры.
14. Перечислите состав проекта Delphi. Опишите назначение различных файлов.
15. Каково назначение обработчиков событий? Каким образом можно инициировать создание процедуры-обработчика событий?
16. Изучите и опишите информацию о компонентах палитры Standard по справке Delphi.
17. Что хранится в файле проекта с расширением `.Dfm`?
18. Что хранится в файле проекта с расширением `.Pas`?
19. Что хранится в файле проекта с расширением `.Dpr`?
20. Что хранится в файле проекта с расширением `.Res`?

Основная часть

Задание 1.

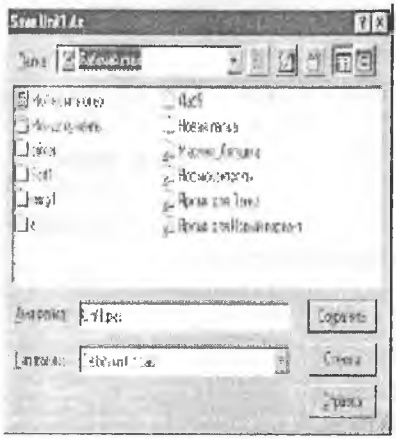
Создать приложение, которое обеспечивает ввод двух целых чисел, вычисляет их сумму и выводит значение результата.

Ход выполнения задания:

1. Запустите Delphi с помощью команды Главного меню Windows Пуск → Программы → Borland Delphi 7 → Delphi 7.

2. Создайте новый проект при помощи команды File → New → Application. В раскрытом окне формы можно размещать визуальные компоненты для реализации проекта приложения.

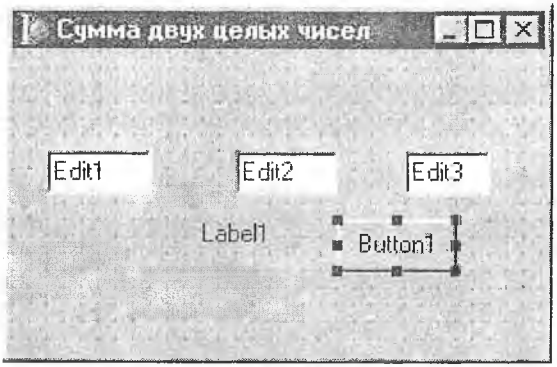
3. Сохраните новый проект командой меню File → Save Project As. В появившемся окне Save Unit1 As с помощью кнопки  создайте новую папку для файлов создаваемого проекта с названием «Сумма чисел» на Рабочем столе. Откройте созданную папку и нажмите кнопку Сохранить. После сохранения файла модуля `Unit1.pas` откроется окно Save Project As. Задайте имя файла проекта «Summa»и нажмите кнопку Сохранить.



4. Измените заголовок формы Form1. Для этого в окне Инспектора объектов откройте страницу Свойства (*Properties*), выберите свойство *Caption* (Заголовок) и задайте его новое значение «Сумма двух целых чисел».

5. Разместите на форме компоненты Edit1, Edit2, Edit3, Label1, Button1, используя соответствующие пиктограммы группы `Standard`.

Для этого щелкните на вкладке Standard палитры компонентов, затем выберите пиктограмму требуемой компоненты и щелкните в окне формы в том месте, где хотите ее расположить.



Примечание. Выделенный компонент формы окружен восемью маркерами. Именно его свойства отображаются в окне Инспектора объектов. Выделенный компонент можно перемещать по форме, меняя его местоположение. Можно также менять его размеры, потянув за один из маркеров.

6. Измените свойства компонента *Label1:* Задайте свойство *Caption* (заголовок) как «+», в списке свойств *Font* (Шрифт) свойству *Size* (Размер) присвойте значение 20.

7. Измените свойства компонента *Button1:* Задайте свойство *Caption* (заголовок) как «=», в списке свойств *Font* (Шрифт) свойству *Size* (Размер) присвойте значение 20.

8. Выровняйте компоненты на форме. Для этого выделите их все при нажатой клавише Shift, затем в контекстном меню выберите команду Position → Align.

9. Измените свойства компонентов *Edit1, Edit2, Edit3*: Удалите текст из свойства *Text*.

10. Добавьте на форму еще три объекта *Label*, расположите их над объектами *Edit1 – Edit3* и задайте их свойствам *Caption* значения «Слагаемое», «Слагаемое» и «Сумма».

11. Активизируйте окно Редактора кода, нажав F12. Просмотрите сгенерированный Delphi-модуль описания формы и размещенных на ней компонентов.

12. Сохраните изменения, внесенные в проект, командой меню File → Save All.

13. Откомпилируйте созданный проект командой меню Project → Compile summa (summa – это имя проекта).

14. С помощью файлового менеджера просмотрите папку проекта.

`summa.cfg` – файл конфигурации проекта

`summa.dof` – файл параметров проекта.

`summa.dpr` – файл проекта Delphi. Главная программа приложения.

`summa.exe` – откомпилированный проект. Исполняемая Windows-программа.

`summa.res` – файл ресурсов Windows.

`Unit1.~dfm, unit1.~pas` – файлы резервных копий.

`Unit1.dcu` – откомпилированный модуль.

`Unit1.dfm` – файл формы (двоичный файл) содержит начальные данные для компонент.

`Unit1.pas` – исходный код модуля формы.

15. Закройте окно файлового менеджера, активизируйте окно Delphi и запустите проект на выполнение командой меню Run → Run.

16. Рассмотрите окно созданной формы. Обратите внимание, что оно имеет стандартные атрибуты окна Windows.

17. Завершите работу приложения любым из стандартных методов.

18. Создайте код обработчика событий: при нажатии на кнопку «=» два введенных значения должны складываться.

Для этого: выделите кнопку *Button1*, в окне Инспектора объектов перейдите на вкладку *Events* (События), выберите событие *OnClick* и дважды щелкните левой кнопкой мыши на пустом поле списка. В открывшемся окне Редактора кода между операторами begin и end разместите необходимые операторы.

Окончательно процедура обработки события щелчка на кнопке Button1 должна выглядеть следующим образом:

```
Procedure TForm1.Button1Click(Sender: TObject);
var
a,b,c: integer {Слагаемые и сумма целые числа}
begin
a:=StrToInt(Edit1.Text);{Преобразование текстовой
строки в целое число}
b:=StrToInt(Edit2.Text);
c:=a+b;
Edit3.Text:=IntToStr(c);{Преобразование целого числа
в текстовую строку}
end;
```

19. Сохраните изменения в проекте.

20. Запустите приложение на выполнение.

21. После проверки работы приложения закройте его.

22. Запустите приложение из Windows, используя исполняемый файл.

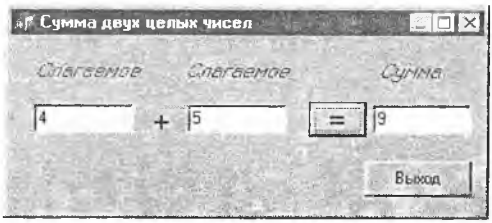
Задания для самостоятельного выполнения

Задание 2.

В созданном проекте измените свойства компонентов `Label2`, `Label3`, `Label4`, задав стиль шрифта – курсив, цвет шрифта – синий, размер символов – 10 пунктов.

Задание 3.

В созданный проект добавьте кнопку на закрытие приложения. В редакторе кода используйте оператор `Close`.



Задание 4.

Создайте новый проект, реализующий ввод двух целых чисел, их вещественное деление и вывод результата на экран. При выводе результата в текстовое окно используйте функцию преобразования вещественного числа в текстовую строку `FloatToStr`.

Задание 5.

Выполните индивидуальное задание первого уровня.

Создать приложение, вычисляющее значения переменных по заданным расчетным формулам и наборам исходных данных. На экран вывести значения вводимых исходных данных и результаты вычислений, сопровождая ввод и вывод поясняющими комментариями.

№	Расчетные формулы	Значения исходных данных
1.	$a = \frac{2\cos(x-\frac{\pi}{6})}{\frac{1}{2}+sin^2 y};$ $b = 1+\frac{z^2}{3+\frac{z^2}{5}}$	x= 1,426 y=-1,22 z=3,5
2.	$\gamma = \lvert x^{y/x} - \sqrt[3]{y/x}\rvert$ $\omega = (y-x)\frac{y-z/(y-x)}{1+(y-x)^2}$	x = 1,825 y = 18,225 z = –3,298
3.	$s = 1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!};$ $w = x(sinx^3+cos^2 y)$	x = 0,335 y = 0,025
4.	$y = e^{-bt}\sin(at+b) - \sqrt{\lvert bt+a\rvert}$ $s = b\,sin(at^2 cos2t) - 1$	a = – 0,5 b = 1,7 t = 0,44
5.	$w = \sqrt{x^2+b} - b^2 sin^3(x+a)/x$ $y = cos^2 x^3 - x/\sqrt{a^2+b^2}$	a = 1,5 b = 15,5 x = -2,9
6.	$s = x^3 tg^2(x+b)^2 + a/\sqrt{x+b}$ $Q = \frac{bx^2-a}{e^{ax}-1}$	a = 16,5 b = 3,4 x = 0,61
7.	$R = \frac{x^2(x+1)}{b} - sin^2(x+a);$ $s = \sqrt{\frac{xb}{a}} + cos^2(x+b)^2$	a = 0,7 b =0,05 x = 0,5
8.	$y = sin^3(x^2+a)^2 - \sqrt{\frac{x}{b}}$ $z = \frac{x^2}{a} + \cos(x+b)^3$	a = 1,1 b = 0,004 x = 0,2
9.	$f = \sqrt[3]{mctb + \lvert c\sin t\rvert}$ $z = m\cos(bt\sin t) + c$	m=2 c=-1 t=1,2 b=0,7
10.	$y = abx^2 - \frac{a}{sin^2\frac{x}{a}}$ $d = ae^{-\sqrt{a}}cos\frac{bx}{a}$	a = 3,2 b = 17,5 x = -4,8

11.	$f = \ln(a + x^2) + \sin^2(x/b)$ $z = e^{-a}\dfrac{x + \sqrt{x+a}}{x - \sqrt{\lvert x-b \rvert}}$	$a = 10.2$ $b = 9.2$ $c = 0.5$ $x = 2.2$
12.	$y = \dfrac{a^{2x} + b^{-x}\cos(a+b)x}{x+1}$ $R = \sqrt{x^2 + b} - b^2 \sin^3(x + a)/x$	$a = 0.3$ $b = 0.9$ $x = 0.61$
13.	$z = \sqrt{ax \sin 2x + e^{-2x}(x+b)}$ $\omega = \cos^2 x^3 - x/\sqrt{a^2 + b^2}$	$a = 0.5$ $b = 3.1$ $x = 1.4$
14.	$U = \dfrac{a^{bx} + e^{-bx}\cos bx}{bx - e^{-bx}\sin bx + 1}$ $f = e^{2x}\ln(a + x) - b^{3x}\ln(b - x)$	$a = 0.5$ $b = 2.9$ $x = 0.3$
15.	$z = \dfrac{\sin x}{\sqrt{m^2 + \sin^2 x}} - cm\ln mx$ $s = e^{-ax}\sqrt{x+1} + e^{-bx}\sqrt{x+1.5}$	$m = 0.7$ $c = 2.1$ $x = 1.7$ $a = 0.5$ $b = 1.08$

Задание 6.

Выполните индивидуальное задание второго уровня. Создать приложение для решения задачи.

Вариант задания	Формулировка задачи
1.	Вычислить площадь и периметр прямоугольника, если задана длина одной стороны (a) и коэффициент n (%), позволяющий вычислить длину второй стороны (b=n*a).
2.	Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
3.	Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b.
4.	Вычислить площади геометрических фигур: прямоугольника и треугольника по заданным сторонам.
5.	По известному радиусу вычислить объем и площадь поверхности шара.
6.	Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.

7.	Даны два числа. Вычислить их сумму, разность, произведение и частное.
8.	Известен объем информации в байтах. Выразить его в мегабайтах и гигабайтах.
9.	Длина выражена в сантиметрах. Выразить ее в дюймах. (1 дюйм = 2,5 см).
10.	Перевести значение веса, выраженное в граммах, в унции (1 унция = 28,3 г).
11.	Три сопротивления соединены последовательно. Найти сопротивление соединения.
12.	Вычислить путь, пройденный лодкой по течению, если известна ее скорость в стоячей воде, скорость течения реки и время движения.
13.	Вычислить расстояние между двумя точками с заданными координатами.
14.	Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
15.	Известны внутренний и внешний диаметры кольца. Найти его площадь.

Задание 7.

Выполните индивидуальное задание третьего уровня. Создать приложение для решения задачи.

Вариант задания	Формулировка задачи
1.	Вычислить объем призмы, боковые грани которой квадраты, а основанием служит равносторонний треугольник, вписанный в круг радиуса R.
2.	Треугольник задан тремя сторонами. Вычислить его медианы.
3.	В шар радиуса R вписан конус с углом α при вершине в осевом сечении конуса. Определить объем и полную поверхность конуса.
4.	Вычислить диагональ и площадь прямоугольника, вписанного в окружность радиуса R, если отношение его сторон равно n.

5.	Даны две стороны треугольника и угол между ними. Определить третью сторону, площадь треугольника и радиус описанной окружности.
6.	Определить плату за квартиру, если известно: площадь квартиры, количество проживающих, плата за содержание жилья, отопление, водопровод, канализацию, подогрев воды, коллективную антенну, лифт, вывоз мусора.
7.	Определить плату за электроэнергию, если известны: старое и новое показания счетчика, стоимость одного квт/часа электроэнергии, количество просроченных дней и размер пени за один день просрочки.
8.	V_1 литров воды нагревается на электроплите от температуры t_0 до температуры t_1, а V_2 литров – до температуры t_2. Сколько будет стоить затраченная на это электроэнергия? КПД электроплитки и стоимость 1 квт/час электроэнергии известны (удельная теплоемкость воды С = 4190 Дж/кгК).
9.	Пусть смешано V_1 литров воды температуры t_1 с V_2 литрами воды температуры t_2 и V_3 литрами воды температуры t_3. Вычислить объем и температуру образовавшейся смеси.
10.	Торговая фирма закупила n количество меховых изделий по цене C_1 для продажи. Фирма облагается налогами: 20% налог от прибыли в местный бюджет, 28% налог в Пенсионный фонд, 5% от объема продажи идет на формирование зарплаты. Определить розничную цену товара, если планируемая прибыль предприятия 15%.
11.	Длина отрезка задана в дюймах (1 дюйм = 2,54 см). Перевести значение длины в метрическую систему, то есть выразить ее в метрах, сантиметрах и миллиметрах. Например, 21 дюйм = 0 м 53 см 3,4 мм.
12.	Заданы моменты начала и конца некоторого промежутка времени в часах, минутах и секундах (в пределах одних суток). Найти продолжительность этого промежутка в тех же единицах.
13.	Текущее время (часы, минуты, секунды) задано тремя переменными: *h, m, s*. Округлить его до целых значений минут и часов. Например, 14 ч 21 мин. 45 с преобразуется в 14 ч 22 мин. или 14 ч, а 9 ч 59 мин. 23 с – соответственно в 9 ч 59 мин. или 10 ч.

14.	Угол α задан в радианах. Найти его величину в градусах, минутах и секундах.
15.	Длина некоторого отрезка составляет p метров. Перевести ее в русскую неметрическую систему: 1 верста = 500 саженей; 1 сажень = 3 аршина; 1 аршин = 16 вершков; 1 вершок = 44,45 мм.

Лист самооценки выполнения лабораторной работы № 1.

Каждый пункт оценивается в 1 балл, если с уверенностью отвечаете «да». Максимальное количество баллов – 5.

1) Самостоятельно ответили на все вопросы из раздела «Вопросы для самоконтроля».
2) Самостоятельно выполнили задания №1-4.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 2

Программирование алгоритмов разветвляющейся структуры в Delphi

4 часа

Вы научитесь:

- использовать компоненты `RadioGroup` и `CheckBox` и устанавливать их свойства;
- использовать операторы *if ... then ... else* и *case* при решении задач разветвляющейся структуры в среде Delphi.

Теоретическая часть

Операторы в программе-обработчике событий выполняются в той последовательности, в которой они записаны. Однако достаточно часто требуется изменить порядок выполнения операторов в зависимости от выполнения (или невыполнения) определенного условия. Существуют управляющие конструкции, предназначенные для управления порядком выполнения операторов. Основанием для принятия решений в управляющих операторах является истинность или ложность условного (логического) выражения.

Условные выражения – это такие выражения, которые возвращают одно из двух значений True (Истина) или False (Ложь). Простые логические выражения содержат операции отношения (операции сравнения): = (равно), > (больше), < (меньше), <> (не равно), >= (больше или равно), <= (меньше или равно). Сложные логические выражения строятся из простых логических выражений и логических операций, примененных к ним.

В приведенной таблице A и B – логические выражения:

№	Операция	Обозначение	Истолкование
1	Отрицание (инверсия)	not A	Не A; Неверно, что A
2	Конъюнкция (логическое произведение, логическое И)	A and B	A и B; A, но B; A, а B; как A, так и B; A вместе с B; A в то время, как B
3	Дизъюнкция (логическое сложение, логическое ИЛИ)	A or B	A или B; A или B или оба
4	Дизъюнкция (исключающее ИЛИ)	A xor B	A либо B; A или B, но не оба

Приоритеты выполнения логических операций в логических выражениях: отрицание (not), логическое произведение (and), логическое сложение (or), исключающее или (xor). Скобки меняют порядок выполнения операций.

Заполните таблицу истинности для основных логических операций

A	B	Not A	A and B	A or B	A xor B
False					
False					
True					
True					

Условные операторы предназначены для выбора на исполнение одного из возможных действий (операторов) в зависимости от некоторого условия, при этом одно из действий может отсутствовать.

Выбор действия в зависимости от выполнения условия может быть реализован при помощи оператора условия.

Вспомните, каким образом записывается оператор условия в языке программирования Паскаль.

Кроме того, можно использовать оператор выбора, который позволяет реализовать множественный выбор.

Вспомните, каким образом записывается оператор множественного выбора в языке программирования Паскаль.

Оператор Case существует в двух вариантах:

```
Case k of
A1:  <инструкция 1>;
A2:  <инструкция 2>;
.....
AN:  <инструкция N>;
End;
```

или:

```
Case k of
A1:  < инструкция 1>;
A2:  < инструкция 2>;
.....
AN:  < инструкция N>
Else
< инструкция, выполняемая в случае, если значение вы-
ражения не попало ни в один из списков констант A1,
A2, …,AN>
End;
```

Здесь

– k – выражение-селектор, от значения которого зависит дальнейший ход программы, может иметь только простой порядковый тип (целый, символьный, логический);

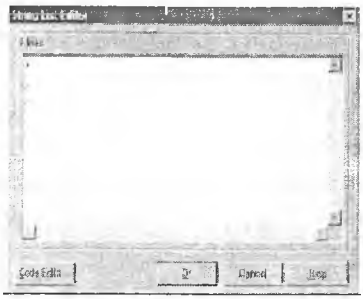
– Список констант (A1,…,AN) – константы, того же типа, что и селектор, выполняющие роль меток ветвей. Если константы представляют диапазон чисел, то вместо списка можно указать первую и последнюю константу диапазона, разделив их двумя точками.

Краткая характеристика основных компонентов Delphi

Компонент RadioGroup (группа переключателей)  . Позволяет отображать поля с ограниченным множеством значений. Относится к группе Standard.

Основные свойства компонента RadioGroup:

Caption	Задает название группы переключателей.
Items	Определяется количество переключателей в группе и надписи около них. Надписи задаются в окне String List Editor.

	
ItemsIndex	Задает номер кнопки, выбранной по умолчанию. 0 – первая, -1 – ни одна кнопка не выбрана.

Пример использования в программе:

{Выбор арифметического действия по номеру отмеченной кнопки}

```
Case RadioGroup1.ItemIndex of
  0: c:=a+b;
  1: c:=a-b;
  2: c:=a*b;
End;
```

Компонент CheckBox (кнопка с независимой фиксацией – флажок Windows  . Позволяет пользователю выбрать/отменить определенную опцию. Состояние кнопки содержится в свойстве Checked. Относится к группе Standard.

Основные свойства компонента CheckBox:

Caption	Задает текст, сопровождающий кнопку.
AllowGrayed	Задает наличие у кнопки третьего состояния. Если значение этого свойства False, то кнопка может находиться в двух состояниях – включенном или выключенном. Если значение свойства равно True, то добавляется третье состояние, когда кнопка неактивна.
Checked	Возвращает или задает, наличие галочки флажка. True – кнопка включена, False – выключена.

Пример использования в программе:

```
{проверка наличия галочки у флажка, если есть - вывод
форматный, нет - бесформатный}

if CheckBox1.Checked then
  Edit3.Text:=FloatToStrF(c,ffFixed,10,4)
else
    Edit3.Text:=FloatToStr(c);
```

Вопросы для самоконтроля:

1. С помощью какого оператора реализуется алгоритмическая структура «если ... то ... иначе ...»? Нарисуйте ее блок-схему.
2. С помощью какого оператора реализуется алгоритмическая структура «Выбор»? Нарисуйте ее блок-схему.
3. Перечислите основные логические операции.
4. Когда применяется условный оператор? Назовите два вида условного оператора.
5. Что позволяет делать оператор выбора?
6. Приведите пример на использование оператора условия. Приведите пример на использование оператора выбора.
7. Изучите и опишите информацию о компоненте `RadioButton` палитры `Standard` по справке Delphi.
8. В чем различие между объектами `CheckBox` и `RadioButton`?
9. Назначение объекта `RadioGroup`. Как задать список элементов-переключателей в панели `RadioGroup`?
10. По какому свойству `RadioGroup` определяется выбранный переключатель? Как задать этому свойству значение?

Основная часть

Задание 1.

Откройте проект, реализующий ввод двух целых чисел, их вещественное деление и вывод результата на экран (задание 4 из лабораторной работы № 1). Дополните программу обработчика события проверкой делителя на равенство нулю. В случае равенства делителя нулю вместо выполнения деления в текстовом окне должно отображаться сообщение «На ноль делить нельзя!»

Проверьте работу модифицированной программы. Сохраните внесенные в проект изменения, используя кнопку Save All панели инструментов Delphi.

Задание 2.

Измените программу таким образом, чтобы сообщение «На ноль делить нельзя!» в окне Edit3 выводилось красным цветом, а ширина окна изменялась так, чтобы отображался весь текст сообщения. Чтобы цвет шрифта и размер окна восстанавливался при вводе правильных данных, перед оператором `if..then..else` задайте свойства текстового окна: цвет шрифта – черный и ширину окна 65 пикселей. Сохраните внесенные в проект изменения.

Задание 3.

Измените программу таким образом, чтобы сообщение «На ноль делить нельзя!» выводилось в отдельном окне сообщения. Для этого используйте вызов процедуры `ShowMessage` («На ноль делить нельзя!»). Сохраните текст модуля под новым именем в той же папке, используя команду меню File → Save As. Сохраните измененный проект под новым именем в той же папке, используя команду меню File → Save As.

Задание 4.

Создайте приложение (Калькулятор), обеспечивающее ввод двух целых чисел и выполнение над ними арифметических операций: сложения, вычитания, умножения и вещественного деления. Для выбора операции используйте переключатели, вывод сообщения об ошибке при вводе делителя, равного нулю, выполните в отдельном окне сообщений.

Задания для самостоятельного выполнения

Задание 5.

Выполните индивидуальное задание первого уровня. Сохраните проект.

Организовать выполнение алгоритма следующим образом: выбор вычисления по той или иной формуле – с помощью элемента управления «переключатель», форматный или бесформатный вывод – с помощью элемента управления «флажок».

Вариант задания	Формулировка задачи
1.	Вычислить площадь и периметр прямоугольника, если задана длина одной стороны (a) и коэффициент n (%), позволяющий вычислить длину второй стороны (b = n*a).
2.	Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b.
3.	Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
4.	Вычислить площади геометрических фигур: прямоугольника и треугольника по заданным сторонам.
5.	Вычислить площади геометрических фигур: трапеции и круга. $S_т$=(a+b)h/2
6.	По известному радиусу вычислить объем и площадь поверхности шара. $V = \frac{4}{3}\pi r^3, s = 4\pi r^2$
7.	Заданы координаты трех вершин треугольника. Найти его периметр и площадь.
8.	Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
9.	Даны два числа. Вычислить их сумму, разность, произведение и частное.
10.	Дана сторона равностороннего треугольника. Найти площадь этого треугольника, его высоту, радиусы вписанной и описанной окружностей.

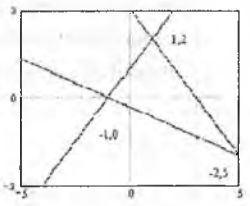
10.	$S=\frac{\sqrt{3}}{4}a^2, h=\frac{\sqrt{3}}{2}a, R=\frac{2}{\sqrt{3}}a, r=\frac{a}{2\sqrt{3}}$
11.	Вычислить объем и площадь полной поверхности цилиндра, если известны высота и радиус основания. $V=\pi r^2 h$, $S=2\pi r(r+h)$
12.	Заданы стороны прямоугольника. Определить его периметр площадь и длину диагонали.
13.	Заданы длина, ширина и высота параллелепипеда. Определить его объем и площадь поверхности.
14.	Для двух целых чисел A и B определить сумму S, разность R и среднее арифметическое SR.
15.	Переменной A присвоить ее значение, увеличенное в N раз, 2N раз, 3N раз.
16.	Поменяйте между собой значения переменных A и B, воспользовавшись третьей переменной C, без использования третьей переменной.
17.	В зависимости от названия реки, выдать сообщение о ее длине. В зависимости от названия горы, выдать сообщение о ее высоте.
18.	В зависимости от дня недели, выдать сообщение о его номере. В зависимости от номера месяца, выдать его название.
19.	Напечатать числа *a, b, c* в порядке возрастания, в порядке убывания.
20.	Найти количество отрицательных чисел среди *a, b, c* и абсолютное значение суммы этих чисел.

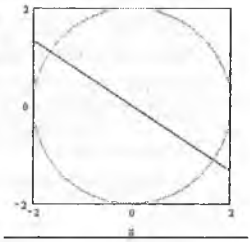
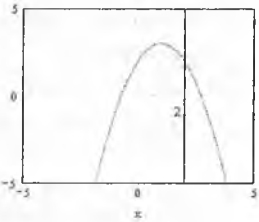
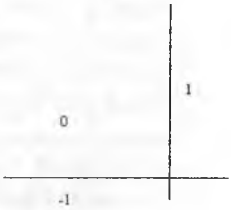
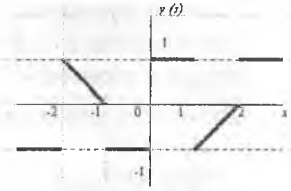
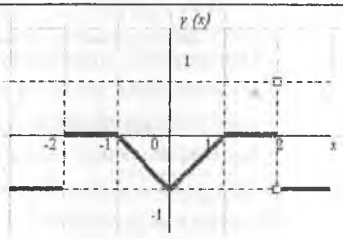
Задание 6.

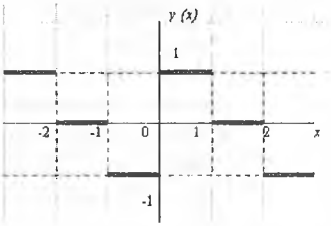
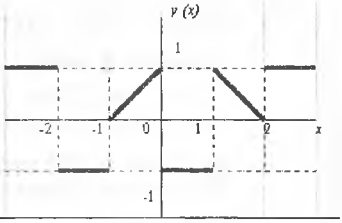
Выполните индивидуальное задание второго уровня. Сохраните проект.

Организовать разветвляющийся процесс для решения следующих задач.

Вариант задания	Формулировка задачи
1.	Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны, и в четвертую степень – отрицательные.

	Даны две точки $A(x_1, y_1)$ и $B(x_2, y_2)$. Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.
	Даны действительные числа x и y, не равные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее – их удвоенным произведением.
	На плоскости *XOY* задана своими координатами точка А. Указать, где она расположена (на какой оси или в каком координатном углу).
	Заданы три стороны треугольника: a, b и c. Определить, является ли этот треугольник прямоугольным, и какая сторона служит гипотенузой.
	Определить результат гадания на ромашке – «любит – не любит», взяв за исходное данное количество лепестков n.
	Заданы радиус круга R и сторона квадрата А. Определить, можно ли вписать квадрат в круг.
	Заданы два натуральных числа. Определить, является ли среднее арифметическое этих чисел целым числом.
).	Заданы три положительных числа a, b и c. Определить, являются ли они последовательно стоящими элементами арифметической прогрессии. Если являются, то определить разность прогрессии.
10.	Записать программу, которая на название фигуры (треугольник, квадрат, ромб, прямоугольник и т.п.) выводит формулу, по которой вычисляется площадь этой фигуры. В программе использовать оператор CASE.
11.	Записать программу, которая на ввод времени суток выводит соответствующее пожелание доброго утра, доброго дня, доброго вечера и спокойной ночи.
12.	Вычислить значение функции по формуле $y = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 1 \end{cases}$
13.	Определить, принадлежит ли некоторая точка М с произвольными координатами x,y закрашенной области. 

14.	Определить, принадлежит ли некоторая точка М с произвольными координатами x,y закрашенной области.	
15.	Определить, принадлежит ли некоторая точка М с произвольными координатами x,y закрашенной области.	
16.	Определить, принадлежит ли некоторая точка М с произвольными координатами x,y закрашенной области.	
17.	Записать программу, которая на ввод значения аргумента выдает значение функции, заданной графиком.	
18.	Записать программу, которая на ввод значения аргумента выдает значение функции, заданной графиком.	

19.	Записать программу, которая на ввод значения аргумента выдает значение функции, заданной графиком.	
20.	Записать программу, которая на ввод значения аргумента выдает значение функции, заданной графиком.	

Задание 7.

Выполните индивидуальное задание третьего уровня. Сохраните проект.

Организовать разветвляющийся процесс для решения следующих задач.

Вариант задания	Формулировка задачи
1.	Пройдет ли кирпич со сторонами *x, y, z* сквозь прямоугольное отверстие со сторонами *r* и *s*. Стороны отверстия должны быть параллельны граням кирпича.
2.	Можно ли коробку размером $a \times b \times c$ упаковать в посылку размером $r \times s \times t$? «Углом» укладывать нельзя.
3.	Можно ли из круглой заготовки радиуса r вырезать две прямоугольные пластинки с размерами $a \times b$ и $c \times d$?
4.	Лежит ли точка $M(x_m, y_m)$ внутри треугольника, заданного координатами своих вершин: $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$?
5.	Путник двигался t_1 часов со скоростью v_1, а затем t_2 часов – со скоростью v_2 и t_3 часов – со скоростью v_3. За какое время он одолел первую половину пути, после чего запланировал привал?
6.	Проверить, лежит ли окружность $(x-a_1)^2+(y-b_1)^2=r_1^2$

6.	целиком внутри окружности $(x-a_2)^2+(y-b_2)^2=r_2^2$ и наоборот.
7.	Можно ли на прямоугольном участке застройки размером a на b метров разместить два дома размером p на q и r на s метров? Дома можно располагать только параллельно сторонам участка.
8.	Дано число x. Вывести в порядке возрастания числа: $\sin x$, $\cos x$, $\ln x$. Если при каком-либо x некоторые из выражений не имеют смысла, вывести сообщение об этом и сравнивать значения только тех, которые имеют смысл.
9.	Даны три положительных числа. Определить, можно ли построить треугольник с длинами сторон, равным этим числам. Если можно, то ответить на вопрос, является ли он остроугольным.
10.	По заданным трем числам определить, является ли сумма каких-либо из них положительной.
11.	Даны действительные числа a, b, c. Удвоить эти числа, если $a < b < c$, и заменить их абсолютными значениями, если это не так.
12.	Осуществить перевод величин из радианной меры в градусную и наоборот. Программа должна запрашивать, какой перевод нужно осуществить и выполнять указанное действие.
13.	Определить, будут ли прямые $A_1x+B_1y+C_1$ и $A_2x+B_2y+C_2$ перпендикулярны. Если нет, то найти угол между ними.
14.	Известно, что из четырех чисел a_1, a_2, a_3 и a_4 одно отлично от трех других, равных между собой; присвоить номер этого числа переменной n.
15.	Даны три точки: $A(x_1,y_1)$, $B(x_2,y_2)$, $C(x_3,y_3)$. Определить, будут ли они расположены на одной прямой. Если нет, то вычислить угол $\angle ABC$.
16.	Написать программу решения уравнения $ax^3+bx=0$ для произвольных a, b.
17.	Даны три положительных числа a, b, c. Проверить, могут ли они быть длинами сторон треугольника. Если да, то вычислить его площадь.
18.	В доме M этажей и всего один подъезд; на каждом этаже по 3 квартиры; лифт может останавливаться только на нечетных этажах. Человек садится в лифт и набирает номер нужной ему квартиры N. На каком этаже остановится лифт?

19.	Написать программу решения системы линейных уравнений: $\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}$
20.	Проверить, не приводит ли к переполнению суммирование двух целых чисел *A* и *B* (т.е. к результату большему, чем 32767). Если переполнение, сообщить об этом, если нет – вывести сумму.

Лист самооценки выполнения лабораторной работы № 2.

Каждый пункт оценивается в 1 балл, если с уверенностью отвечаете «да». Максимальное количество баллов – 5.

1) Самостоятельно ответили на все вопросы из раздела «Вопросы для самоконтроля».
2) Самостоятельно выполнили задания № 1–4.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 3

Программирование алгоритмов циклической структуры в Delphi

6 часов

Вы научитесь:

- использовать компоненты `BitBtn`, `Memo`, `Image` и устанавливать их свойства;
- реализовывать в среде Delphi подпрограммы-функции и подпрограммы-процедуры,
- использовать операторы *цикла* при решении задач разветвляющейся структуры в среде Delphi.

Теоретическая часть

Часть 1. Циклические конструкции.

Циклические конструкции обеспечивают многократное выполнение одной и той же последовательности инструкций, которая называется *телом цикла.* Существуют два вида элементарных циклических структур:

– *циклы с параметром;*

– *итерационные циклы или циклы с условием.*

Циклы с параметром используют тогда, когда количество повторов тела цикла заранее известно.

Вспомните, каким образом записывается оператор цикла с параметром в языке программирования Паскаль.

Итерационные циклы используются тогда, когда число повторений заранее неизвестно, но задано условие окончания цикла. Причем, если условие окончания цикла проверяется перед выполнением тела цикла, то такие циклические структуры называют итерационными циклами *с предусловием* («Выполнять пока»), а если проверка условия происходит после выполнения тела цикла – итерационными циклами *с постусловием* («Выполнять до тех пор пока не»).

На практике циклы с условием чаще всего используют в двух случаях:

• Число повторений заранее неизвестно (например, цикл до достижения требуемой точности результата).

• Число повторений заранее известно, но шаг параметра цикла не равен 1 (или –1).

Итерационные циклы *с предусловием* реализуются с помощью оператора `While`, а итерационные циклы *с постусловием* – с помощью оператора `Repeat … Until`.

Вспомните, каким образом работают циклы с предусловием и с постусловием в языке программирования Паскаль.

Рекуррентной называется всякая формула, выражающая каждый член последовательности через предыдущие члены этой последовательности. Используется чаще всего в целях избавления от больших чисел при вычислении суммы членов последовательности.

Как правило, в этом случае рекуррентная формула имеет вид: $a_{k+1} = c \cdot a_k$, откуда, зная общий член последовательности, можно будет найти коэффициент С, на который нужно умножать каждый предыдущий член последовательности, чтобы найти следующий:

$$c = \frac{a_{k+1}}{a_k}$$

Пример. Вычислить сумму бесконечного ряда: $S = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{2k-1}{(2k)!} x^{2k}$

$$a_{k+1} = (-1)^k \frac{2k+1}{(2k+2)!} x^{2k+2} \qquad a_k = (-1)^{k-1} \frac{2k-1}{(2k)!} x^{2k}$$

$$c = \frac{(-1)^k (2k+1) \cdot x^{2k+2} (2k)!}{(2k+2)! \cdot (-1)^{k-1} (2k-1) x^{2k}} = -\frac{(2k+1) x^2 (2k)!}{(2k-1)(2k+2)!} = -\frac{x^2 (2k+1)}{(2k-1)(2k+1)(2k+2)} = -\frac{x^2}{(2k-1)(2k+2)}$$

Таким образом, очередной член ряда можно вычислить по рекуррентной формуле: $a_{k+1} = -\frac{x^2}{(2k-1)(2k+2)} a_k$

Если в программе возникает необходимость частого обращения к некоторой группе операторов, то рационально сгруппировать такую группу операторов в самостоятельный блок, к которому можно обращаться, указывая его имя. Такие самостоятельные программные блоки называются *подпрограммами пользователя.*

Передача данных из главной программы в подпрограмму и возврат результата выполнения осуществляется с помощью параметров. Различают *формальные параметры* – параметры, определенные в заголовке подпрограммы, и *фактические параметры* – выражения, задающие конкретные значения при обращении к подпрограмме.

Реализуют подпрограммы в виде процедур или функций, которые определяются в разделе описания функций и процедур.

Вспомните, каким образом оформляется раздел описания функций и процедур в языке программирования Паскаль.

Главное отличие функции от процедуры заключается в том, что результат работы функции – единственное значение, а результат работы процедуры – одно значение, несколько значений или ни одного. Кроме того, обращение к функции является разновидностью операнда, а вызов процедуры – разновидностью оператора.

Структура процедуры:

```
Procedure <имя процедуры> (список формальных параметров);
<Раздел описаний программного кода процедуры>
Begin
<Операторы тела процедуры>
End;
```

Список формальных параметров может включать:

- параметры-значения или входные параметры, значения которых должны быть установлены до начала работы данной процедуры (определяют исходные данные для работы процедуры);

- параметры-переменные или выходные параметры, получающие свое конкретное значение в результате работы процедуры (определяют выходные данные процедуры). Перед перечислением параметров-переменных в списке формальных параметров должно стоять ключевое слово `var`.

– Каждый параметр имеет имя и тип, указанный через «:». Параметры отделяются друг от друга «;».

Обращение к процедуре осуществляется в основной программе путем задания ее имени и списка фактических параметров того же типа и количества, что и формальные.

Функции – это подпрограммы, в результате которых вычисляется только одно значение, которое присваивается имени функции.

Структура функции:

```
Function <имя функции>(список формальных
параметров):<тип результата>;
<Раздел описаний программного кода функции>
Begin
<Операторы тела функции>
<Имя функции>:=<Вычисленное значение>
End;
```

Список формальных параметров включает имена переменных со своими типами, с помощью которых определяются исходные данные, необходимые для работы функции.

В разделе операторов должен присутствовать, по крайней мере, один оператор, присваивающий вычисленное значение имени функции. В точку вызова возвращается результат последнего такого присваивания. Если такого оператора нет, то значение функции не определено.

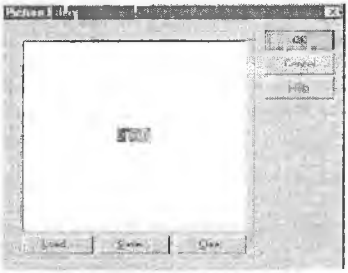
Обращение к функции осуществляется в основной программе путем задания ее имени и списка фактических параметров того же типа и количества, что и формальные.

Приведите по одному примеру программы в языке программирования Паскаль с использованием процедуры и функции.

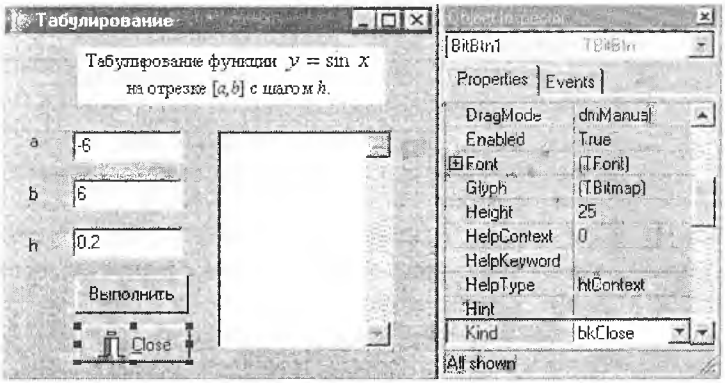
Если процедура (или функция) обращается сама к себе как к процедуре *(или функции) непосредственно или через цепочку подпрограмм, то это называется* рекурсией. Для того чтобы подобного типа программы не зацикливались (что очень реально), в первую очередь необходимо обеспечить выход из рекурсии.

Каким образом можно обеспечить выход из рекурсии в языке программирования Паскаль?

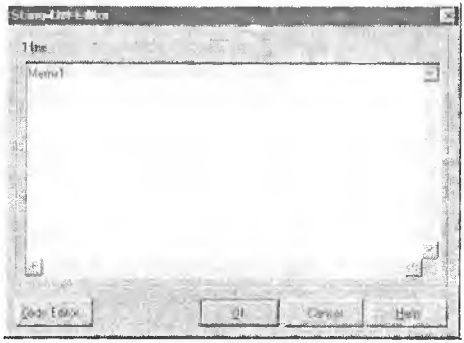
Компонент **BitBtn** (кнопка с изображением)  . Используется как обычная кнопка для инициирования некоторого события, но может содержать графическое изображение. Относится к группе `Additional`.

`Caption`	Задает текст надписи на кнопке.
`Kind`	Предлагает на выбор десять предопределенных типов кнопок.
`Glith`	Предлагает создание собственного типа кнопки с помощью окна Picture Editor. 

Пример использования в программе:



Компонент **Memo** (многострочное окно редактирования)  . Используется для ввода, отображения и редактирования многострочных текстов. Относится к группе `Standard`.

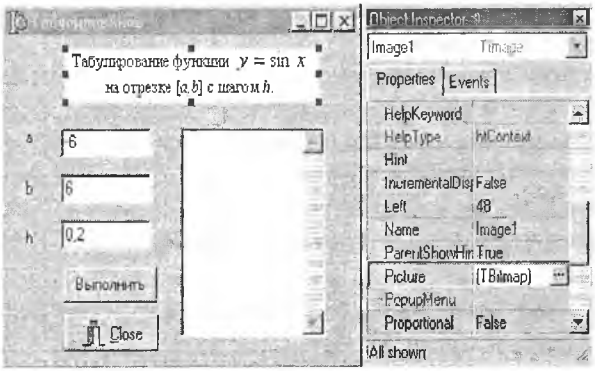
Alignment	Задает режим выравнивания текста внутри Memo.
AutoSize	Задает необходимость изменения размера компонента при изменении размера шрифта.
BorderStyle	Задается стиль обрамления Memo.
Color	Задает цвет, которым изображается элемент Memo на экране.
Lines	Определяет текст, который будет выведен построчно в окне Memo при запуске программы. Текст задается в окне String List Editor. 
MaxLength	Позволяет ограничивать число вводимых пользователем символов.
ScrollBars	Задает наличие полос прокрутки.
Text	Используется, чтобы получить текст компонента Memo как одну строку. Значение этого свойства не отображается в окне Object Inspector, к нему можно обратиться только во время выполнения программы.

Компонент **Image** (графический образ)  . Позволяет отображать рисунок, загруженный из графического файла. Относится к группе Additional.

Align	Задает режим выравнивания расположения объекта Image внутри формы.
Picture	Задает имя файла графического файла с рисунком с помощью окна Picture Editor.

	
`Streth`	Задает разрешение на автоматическое масштабирование рисунка относительно Image.

Пример использования в программе:



Вопросы для самоконтроля:

1. Каково назначение операторов цикла?
2. Какие операторы цикла вы знаете?
3. Когда используется цикл с параметром `For` (ДЛЯ)?
4. Когда используются циклы с предусловием или постусловием?

5. Как изменяется управляющая переменная в цикле For?
6. Когда в цикле применяется составной оператор?
7. Когда выполняется тело оператора цикла с предусловием While (ПОКА)?
8. Когда выполняется тело оператора цикла Repeat с постусловием (ДО)?
9. Когда осуществляется выход из оператора Repeat?
10. Когда осуществляется выход из оператора While?
11. Чем отличаются циклы While и Repeat?
12. Для чего используется объект Memo? Как используется свойство Text компоненты Memo?
13. Для чего необходимы коды ASCII-таблицы 13 и 10?
14. Для чего используется свойство Lines компоненты Memo?
15. Как заполняется свойство Lines компоненты Memo при проектировании проекта? Как заполняется свойство Lines компоненты Memo при выполнении проекта?
16. Для чего используется объект BitBtn?
17. Для чего используется объект Image?
18. Что такое подпрограмма?
19. Что такое функция и процедура?
20. Опишите различия между функцией и процедурой.
21. В каком разделе описываются процедуры и функции?
22. Что такое формальные параметры?
23. Что такое фактические параметры?
24. В чем разница между параметрами-значениями и параметрами-переменными?
25. Как осуществляется обращение к функциям и переменным?

Основная часть

Задание 1.

Протабулировать функцию (найти значения функции) $y=\sin x$ на отрезке $[a, b]$ с шагом h.

Программа, реализующая данный алгоритм имеет вид:

```
var
a,b,h,x,y: real;
n,i: integer;
begin
a:=StrToFloat(Edit1.Text); {Начало отрезка}
b:=StrToFloat(Edit2.Text); {Конец отрезка}
h:=StrToFloat(Edit3.Text); {Шаг табуляции}
n:=ceil((b-a)/h)+1;
x:=a;
for i:=1 to n do
begin
y:=sin(x);
x:=x+h;
end;
end;
```

Задание 2.

Подсчитать количество цифр в заданном целом числе.

Программа, реализующая данный алгоритм, будет иметь вид:

```
var
a,k: integer;
begin
a:=StrToInt(Edit1.Text);
k:=0;
while a<>0 do
begin
k:=k+1;
a:=a div 10;
end;
Edit2.Text:=IntToStr(k);
end;
```

Задание 3.

Вычислить сумму бесконечного ряда $y=\sum_{1}^{\infty}\frac{k+0.3}{3k^2+5}$ с заданной точностью Eps, (т.е. вычислить сумму всех членов последовательности $\frac{k+0.3}{3k^2+5}$, не меньших заданного числа Eps).

Программа, реализующая данный алгоритм будет иметь вид:

```
var
  k: integer;
  e,y,s: real;
begin
e:=StrToFloat(Edit1.Text);
k:=1;
s:=0;
repeat
  y:=(k+0.3)/(3*k*k+5); {Вычисление очередного члена
последовательности}
  s:=s+y;   {Вычисление суммы последовательности}
  k:=k+1;  {Вычисление количества членов последова-
тельности}
until y<=e;
{Вывод полученной суммы}
Edit2.Text:=FloatToStr(s)+'  '+IntToStr(k-1);
end;
```

Задание 4.

Протабулировать функцию (найти значения функции) *y=sin x* на отрезке *[a, b]* с шагом *h*. При этом:

А) использовать заполнение Memo с использованием свойства Text,

Б) Memo заполняется с использованием свойства Lines. Метод Add, примененный к Lines, позволяет добавить строку в Memo. Оператор memo1.Lienes[0] :=' x | y '; задает первую строку в Memo.

А) Программа, реализующая данный алгоритм, будет иметь вид:

```
{Табулирование функции на отрезке}
var
a,b,h,x,y: real;
begin
a:=StrToFloat(Edit1.Text); {Начало отрезка}
b:=StrToFloat(Edit2.Text); {Конец отрезка}
h:=StrToFloat(Edit3.Text); {Шаг табуляции}
n:=ceil((b-a)/h)+1;
```

```
x:=a;

Memo1.Clear;
{Формирование заголовка таблицы табуляции, которая выводится
в Memo}
Memo1.Lines[0]:='  x  |  y  ';
Memo1.Lines.Add('-----------------');
While x<=b do
begin
y:=sin(x); {Вычисление следующего значения функции}
{Формирование очередной строки  с результатами табулирова-
ния}
Memo1.Lines.Add(FloatToStrF(x,ffFixed,5,2)+'    |    '
+FloatToStrF(y,ffFixed,5,2));
x:=x+h; {Вычисление следующей точки табуляции}
end;
end;
```

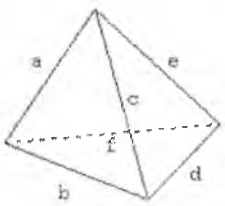
Б) Программа, реализующая данный алгоритм, будет иметь вид:

```
var
a,b,h,x,y: real;
n,i: integer;
begin
a:=StrToFloat(Edit1.Text); {Начало отрезка}
b:=StrToFloat(Edit2.Text); {Конец отрезка}
h:=StrToFloat(Edit3.Text); {Шаг табуляции}
n:=ceil((b-a)/h)+1;        {Расчет количества точек}
x:=a;
{Формирование заголовка таблицы табуляции, которая выводится
в Memo. Добавление в конец строки chr(13)+chr(10) - кодов 13
- возврат каретки и 10 - переход на одну строку, позволяет
разбить таблицу на строки}
Memo1.Text:='  x  |  y  '+chr(13)+chr(10);
Memo1.Text:= Memo1.Text+'-----------------' +chr(13)+chr(10);
for i:=1 to n do
begin
y:=sin(x); {Вычисление следующего значения функции}
{Формирование очередной строки  с результатами табулирова-
ния}
Memo1.Text:= Memo1.Text+FloatToStrF(x,ffFixed,5,2)+'   |   '
+FloatToStrF(y,ffFixed,5,2)+chr(13)+chr(10);
x:=x+h; {Вычисление следующей точки табуляции}
end;
end;
```

Задание 5.

Вычислить площадь четырехгранника, если даны длины его ребер. Использовать подпрограмму-процедуру и подпрограмму-функцию.

Код программы с использованием подпрограммы-процедуры.



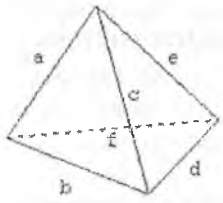
Вычисление площади треугольника по формуле Герона оформлено с помощью процедуры.

```
Вычисление площади треугольника по формуле Герона оформлено с помощью процедуры.

var
a,b,c,d,e,f,s: real;
s1,s2,s3,s4: real;
{Описание процедуры вычисления площади треугольника по формуле Герона}
procedure Sq(x,y,z: real; var s: real);
var
p: real;
begin
  p:=(x+y+z)/2;
  s:=sqrt(p*(p-x)*(p-y)*(p-z));
end;
begin
{Задание длин ребер четырехгранника}
a:=StrToFloat(Edit1.Text);
b:=StrToFloat(Edit2.Text);
c:=StrToFloat(Edit3.Text);
d:=StrToFloat(Edit4.Text);
e:=StrToFloat(Edit5.Text);
f:=StrToFloat(Edit6.Text);
{Обращение к процедуре для вычисления площади очередной грани}
Sq(a,b,c,s1); Sq(c,d,e,s2);
Sq(b,e,f,s3); Sq(a,f,d,s4);
{Вычисление площади поверхности четырехгранника}
s:=s1+s2+s3+s4;
Edit7.Text:=FloatToStr(s);
end;
```

Код программы с использованием подпрограммы-функции.

```
var
a,b,c,d,e,f,s: real;
{Описание функции вычисления площади треугольника по формуле Герона}
function Sq(x,y,z: real): real;
var
p: real;
```



Вычисление площади треугольника по формуле Герона оформлено с помощью функции.

```
begin
  p:=(x+y+z)/2;
  sq:=sqrt(p*(p-x)*(p-y)*(p-z));
end;
begin
{Задание длин ребер четырехгранника}
a:=StrToFloat(Edit1.Text);
b:=StrToFloat(Edit2.Text);
c:=StrToFloat(Edit3.Text);
d:=StrToFloat(Edit4.Text);
e:=StrToFloat(Edit5.Text);
f:=StrToFloat(Edit6.Text);
{Обращение к функции для вычисления площади каждой грани и вычисление площади поверхности четырехгранника }
S:=Sq(a,b,c)+Sq(c,d,e)+Sq(b,e,f)+Sq(a,f,d);
Edit7.Text:=FloatToStr(s);
end;
```

Задание 6.

Вычислить факториал заданного числа с помощью подпрограммы-функции, использующей рекурсию.

```
var
n: integer;
F: longint;
{Описание функции вычисления факториала,
n - формальный параметр-значение типа integer,
Результат выполнения функции типа longint}
function Fakt(n: integer): longint;
begin
  if n=1 then Fakt:=1      {Проверка условия завершения рекурсии}
  else Fakt:=n*Fakt(n-1) {Рекурсивное вычисление n!}
end;
{Начало главной программы}
begin
n:=StrToInt(Edit1.Text);
{Вызов функции для фактического параметра n}
F:=Fakt(n);
Edit2.Text:=IntToStr(F);
end;
```

Задания для самостоятельного выполнения

Задание 7.

Выполните индивидуальное задание из предложенных задач первого уровня.

7.1. Табулирование функции

Вариант задания	Функция	Границы отрезка и шаг
1.	$y = 3\sin\sqrt{x} + 0{,}35x - 3{,}8$	[2,3] 0,1
2.	$y = 0{,}25x^3 + x - 1{,}2502$	[0,2] 0,2
3	$y = x + \sqrt{x} + \sqrt[3]{x} - 2{,}5$	[0,4;1] 0,05
4.	$y = \sin(\ln x) - \cos(\ln x) + 2\ln x$	[1;3] 0,2
5.	$y = \cos\frac{2}{x} - 2\sin\frac{1}{x} + \frac{1}{x}$	[1;2] 0,1
6.	$y = 3x - 4\ln x - 5$	[2;4] 0,2
7.	$y = \sqrt{1-x} - \cos\sqrt{1-x}$	[0;1] 0,1
8.	$y = tgx - \frac{1}{3}tg^3x + \frac{1}{5}tg^5 - \frac{1}{3}$	[0;0,8] 0,05
9.	$y = 0{,}1x^2 - x\ln x$	[1;2] 0,1
10.	$y = x - 1/(3 + \sin 3{,}6x)$	[0;0,85] 0,05
11.	$y = x + \cos(x^{0{,}52} + 2)$	[0,5;1] 0,05
12.	$y = 3\ln^2 x + 6\ln x - 5$	[1;3] 0,2
13.	$y = 3x - 14 + e^x - e^{-x}$	[1;3] 0,2
14.	$y = \sqrt{1-x} - tgx$	[0;1] 0,1
15.	$y = \cos x - e^{-\frac{x^2}{2}} + x - 1$	[1;2] 0,1

7.2. Вычисление суммы ряда

По заданной формуле члена последовательности с номером k составить две программы:

– программу вычисления суммы первых n членов последовательности $(k=1,2,3..,n)$;

– программу вычисления суммы всех членов последовательности, не меньших заданного числа e.

Вариант задания	Член послед.	Вариант задания	Член послед.	Вариант задания	Член послед.
1.	$\frac{1}{(2k-1)(2k+1)}$	6.	$\frac{k+4}{(k^2+2)(k+8)}$	11.	$\frac{k+2}{k^2+4}$
2.	$\frac{k}{(k+1)^2+3}$	7.	$\frac{3(k+1)}{7k^2+9}$	12.	$\frac{2k}{3k+k^2+4}$
3.	$\frac{2k}{(k^2+1)(k+2)}$	8.	$\frac{1}{\sqrt{k}+15}$	13.	$\frac{\sqrt{k+1}}{2\sqrt{k^2+1}}$
4.	$\frac{k+1}{k-\sqrt{k+2}}$	9.	$\frac{1}{k^2+3k+4}$	14.	$\frac{4k}{5k^2+8k-1}$
5.	$\frac{k}{(3k^2+7)(k^2+1)}$	10.	$\frac{k+1}{k(k+2)(k+3)}$	15.	$\frac{2k+1}{(2k^2+3)k}$

7.3. Арифметические задачи

1. Определить сумму цифр введенного числа a (a<1000).

2. Определить сумму чисел от 3 до 99, кратных числу 3.

3. Вывести на экран таблицу квадратов целых чисел от 1 до 10.

4. Напечатать таблицы температур по Цельсию от 0 до 100 градусов с дискретностью в один градус и их эквивалентов по шкале Фаренгейта, используя для перевода формулу $t_F = 9 \cdot t_C / 5 + 32$.

5. Вывести в столбец произведения чисел a = 143, b = 777 и числа c, последовательно принимающего значения 1, 2, 3, ... 9.

6. Получить произведения числа a = 12345689 на числа 9, 18, 27, ... 81.

7. Произвести суммирование натуральных чисел 1, 2, 3,..., пока их сумма s не станет равной или превысит величину h. Вывести на экран последнее слагаемое и значение суммы.

8. Сколько чисел последовательности 2, 4, 6, 8, ... нужно взять, чтобы их сумма превысила 1000? Вывести величину последнего слагаемого и суммы.

9. Определить количество цифр в натуральном числе N.

10. Вычислить факториал натурального числа N.

11. Определить произведение цифр натурального числа N.

12. Найти сумму всех четных натуральных чисел от 1 до 100.

13. Найти сумму первой и последней цифры натурального числа N.

14. Дано натуральное число. Верно ли, что оно начинается и заканчивается одной и той же цифрой?

15. Дано натуральное число. Верно ли, что в данном числе нет данной цифры A? A задается.

16. Дано натуральное число. Верно ли, что оно заканчивается нечетной цифрой?

17. Дано натуральное число. Верно ли, что в данном числе цифра A встречается более двух раз? A задается.

18. Дано натуральное число. Верно ли, что в данном числе сумма цифр больше A, а само число делится на A? A задается.

19. Дано натуральное число. Верно ли, что число принадлежит промежутку от A до B и кратно 3, 4 и 5? A и B задаются.

20. Сколько раз первая цифра встречается в данном числе?

Задание 8.

Выполните индивидуальное задание из предложенных задач второго уровня.

8.1. Табулирование функции

Вариант задания	Функция	Исходные данные	Диапазон и шаг изменения аргумента
1.	$y=\begin{cases} at^{2bt}, & 1\le t\le 2 \\ 1, & t<1 \\ e^{at}\cos bt, & t>2 \end{cases}$	$a=-0.5$ $b=2$	$t\in[0;3]$ $\Delta t=0.15$
2.	$y=\begin{cases} \pi x^2-7/x^2, & x<1.3 \\ ax^3+7\sqrt{x}, & x=1.3 \\ \lg(x+7\sqrt{x}), & x>1.3 \end{cases}$	$a=1.5$	$x\in[0.8;2]$ $\Delta x=0.1$
3.	$y=\begin{cases} ax^2+bx+c, & x<1.2 \\ a/x+\sqrt{x^2+1}, & x=1.2 \\ (a+bx)/\sqrt{x^2+1}, & x>1.2 \end{cases}$	$a=2.8$ $b=-0.3$ $c=4$	$x\in[1;2]$ $\Delta x=0.05$
4.	$y=\begin{cases} \pi x^2-7/x^2, & x<1.34 \\ ax^3+7\sqrt{x}, & x=1.4 \\ \ln(x+7\sqrt{\lvert x+a\rvert}), & x>1.4 \end{cases}$	**a=1,65**	$x\in[0.7;2]$ $\Delta x=0.1$
5.	$y=\begin{cases} 1.5\cos^2 x, & x=1 \\ 1.8ax, & 1<x<2 \\ (x-2)^2+6, & x>2 \end{cases}$	$a=2.3$	$x\in[0.2;8]$ $\Delta x=0.2$
6.	$y=\begin{cases} x\sqrt[3]{x-a}, & x>a \\ x\sin ax, & x=a \\ e^{-ax}\cos ax, & x<a \end{cases}$	$a=2.5$	$x\in[1;5]$ $\Delta x=0.5$
7.	$y=\begin{cases} bx-\lg bx, & bx<1 \\ 1, & bx=1 \\ bx+\lg bx, & bx>1 \end{cases}$	$b=1.5$	$x\in[0.1;1]$ $\Delta x=0.1$
8.	$y=\begin{cases} \sin x\lg x, & x>3.5 \\ \cos^2 x, & x\le 3.5 \end{cases}$	-	$x\in[2;5]$ $\Delta x=0.25$

9.	$y=\begin{cases}\lg(x+1), & x>1\\ \sin^2\sqrt{\lvert ax\rvert}, & x\le 1\end{cases}$	$a=20.3$	$x\in[0.5;2]$ $\Delta x=0.1$
10.	$y=\begin{cases}(\ln^3 x+x^2)/\sqrt{x+t}, & x<0.5\\ \sqrt{x+t}+1/x, & x=0.5\\ \cos x+t\sin^2 x, & x>0.5\end{cases}$	$t=2.2$	$x\in[0.2;2]$ $\Delta x=0\,2$
11.	$y=\begin{cases}\dfrac{a+b}{e^x+\cos x}, & x<2.8\\ (a+b)/(x+1), & 2.8\le x<6\\ e^x+\sin x, & x\ge 6\end{cases}$	$a=2.6$ $b=-0.39$	$x\in[0;7]$ $\Delta x=0.5$
12.	$y=\begin{cases}a\lg x+\sqrt[3]{\lvert x\rvert}, & x>1\\ 2a\cos x+3x^2, & x\le 1\end{cases}$	$a=0.9$	$x\in[0.8;2]$ $\Delta x=0.1$
13.	$\overline{y=\begin{cases}\dfrac{a}{i}+bi^2+c, & i<4\\ i, & 4\le i\le 6\\ ai+bi^3, & i>6\end{cases}}$	$a=2.1$ $b=1.8$ $c=-20.5$	$\overline{i\in[0;12]}$ $\Delta i=1$
14.	$y=\begin{cases}a\sin\left(\dfrac{i^2+1}{n}\right), & \sin\left(\dfrac{i^2+1}{n}\right)>0\\ \cos\left(i+\dfrac{1}{n}\right), & \sin\left(\dfrac{i^2+1}{n}\right)\le 0\end{cases}$	$a=0.3$ $n=10$	$\overline{i\in[1;10]}$ $\Delta i=1$
15.	$y=\begin{cases}\sqrt{at^2+b\sin t+1}, & t<0.1\\ at+b, & t=0.1\\ \sqrt{at^2+b\cos t+1}, & t>0.1\end{cases}$	$a=2.5$ $b=0.4$	$t\in[-1;1]$ $\Delta t=0.2$

8.2. Вычисление суммы ряда

Составить две программы:

– программу вычисления суммы первых n членов последовательности ($k=1,2,3..,n$);

– программу вычисления суммы всех членов последовательности, не меньших заданного числа e.

При вычислении факториала использовать рекурсивную подпрограмму-функцию.

Вар. зад.	Ряд	Вар. зад.	Ряд	Вар. зад.	Ряд
1.	$\sum_{k=1}^{n} \frac{k^4}{k!}$	6.	$\sum_{k=1}^{n} \frac{k!}{(5+k-1)!}$	11.	$\sum_{k=1}^{n} \frac{(k+1)^3}{k!}$
2.	$\sum_{k=0}^{n} \frac{(-1)^k}{k!(k+1)!}$	7.	$\sum_{k=1}^{n} \frac{k}{(k+1)!}$	12.	$\sum_{k=0}^{n} \frac{(-1)^k}{(2k)!}$
3.	$\sum_{k=0}^{n} \frac{(-1)^k}{k!}$	8.	$\sum_{k=0}^{n} \frac{1}{(2k)!}$	13.	$\sum_{k=0}^{n} (-1)^k \frac{1}{k!(k+1)!}$
4.	$\sum_{k=1}^{n} (-1)^k \frac{1}{(k!)^2}$	9.	$\sum_{k=1}^{n} \frac{(-1)^{k-1}}{(2k-1)!}$	14.	$\sum_{k=0}^{n} \frac{1}{k!}$
5.	$\sum_{k=0}^{n} \frac{1}{(k+2)!(k+3)!}$	10.	$\sum_{k=0}^{n} \frac{(-1)^k}{k!} k$	15.	$\sum_{k=1}^{n} (-1)^k \frac{k}{(k!)^2+(k+1)!}$

8.3. Арифметические задачи

1. В переменную последовательно вводятся числа. Окончание ввода либо по желанию пользователя, либо когда сумма отрицательных чисел превысит – 1000. Определить среднее арифметическое отрицательных чисел.

2. В переменную последовательно вводятся десять чисел. Определить среднее арифметическое отрицательных чисел.

3. В простую переменную последовательно вводятся N вещественных чисел. Вычислить максимальное значение.

4. В простую переменную последовательно вводятся N чисел. Сколько чисел больше своих соседей слева?

5. В простую переменную последовательно вводятся N чисел. Все ли числа меньше заданного числа К?

6. Найти наибольшую и наименьшую цифры в записи данного натурального числа.

7. Дано натуральное число $N \leq 99$. Дописать к нему цифру К в конец и начало.

8. При каком натуральном числе N произведение предшествующего числа и числа, следующего за N, равно 2208?

9. Существуют ли натуральные числа a < 100, которые обладают следующими свойствами: а) a mod 3 = 1, б) a mod 4 = 2, в) a mod 5 = 3, г) a mod 6 = 4. Сколько их?

10. Найти все трехзначные числа, сумма цифр которых равна *A*, а само число делится на *B*. *A* и *B* задаются.

11. Найти все четырехзначные числа, у которых сумма крайних цифр равна сумме средних цифр, а само число делится на 6 и 27.

12. Найти все четырехзначные числа, в которых есть две одинаковые цифры.

13. Найти количество трехзначных чисел, сумма цифр которых равна *A*, а само число заканчивается цифрой *B*. *A* и *B* задаются.

14. Найти все двузначные числа, которые при умножении на 2 заканчиваются на 8, а при умножении на 3 – на 4.

15. Найти количество делителей натурального числа. Сколько из них четных?

16. Найти количество делителей натурального числа, больших *K*. *K* задается.

17. Найти все натуральные числа *a, в* и *c* из интервала от 1 до 20, для которых выполняется равенство a2 + в2 = c2.

18. Найти сумму нечетных делителей натурального числа.

Задание 9.

Выполните индивидуальное задание из предложенных задач третьего уровня.

9.1. Табулирование функции и суммы функционального ряда

Протабулировать:

1. функцию y на отрезке $[a, b]$ с шагом h;
2. сумму S функционального ряда разложения этой функции на отрезке $[a, b]$ с шагом h.

Сравнить значения заданной функции и ее разложения.

Вид вычисляемой функции задать с помощью подпрограммы-функции. Функциональный ряд S вычисляется с точностью до ε по соответствующей рекуррентной формуле.

Возможная форма проекта:

Вар зад	Сумма	Диапазон изменения аргумента	ε	Функция y
1.	$S=1+\frac{\ln 3}{1!}x+\frac{\ln^2 3}{2!}x^2+\ldots+\frac{\ln^n 3}{n!}x^n+\ldots$	$0,1\le x\le 1$	10^{-4}	3^x
2.	$S=x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}+\ldots$	$0,1\le x\le 1$	10^{-4}	$\sin(x)$
3.	$S=1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}+\ldots$	$1\le x\le 2$	15^{-4}	e^x
4.	$S=1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}+\ldots$	$0,1\le x\le 1$	10^{-4}	$\cos(x)$
5.	$S=x+\frac{x^5}{5}+\ldots+\frac{x^{4n+1}}{4n+1}+\ldots$	$0,1\le x\le 0,8$	30^{-5}	$\frac{1}{4}\ln\frac{1+x}{1-x}+\frac{1}{2}\operatorname{arctg}x$
6.	$S=1+3x^2+\ldots+\frac{2n+1}{n!}x^{2n}+\ldots$	$0,1\le x\le 1$	10^{-4}	$\left(1+2x^2\right)e^{x^2}$
7.	$S=\frac{x-1}{x+1}+\frac{1}{3}\left(\frac{x-1}{x+1}\right)^3+\ldots+\frac{1}{2n+1}\left(\frac{x-1}{x+1}\right)^{2n+1}+\ldots$	$0,2\le x\le 1$	10^{-4}	$\frac{1}{2}\ln x$
8.	$S=1+\frac{x^2}{2!}+\ldots\frac{x^{2n}}{(2n)!}+\ldots$	$0,1\le x\le 1$	10^{-4}	$\frac{e^x+e^{-x}}{2}$
9.	$S=1+\frac{2x}{1!}+\ldots+\frac{(2x)^n}{n!}+\ldots$	$0,1\le x\le 1$	20^{-4}	e^{2x}
10.	$S=1+2\frac{x}{2}+\ldots+\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n+\ldots$	$0,1\le x\le 1$	30^{-4}	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
11.	$S=x-\frac{x^3}{3}+\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}+\ldots$	$0,1\le x\le 0,5$	40^{-5}	$\operatorname{arctg}(x)$
12.	$S=-\frac{(2x)^2}{2}+\frac{(2x)^4}{24}+\ldots+(-1)^n\frac{(2x)^{2n}}{(2n)!}+\ldots$	$0,1\le x\le 1$	15^{-4}	$2\left(\cos^2 x-1\right)$
13.	$S=-(1+x)^2+\frac{(1+x)^4}{2}+\ldots+(-1)^n\frac{(1+x)^{2n}}{n}+\ldots$	$-2\le x\le -0,1$	40^{-5}	$\ln\frac{1}{2+2x+x^2}$
14.	$S=x+\frac{x^3}{3!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}+\ldots$	$0,1\le x\le 1$	20^{-4}	$\frac{e^x-e^{-x}}{2}$
15.	$S=3x+8x^2+\ldots+n(n+2)x^n+\ldots$	$0,1\le x\le 0,8$	40^{-5}	$\frac{x(3-x)}{(1-x)^3}$

9.2. Арифметические задачи

1. В простую переменную последовательно вводятся 10 чисел. Определить максимальное значение и количество чисел, равных максимуму.

2. В простую переменную последовательно вводятся N положительных чисел. Определить минимальное значение и порядковый номер последнего числа, равного этому значению.

3. В простую переменную последовательно вводятся числа. Окончание ввода 0. Сколько чисел больше своих соседей слева и справа?

4. В простую переменную последовательно вводятся N чисел, отличных от нуля. Положительные или отрицательные числа завершают последовательность и сколько их?

5. Вводится положительное целое число B < 10000. Является ли оно палиндромом? Палиндром – это строка или число, которое читается как слева направо, так и справа налево?

6. Найти все простые числа, лежащие в заданном диапазоне. Простые числа – это числа больше 1 и делящиеся нацело только на 1 и на само себя.

7. Найти все делители натурального числа N.

8. Проверить, все ли цифры данного натурального числа N различны.

9. Поменять порядок следования цифр в натуральном числе N на обратный.

10. Определить все двузначные числа, сумма квадратов цифр которых кратна числу 15.

11. Найти совершенные числа, меньшие заданного числа N. Совершенные числа – это числа, которые равны сумме всех своих собственных делителей, включая 1 (6 = 1+2+3).

12. Проверить утверждение, что разность любого натурального числа и суммы его цифр кратна 9, для всех чисел, лежащих между заданными *m* и *n*.

13. Дано натуральное число. Приписать к нему такое же число.

14. Из данного натурального числа удалить все цифры *A*. *A* задается.

15. Найти все симметричные натуральные числа (палиндромы) из промежутка от *A* до *B*. *A* и *B* задаются.

16. Найти количество различных цифр данного натурального числа.

17. Найти все натуральные числа из промежутка от 1 до 200, у которых количество делителей равно *N*. *N* задается.

18. Найти сумму целых чисел из промежутка от 1 до 200, у которых ровно 5 делителей.

Лист самооценки выполнения лабораторной работы № 3.

Каждый пункт оценивается в 2 балла, если с уверенностью отвечаете «да». Максимальное количество баллов – 10.

1) Самостоятельно ответили на все вопросы из раздела «Вопросы для самоконтроля».
2) Самостоятельно выполнили задания № 1–6.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 4

Работа с одномерными массивами в Delphi

4 часа

Вы научитесь:

- использовать компонент `GroupBox` и устанавливать его свойства;
- реализовывать ввод, редактирование, поиск и сортировку данных через компонент `GroupBox`;
- использовать данный компонент при решении задач с использованием одномерных массивов в среде Delphi.

Теоретическая часть

Многие задачи, которые решаются с помощью компьютера, связаны с обработкой больших объемов информации, представляющей совокупность данных, объединенных единым математическим содержанием или связанных между собой по смыслу. Такие данные удобно представлять в виде линейных или прямоугольных таблиц.

В программе для представления таких данных используются массивы.

Вспомните, что называют массивом. Приведите примеры, когда необходимо использовать массивы.

Массив – это упорядоченная совокупность однотипных данных, с каждым из которых связан упорядоченный набор целых чисел, называемых *индексами*. Массив характеризуется именем, размерностью и размером.

Имя массива образуется по общему правилу образования имен, т.е. представляет собой идентификатор, например `A`, `B1`, `C8` и т.д.

Работа с массивом сводится к действиям над его элементами. Для того чтобы указать, какой элемент в данный момент используется, достаточно задать его порядковый номер, который приписывается к имени соответствующего массива. Таким образом, элементы массива обозначаются переменной с индексами. *Запись переменной с индексами* состоит из имени массива и следующего за ним в квадратных скобках списка индексов, например `A[1]`, `A[I]`, `B1[K]`, `C8[I, J]`, `C8[2, 1]`.

Индексы определяют положение элемента в массиве. *Число индексов определяет размерность массива,* т.е. форму его компоновки. Одномерный массив соответствует линейной таблице. Его элемент обозначается переменной с одним индексом: `A[1]`, `A[I]` – соответственно первый и *i*-й элементы одномерного массива A.

Для записи элементов массива в память компьютера нужно выделить для их хранения необходимое количество (массив) ячеек памяти, которое определяется размером массива. *Размеры массива* задаются границами изменения индексов по каждому измерению (минимальное и максимальное значение индекса).

По умолчанию применяется так называемая нумерация с нулевой базой, т.е. элементы массива нумеруются, начиная с 0.

В программе для каждого массива должны быть указаны его параметры: *имя, размерность и размеры*. Эта информация нужна для резервирования необходимого объема памяти для хранения числовых значений; она задается специальным *оператором описания массивов*.

Описание статического массива определяет имя, размер массива и тип данных, которые в нем хранятся. Формат описания в разделе переменных:

```
Var
<имя_массива>: array <[тип_индекса]> of <тип_данных>.
```

Начиная с версии Delphi 4 можно использовать также и динамические массивы, когда количество элементов может меняться по ходу выполнения программы.

Динамические массивы отличаются от обычных статических тем, что для них не объявляется заранее длина – число элементов. Объявление такого массива содержит только имя и тип элементов.

```
Var  <имя_массива>: array of <тип_данных>.
```

При объявлении динамического массива место под него не отводится. Прежде чем использовать такой массив, надо задать в программе его размер процедурой `SetLength`. Параметры данной процедуры – количество элементов по каждой размерности.

Краткая характеристика основных компонентов Delphi

Панель **`GroupBox`** группы `Standard` – это контейнер с рамкой и надписью, объединяющий группу связанных органов управления, таких, как переключатели `RadioButton`, флажки `CheckBox` и т.д.

Свойства панели `GroupBox`:

Caption	Задает надпись для рамки, выделяющей группу объединенных компонент.

Если компоненты, размещаемые на панели, оказываются под панелью и не отображаются, то следует выделить панель и выбрать в контекстном меню команду Control → Send to Back (Порядок → На задний план).

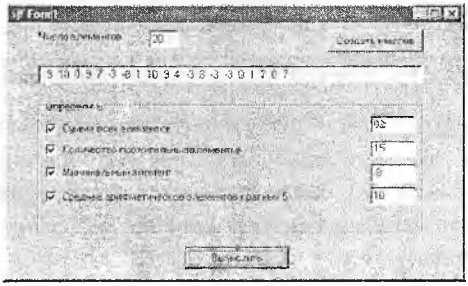
Вопросы для самоконтроля:

1. Что такое массив данных?
2. Как и в каком разделе программного кода описываются массивы?
3. Как определить местоположение элемента в массиве?
4. Что такое индекс? Каким требованиям он должен удовлетворять?
5. Как осуществляется доступ к элементам массива?
6. Что такое динамический массив?
7. Опишите отличия статических и динамических массивов. Каков порядок описания и применения динамических массивов?
8. Когда и для чего используется процедура `SetLength`?
9. Каково назначение компоненты `GroupBox`? Как задать надпись в этом компоненте?
10. Чем отличаются компоненты `RadioGroup` и `GroupBox`? Как задать список элементов в объекте `RadioGroup`?

Основная часть

Задание 1.

Сформируйте массив, произведите вычисление суммы, произведения, количества элементов, среднего арифметического элементов массива, максимального и минимального элемента массива.

`Edit`	
`Label`	
`CheckBox`	
`Button`	
`GroupBox`	

Приложение предлагает пользователю задать размер линейного массива, заполняет этот массив случайными целыми числами в диапазоне от –10 до 10, выводит список элементов массива, затем по выбору пользователя определяет: сумму всех элементов массива, количество положительных элементов массива, наименьший элемент массива и среднее арифметическое элементов массива кратных пяти.

Примечание. Размер массива N и сам массив M описаны в разделе объявления типов, констант, переменных, функций и процедур, доступном для всех модулей приложения, т.к. эти переменные будут использоваться в разных событийных процедурах: в процедуре заполнения массива случайными числами.

```
var

Form1: TForm1;
N: integer;
M: array of integer; {Описание динамического массива целых
чисел}

Implementation
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
i: integer;
begin
Randomize;
N:=StrToInt(Edit1.Text);          {Число элементов массива}
SetLength(M,N);                   {Задать массиву M длину N}
Edit2.Text:='';                   {Очистить окно Edit2}
{Заполнить массив случайными значениями целых чисел}
For i:=0 to n do
Begin
{Присвоить элементу массива случайное число из отрезка
[-10,10]}
M[i]:=Round(sin(Random(10))*10);
{Вывести элементы массива}
Edit2.Text:=Edit2.Text+'  '+IntToStr(M[i]);
end;
end;

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

```
var
i: integer;
Sum, Min, CountP, Sum1, Kol: integer;
begin
if CheckBox3.Checked then Min:=M[0]; {Пусть минимальным эле-
ментом будет первый элемент массива}
Edit3.Text:='';
Edit4.Text:='';
Edit5.Text:='';
Edit6.Text:='';
Sum:=0;
Sum1:=0;
CountP:=0;
Kol:=0;
For i:=0 to N-1 do
begin
if CheckBox1.Checked then {Определить сумму всех     элемен-
тов}
Sum:=Sum+M[i];
if CheckBox2.Checked then {Определить количество положитель-
ных чисел}
if M[i]>=0 then CountP:=CountP+1;
if CheckBox3.Checked then {Определить минимальный элемент
массива}
  if Min>M[i] then Min:=M[i];
if CheckBox4.Checked then {Определить количество и сумму
элементов массива кратных 5}
  if (M[i] mod 5 = 0) and (M[i]<>0) then
  begin
   Sum1:=Sum1+M[i];
   Kol:=Kol+1;
  end;
end;
{Вывести результаты обработки массива}
if CheckBox1.Checked then Edit3.Text:=IntToStr(Sum);
if CheckBox2.Checked then Edit4.Text:=IntToStr(CountP);
if CheckBox3.Checked then Edit5.Text:=IntToStr(Min);
if CheckBox4.Checked then Edit6.Text:=FloatToStr(Sum1/Kol);
end;
end.
```

Задание 2.

Организовать сортировку линейного массива.

Существует множество алгоритмов для сортировки массивов. Ниже рассмотрены два из них: *сортировка выбором* и *методом пузырька.*

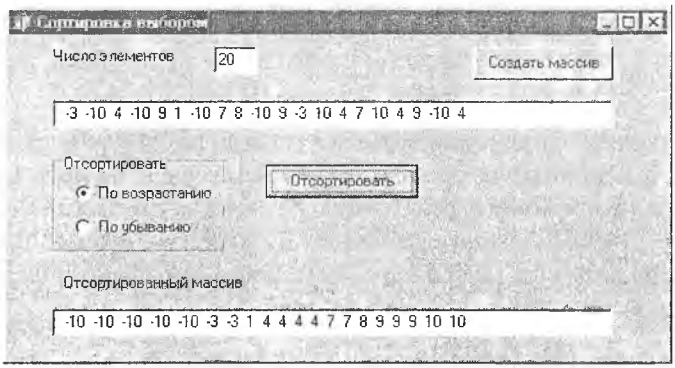
Суть метода сортировки выбором очень проста и может быть описана так:

1. В последовательности из n элементов выбирается наименьший (наибольший) элемент;

2. Меняется местом с первым;

3. Далее процесс повторяется с оставшимися n-1 элементами, затем с оставшимися n-2 элементами и т.д., до тех пор, пока не останется один самый большой (маленький) элемент.

Для реализации этого алгоритма необходимо использовать два вложенных цикла с параметром For. Внешний цикл (по i) предназначен для последовательного фиксирования элементов массива, внутренний (по j) – осуществляет поиск минимального (максимального) и его позиции в неотсортированной части массива. После выхода из внутреннего цикла следует перестановка элементов. Последний элемент во внешнем цикле не рассматривается: он сам встанет на свое место.



Программный код процедуры сортировки выбором:

```
procedure TForm1.Button2Click(Sender: TObject);
var
I,j,km,m1,p: integer;
begin
Edit3.Text:='';
```

```
For i:=0 to n-2 do
Begin
M1:=m[i];
Km:=I;
For j:=i+1 to n-1 do
begin
  If (RadioButton1.Checked) and (m[j]<m1) then {Сортировка
по убыванию}
    begin
      M1:=m[j];
      Km:=j;
    end;

  If (RadioButton2.Checked) and (m[j]>m1) then {Сортировка
по возрастанию}
    begin
     m1:=m[j];
     km:=j;
    end;
  end;
 p:=m[i]; {Перестановка элементов}
 m[i]:=m[km];
 m[km]:=p;
end;
{Вывод элементов отсортированного массива}
for i:=0 to n-1 do
Edit3.Text:=Edit3.Text+'  '+IntToStr(m[i]);
end;
```

Метод сортировки методом пузырька основан на сравнении соседних элементов. «Неправильно» расположенные по отношению друг к другу элементы меняются местами. Во вложенных циклах поочередно фиксируется пара соседних элементов массива. В результате первого прохода элемент с минимальным значением оказывается в первой позиции массива (всплывает).

Фрагмент программного кода сортировки методом пузырька:

```
For i:=0 to n-2 do
       For j:=n-1 Downto i+1 do
       If a[j-1]<a[j] then
       Begin
         p:=a[j-1];
         a[j-1]:=a[j];
         a[j]:=p;
   End;
```

Задание 3.

Удалить из сформированного массива числа, кратные трем. При этом организовать уплотнение массива.

Уплотнение массива – это удаление из него элементов, отвечающих тем или иным условиям. Образующиеся пустоты заполняются за счет сдвига всех оставшихся элементов. Так как массив укорачивается, при обработке массива необходимо использовать не цикл с параметром, а цикл с условием.

Фрагмент программного кода уплотнения массива:

Обратите внимание: на место удаленного *i*-го элемента переписывается *i*+1-ый элемент, на место *i*+1-го элемента переписывается *i*+2-ой элемент и т.д.

```
i:=0;
while i<=N do {Цикл последовательного перебора имеющихся в
массиве элементов. Изначально их N штук}
begin
if (a[i] mod 3 <> 0) or (a[i]=0) {Проверка кратности 3}
then i:=i+1 {Увеличение параметра цикла, если элемент не
кратен 3}
else
begin
{Цикл для удаления из массива элемента кратного 3,
т.е. на его место записывается значение элемента
С индексом i+1, на его место следующего и т.д.}
for j:=I to N-1 do
a[j]:=a[j+1];
{Уменьшение на единицу количества элементов массива}
N:=N-1;
end;
end; {Конец цикла While}
```

Задание 4.

В заданный упорядоченный по возрастанию массив вставить заданное число, не нарушая его упорядоченности. Последний элемент вытеснить.

Вставка элемента в массив – задача обратная предыдущей. Прием используется тот же – смещение группы элементов на одну позицию. Только при уплотнении сдвиг производится влево, при вставке – вправо. При вставке возникает проблема: что делать с последними элементами? Если в дальнейшей работе с массивом участвуют только заявленные элементы, то «хвост» придется вытеснить, последние значения при этом будут утрачены. Иначе, нужно создавать дополнительный массив, размерность которого будет больше исходного на количество вставленных элементов.

Фрагмент программного кода:

```
{Цикл для перебора элементов массива – поиск места для
вставки нового значения k}
for i:=0 to n-1 do
begin
if k<a[i] then
begin
{Место найдено, цикл для смещения элементов на единицу
вправо}
for j:=n-1 downto i+1 do
a[j]:=a[j-1];
f[i]:=k;  {Вставка нового значения на освобожденное место}
break;
end;  {if}
end;  {for}
```

Задание 5.

В одномерном массиве, состоящем из n-элементов, изменить порядок следования значений элементов на обратный от позиции n1 до позиции *n*2 (*n*1<*n*2<*n*).

При решении задач такого типа очень важен контроль за границами диапазона изменения индексов: они должны быть целыми, не выходить за пределы диапазона, кроме того, нижняя граница диапазона должна быть меньше верхней.

Фрагмент программного кода:

```
{Индексация массива начинается с 0!!}
n1:=n1-1;
n2:=n2-1;
```

```
{Цикл по изменению порядка следования элементов}
for i:=n1 to (n1+n2) div 2 do
begin
  p:=a[i];    {Поменять местами}
  a[i]:=a[n1+n2-i];
  a[n1+n2-i]:=p;
end;
```

Задание 6.

В одномерном массиве, состоящем из n элементов, произвести кольцевой сдвиг элементов на *k* позиций. Значение *k* задается, оно может быть как положительным, так и отрицательным, но целым и лежать в диапазоне: $(n\text{-}1) < k < n\text{-}1$.

Кольцевой сдвиг – это смещение элементов массива вправо либо влево, причем вытесненные элементы занимают освободившиеся в результате смещения позиции в противоположном конце массива – так, словно массив представляет собой кольцо (первый и последний элементы смыкаются). Порядок следования элементов при этом сохраняется.

Фрагмент программного кода:

```
{Анализ числа k, сдвиг будет происходить только если k от-
лично от 0}
if k<>0 then
begin
if k>0 then sdvig:=k {Формирование переменной sdvig}
else sdvig:=n+k;     {Дополнение до положительного значения}

for i:=1 to sdvig do  {Кольцевое смещение осуществляется на
1 позицию sdvig раз}
begin
tmp:=a[n-1];
{Смещение начинается с последнего элемента, который помещают
во вспомогательную переменную tmp, чтобы его значение не по-
терялось, откуда извлекают и отправляют в первую позицию по-
сле выхода из внутреннего цикла}
for j:=n-2 downto 0 do
a[j+1]:=a[j];
a[0]:=tmp;
end;   {for i}
end;   {if}
```

Задания для самостоятельного выполнения

Задание 7.

Выполните индивидуальное задание из предложенных задач первого уровня.

№	Задание
1.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–10, 10]. Найти сумму элементов, имеющих нечетное значение. 2. Вывести индексы тех элементов, значения которых больше заданного числа *A*. 3. Определить, есть ли в данном массиве положительные элементы, кратные заданному числу *K*.
2.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–15, 15]. Найти произведение элементов, имеющих четное значение. 2. Вывести индексы тех элементов, значения которых по модулю меньше заданного числа *A*. 3. Определить, есть ли в данном массиве положительные элементы, делящиеся на заданное число *k* с остатком 2.
3.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–10, 20]. Найти сумму элементов, имеющих нечетные индексы. 2. Подсчитать количество элементов массива, значения которых больше заданного числа *A* и кратных 5. 3. Найти номер первого отрицательного элемента, делящегося на 5 с остатком 2.
4.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–1000, 1000]. Найти сумму четных элементов. 2. Подсчитать количество элементов массива, значения которых состоят из двух цифр. 3. Найти номер первого положительного элемента, делящегося на 5 с остатком 2.

5.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–100, 100]. Найти сумму положительных элементов, значения которых меньше 10. 2. Вывести индексы тех элементов, значения которых кратны 3 и 5. 3. Определить, есть ли пара соседних элементов с суммой, равной заданному числу.
6.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–1000, 1000]. Найти сумму отрицательных элементов, значение которых кратно 10. 2. Вывести индексы тех элементов, значения которых кратны 5 и 10. 3. Определить, есть ли пара соседних элементов с произведением, равным заданному числу.
7.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–1000, 1000]. Найти сумму четных отрицательных элементов. 2. Вывести индексы тех элементов, значения которых кратны 3 и 6. 3. Определить, есть ли пара соседних элементов с суммой, равной заданному числу.
8.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–20, 40]. Найти удвоенную сумму положительных элементов. 2. Вывести индексы тех элементов, значения которых больше значения предыдущего элемента (начиная со второго). 3. Определить, есть ли две пары соседних элементов с одинаковыми знаками.
9.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–40, 40]. Найти сумму элементов, значения которых по модулю меньше 10. 2. Вывести индексы тех элементов, значения которых больше значения последующего элемента. 3. Определить, есть ли две пары соседних элементов с разными знаками
10.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–100, 200]. Найти сумму отрицательных элементов. 2. Найти количество тех элементов, значения которых положительны и не превосходят заданного числа A. 3. Найти номер последней пары соседних элементов с разными знаками.
11.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–100, 200]. Найти сумму четных элементов, значения которых больше заданного числа.

	2. Найти количество тех элементов, значения которых отрицательны и по модулю не превосходят заданного числа A. 3. Найти номер первой пары соседних элементов с разными знаками.
12.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–10, 20]. Найти произведение четных элементов, значения которых по модулю меньше 5. 2. Найти количество тех элементов, значения которых нечетны и по модулю превосходят заданное число A. 3. Найти номер последней пары соседних элементов, сумма которых больше заданного числа.
13.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–10, 10]. Найти сумму элементов, значения которых кратны 3 и 5. 2. Найти количество тех элементов, значения которых положительны и по модулю не превосходят заданное число A. 3. Найти номер первой пары соседних элементов, сумма которых меньше заданного числа.
14.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–100, 100]. Найти сумму элементов, значения которых состоят из одной цифры. 2. Найти количество тех элементов, значения которых положительны и кратны 3 и 5. 3. Найти номер последней пары соседних элементов с одинаковыми знаками, произведение которых меньше заданного числа.
15.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–1000, 1000]. Найти сумму положительных элементов, значения которых состоят из двух цифр. 2. Найти количество тех элементов, значения которых по модулю превосходят 100 и кратны 5 и 10. 3. Найти номер первой пары соседних элементов с разными знаками, сумма которых меньше заданного числа.

Задание 8.

Выполните индивидуальное задание из предложенных задач второго уровня.

№	Задание
1.	1. Заменить первый элемент массива, кратный 5, нулем. 2. Заменить элементы массива с нечетными номерами на квадраты их номеров. 3. Из элементов массива *D* сформировать массив *A* той же размерности

по правилу: если номер четный, то значение элемента находится по формуле $A_i = D_i^2$, а если нечетный, то по формуле $A_i = D/i$.
1. Заменить последний элемент массива, кратный 3, нулем. 2. Заменить элементы массива с четными номерами на произведение значения этого элемента и его номера. 3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то значение элемента находится по формуле $A_i = i*D_i^2$, а если нечетный, то по формуле $A_i = D/(i-1)$.
1. Заменить последний положительный элемент массива на второй элемент массива. 2. Разделить все элементы массива с четными номерами на первый элемент (первый элемент отличен от 0). 3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то значение элемента находится по формуле $A_i = D_i^2$, если нечетный, то по формуле $A_i = 2D_i$.
1. Заменить последний отрицательный элемент массива на модуль первого элемента массива. 2. Разделить все элементы массива с нечетными номерами на последний элемент (последний элемент отличен от 0). 3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то значение элемента находится по формуле $A_i = D_{i+1}^2 \cdot i$, а если нечетный, то по формуле $A_i = i*D_i$.
1. Заменить минимальный по модулю отрицательный элемент массива первым элементом. 2. Заменить последние k элементов массива на противоположные по знаку. 3. Из элементов массива C сформировать массив A той же размерности по правилу: если номер i элемента четный, то $A_i = (i-1)*C_i$, если нечетный, то $A_i = 2*i*C_i$.
1. Заменить максимальный по модулю отрицательный элемент массива нулем. 2. Заменить первые k элементов массива на противоположные по знаку. 3. Из элементов массива С сформировать массив А той же размерности по правилу: если номер i элемента четный, то $A_i = C_i^2$, если нечетный, то $A_i = 2C_i$.
1. Заменить минимальный по модулю положительный элемент массива нулем. 2. Заменить элементы массива с $k1$-го по $k2$-й на те же элементы в обратном порядке. 3. Из элементов массива D сформировать массив A той же размерности по правилу: первые 10 элементов находятся по формуле $A_i = D_i+i$, остальные по формуле $A_i = Di-i$.

8.	1. Заменить минимальный по модулю положительный элемент массива последним элементом. 2. Заменить первые *k* элементов массива на те же элементы в обратном порядке. 3. Из элементов массива *D* сформировать массив *A* той же размерности по правилу: все четные элементы находятся по формуле $A_i = D_i+i$, , а нечетные по формуле $A_i = Di-i$.
9.	1. Заменить максимальный элемент массива на противоположный по знаку. 2. Заменить нулями элементы массива между минимальным и максимальным, кроме их самих. 3. Из элементов массива *D* сформировать массив *A* той же размерности по правилу: элементы с 3-го по 12-й находятся по формуле $A_i = -D_i^2$, остальные по формуле $A_i = Di-1$.
10.	1. Заменить минимальный элемент массива на средний (количество элементов – нечетно). 2. Заменить нулями элементы массива между минимальным по модулю и максимальным по модулю, кроме их самих. 3. Из элементов массива *D* сформировать массив *A* той же размерности по правилу: элементы с первого по k-ый находятся по формуле $A_i = -D_i^2$, остальные по формуле $A_i = Di-1$.
11.	1. Заменить первый отрицательный элемент массива нулем. 2. Умножить все элементы массива, кратные 3, на третий элемент массива. 3. Из элементов массива *P* сформировать массив *M* той же размерности по правилу: если номер четный, то $M_i=i*P_i$, если нечетный, то $M_i=-P_i$.
12.	1. Заменить первый отрицательный элемент массива на первый положительный. 2. Умножить все четные положительные элементы на последний элемент массива. 3. Из элементов массива *P* сформировать массив *M* той же размерности по правилу: если элемент четный, то $M_i=i*P_i$, если нечетный, то $M_i=-P_i$
13.	1. Заменить последний отрицательный элемент массива предпоследним элементом массива. 2. Умножить все элементы массива, кратные 3, на его номер. 3. Из элементов массива *P* сформировать массив M той же размерности по правилу: каждый третий элемент по формуле $M_i=i*P_i$, а все остальные по формуле $M_i = -P_i*(1 + 1)$.

4.	1. Заменить предпоследний элемент массива на максимальный по модулю. 2. Умножить все нечетные элементы массива, кратные 3, на его номер. 3. Из элементов массива *P* сформировать массив *M* той же размерности по правилу: первый и последний элементы равны нулю, каждый четвертый элемент по формуле $M_i = 4*abs(P_i)$, а все остальные по формуле $M_i = -P_i*(i+1)$.
5.	1. Заменить второй элемент массива на максимальный среди отрицательных. 2. Заменить элементы массива между минимальным и максимальным на те же элементы в обратном порядке. 3. Из элементов массива *P* сформировать массив *M* той же размерности по правилу: первый и последний элементы равны нулю, а все остальные по формуле $M_i = -P_i * I + 1$).

Задание 9.

Выполните индивидуальное задание из предложенных задач третьего уровня.

№	Задание
1.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–40, 30]. Удалить из него все элементы, которые состоят из одинаковых цифр (включая однозначные числа). 2. Вставить число К перед всеми элементами, в которых есть цифра 1. 3. Переставить первые три и последние три элемента местами, сохраняя порядок их следования.
2.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–10, 60]. Удалить из него все элементы, в которых последняя цифра четная, а само число делится на нее. 2. Вставить элемент со значением *K* до и после всех элементов, заканчивающихся на цифру *K*. 3. Переставить элементы следующим образом: `a[1], a[12], a[2], a[11], a[3], a[10], a[4], a[9], a[5], a[8], a[6], a[7].`
3.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–35, 75]. Удалить из него все элементы, первая цифра которых четная. 2. Вставить число *K*1 после всех элементов, больших заданного числа, а число *K*2 – перед всеми элементами, кратными трем. 3. Перенести первые k элементов в конец: `a[k+1], a[k+2], .., a[n], a[2], .., a[k].`

4.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–45, 95]. Удалить из него все элементы кратные 7 и принадлежащие промежутку [*a, b*]. 2. Вставить число *K* между всеми соседними элементами, которые имеют одинаковые знаки. 3. Переставить в обратном порядке часть массива между элементами с номерами *K1* и *K2*, включая их.
5.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–20, 50]. Удалить из него все элементы, в записи которых есть цифра 5. 2. Вставить число *K* после всех элементов, кратных своему номеру. 3. Поменять местами первый положительный и последний отрицательный элементы.
6.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–40, 30]. Удалить из него все четные элементы, у которых последняя цифра 2. 2. Вставить максимальный элемент массива после всех элементами, в которых есть цифра 1. 3. Переставить первые *k* и последние *k* элемента местами, сохраняя порядок их следования.
7.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–10, 60]. Удалить из него все элементы, в которых последняя цифра нечетная, а само число кратно 3. 2. Вставить элемент со значением K после всех четных элементов, начинающихся на цифру K. 3. Переставить элементы следующим образом: `a[1], a[12], a[2], a[11], a[3], a[10],a[4], a[9], a[5], a[8], a[6], a[7].`
8.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–135, 175]. Удалить из него все элементы, первая и последняя цифра которых четная. 2. Вставить число *K1* после всех элементов, больших заданного числа, а число *K2* – после всех элементов, кратных пяти. 3. Перенести первые k элементов в конец: `a[k+1], a[k+2], .., a[n], a[2], .., a[k].`
9.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–95, 95]. Удалить из него все отрицательные элементы кратные 5 и принадлежащие промежутку [a,b].

	2. Вставить число *K* между всеми соседними элементами, которые имеют разные знаки. 3. Переставить в обратном порядке часть массива между элементами с номерами *K1 и K2*, включая их.
10.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–50, 50]. Удалить из него все элементы, в записи которых последняя цифра равна 0. 2. Вставить значение минимального элемента массива после всех четных элементов. 3. Поменять местами две половины массива, сохраняя их порядок.
11.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–140, 140]. Удалить из него все элементы, у которых первая и вторая цифры одинаковые. 2. Вставить число *K* после всех элементами, в записи которых есть цифра 0. 3. Переставить первые два и средние два элемента местами, сохраняя порядок их следования (количество элементов – четное).
12.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–100, 100]. Удалить из него все максимальные элементы. 2. Вставить максимальное значение элементов массива перед всеми элементами, в записи которых есть цифра 1. 3. Переставить последние три и средние три элемента местами, сохраняя порядок их следования (количество элементов – нечетное).
13.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–200, 500]. Удалить из него все элементы, в записи которых есть цифра 0. 2. Вставить число *K* после всех четных элементов. 3. Поменять местами три первых положительных элемента с тремя первыми отрицательными элементами, сохраняя порядок их следования.
14.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–60, 60]. Удалить из него все элементы, в которых последняя цифра нечетная. 2. Вставить элемент со значением *K* после всех нечетных элементов, начинающихся на цифру *K*. 3. Поменять местами три последних отрицательных элемента с тремя первыми отрицательными элементами, сохраняя порядок их следования.

15.	1. Дан массив целых чисел из *n* элементов, заполненный случайным образом числами из промежутка [–600, 600]. Удалить из него все элементы, в которых последняя цифра ноль. 2. Вставить элемент со значением *K* после всех четных элементов, оканчивающихся на ноль. 3. Поменять местами три средних элемента с тремя последними отрицательными элементами, сохраняя порядок их следования.

Лист самооценки выполнения лабораторной работы № 4.

Каждый пункт оценивается в 1 балл, если с уверенностью отвечаете «да». Максимальное количество баллов – 5.

1) Самостоятельно ответили на все вопросы из раздела «Вопросы для самоконтроля».
2) Самостоятельно выполнили задания № 1–6.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 5

Многомерные массивы в Delphi

8 часов

Вы научитесь:

- использовать компонент `GroupBox` для двумерного массива;
- реализовывать ввод, редактирование, поиск и сортировку данных через компонент `GroupBox`;
- использовать данный компонент при решении задач с использованием двумерных массивов в среде Delphi.

Теоретическая часть

Вспомните, что называют многомерным массивом. Приведите примеры, когда необходимо использовать двумерные массивы.

Доступ к элементам двухмерного массива осуществляется с помощью двух индексов. Первый индекс отвечает за строку, второй – за столбец: a[1,3],a[0,4] и т.д.

Формат описания двумерного массива в разделе объявления переменных:

```
  Var
<имя_массива>:array<[тип_индекса,тип_индекса]>of
  <тип_данных>
```

Заполните таблицу «Знаю. Узнал. Хочу узнать» по теме «Массивы».

Знаю	*Узнал*	*Хочу узнать*

Краткая характеристика компонента StringGrid

Компонент `StringGrid` группы `Additional` представляет собой таблицу, содержащую строки. Таблица может иметь полосы прокрутки, причем, заданное число первых строк и столбцов может быть фиксированным и не подвергаться прокрутке. Таким образом можно задать заго-

ловки столбцов и строк, постоянно присутствующих в окне компонента. Каждой ячейке таблицы может быть поставлен в соответствии некоторый объект.

Свойства компонента `StringGrid`:

`Cells`	В этом свойстве хранятся все элементы таблицы. Имеет тип `String`.
`FixedCols`	Задает фиксированное количество столбцов в таблице.
`FixedRows`	Задает фиксированное количество строк в таблице.
`Options → GoEditing`	По умолчанию данные в таблицу вводить нельзя. Чтобы снять это запрет в этом свойстве надо задать `True`.
`ColCount`	Задает общее количество столбцов таблицы.
`RowCount`	Задает общее количество строк таблицы.

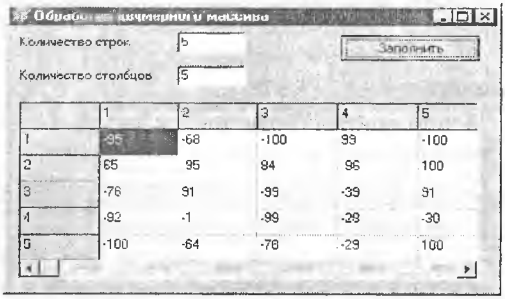
Вопросы для самоконтроля:

1. Что такое массив данных?
2. Как и в каком разделе программного кода описываются массивы?
3. Что такое индекс? Каким требованиям он должен удовлетворять?
4. Что такое динамический массив? Опишите отличия статических и динамических массивов.
5. Когда и для чего используется процедура `SetLength`?
6. Какие компоненты удобно применять при работе с одномерными, двухмерными массивами?
7. Что следует сделать, чтобы в компоненте `StringGrid` можно было вводить данные?
8. Как используется свойство `Cells` компоненты `StringGrid`?
9. Для чего используются свойства `FixedRows` и `FixedCols` в `StringGrid`?
10. Каково назначение компоненты `GroupBox`? Как задать надпись в этом компоненте?

Основная часть

Задание 1.

Организовать вывод элементов двумерного массива.

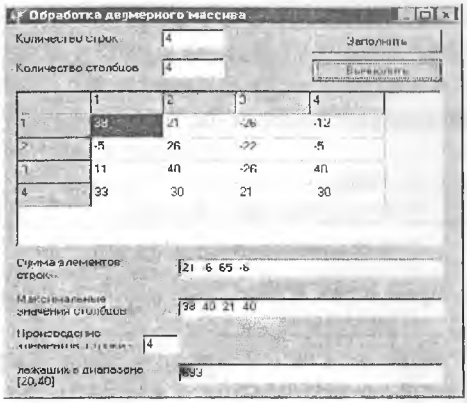



```
procedure TForm1.Button1Click(Sender: TObject);
var
i,j,n,m: integer;
a: array[0..30,0..30] of integer; {Описание массива}
begin
Randomize; {Инициализация датчика случайных чисел}
n:=StrToInt(Edit1.Text);   {Количество строк}
m:=StrToInt(Edit2.Text);   {Количество столбцов}
For i:=0 to n-1 do
For j:=0 to m-1 do
A[i,j]:=Round(Sin(Random(100))*100); {Заполнение массива
случайными числами из диапазона [-100,100]}
StringGrid1.RowCount:=n+1; {Количество строк в заголовке
таблицы}
StringGrid1.ColCount:=m+1; {Количество столбцов в заголовке
таблицы}

{Формирование заголовков строк и столбцов}
with StringGrid1 do
{Оператор with в данном случае позволяет не использовать имя
объекта при обращении к свойствам этого объекта}
begin
i:=0; {Столбец 0}
for j:=1 to RowCount do {Вывод номеров строк}
cells[i,j]:=inttostr(j);

j:=0; {Строка 0}
for i:=1 to ColCount do {Вывод номеров столбцов}
cells[i,j]:=IntToStr(i);
end;
{Вывод элементов массива в таблицу}
for i:=1 to n do
for j:=1 to m do
cells[j,i]:=IntToStr(a[i-1,j-1]);
end;
end.
```

Задание 2.



Заполнить двухмерный массив $M * N$ случайными целыми числами из диапазона [–40, 40]. Определить:

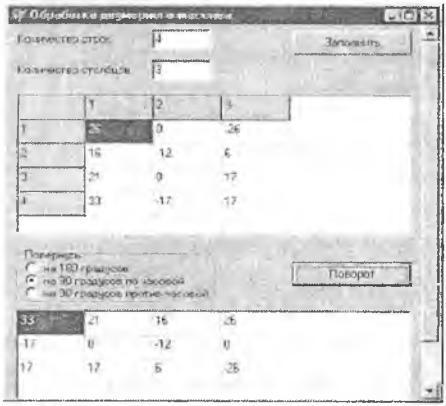
1. Сумму элементов каждой строки;
2. Максимальные значения для каждого столбца;
3. Произведение элементов k-ой строки, значения которых лежат в диапазоне от 20 до 40.

```
var
i,j,s,max,k,p: integer;
flag: boolean;
begin
{Очистка текстовых окон}
Edit4.Text:=''; Edit5.Text:=''; Edit6.Text:='';
For i:=0 to n-1 do {Вложенный цикл для нахождения сумм по
строкам}
begin
s:=0;
for j:=0 to m-1 do  s:=s+a[I,j];
Edit4.Text:= Edit4.Text+IntToStr(s)+'  ';
end;
for j:=0 to m-1 do  {Вложенный цикл для нахождения
максимумов по столбцам}
begin
max:=a[0,j];
for i:=0 to n-1 do if a[i,j] > max then max:=a[i,j];
Edit5.Text:= Edit5.Text+IntToStr(max)+'  ';
end;
k:=StrToInt(Edit3.Text)-1;
p:=1;
flag:=False;
{Цикл по столбцам для вычисления произведения чисел k-ой
строки, попавших в диапазон (20,40)}
for j:=0 to m-1 do
if (a[k,j] >20) and (a[k,j] <40)  then
begin p:=p*a[k,j]; flag:=True end;
if flag then Edit6.Text:=IntToStr(p) else Edit6.Text:='Таких
элементов нет';
```

Задание 3.

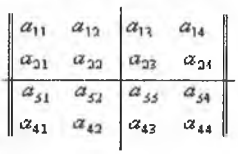
Сформировать массив из *M-x N* элементов. Получить новый массив, повернув исходный на:

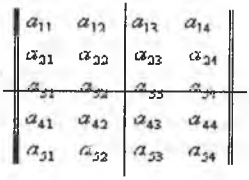
1. 1800;
2. 900 по часовой стрелке;
3. 900 против часовой стрелки.

Фрагменты кодов программы	Окно формы приложения
{Поворот на 180 градусов} `for i:=0 to n-1 do` `for j:=0 to m-1 do` `b[i,j]:=a[n-i-1,m-j-1];` *{Поворот на 90 градусов по часовой}* `for i:=0 to m-1 do` `for j:=0 to n-1 do` `b[i,j]:=a[n-j-1,i];` *{Поворот на 90 градусов против часовой}* `for i:=0 to m-1 do` `for j:=0 to n-1 do` `b[i,j]:=a[j,m-1-i];`	

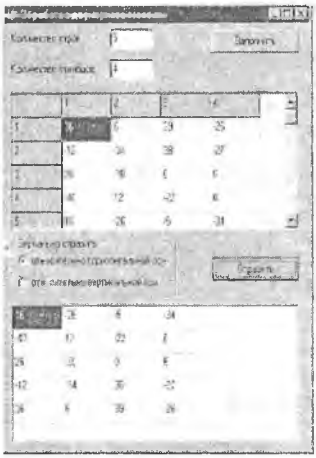
Задание 4.

Сформировать массив из *M * N* элементов. Зеркально отразить (повернуть) его относительно: 1) горизонтальной оси; 2) вертикальной оси. Дополнительных массивов не создавать. *Примечание*. Зеркальное отображение (поворот) массивов относительно горизонтальной и вертикальной осей:





Важно! В первой задаче поворота массива относительно горизонтальной оси внешний цикл по строкам организуется *только* до (n-1) div 2 (горизонтальной оси массива). Во второй задаче поворота массива относительно вертикальной оси внутренний цикл по столбцам организуется *только* до (m-1) div 2 (вертикальной оси массива).

Фрагменты кодов программы	Окно формы приложения
{Относительно горизонтальной оси} `for i:=0 to (n-1) div 2 do` `for j:=0 to m-1 do` `begin` `d:=a[i,j];` `a[i,j]:=a[n-i-1,j];` `a[n-i-1,j]:=d;` `end;` *{Относительно вертикальной оси}* `for i:=0 to n-1 do` `for j:=0 to (m-1) div 2 do` `begin` `d:=a[i,j];` `a[i,j]:=a[i, m-j-1];` `a[i, m-j-1]:=d;` `end;`	

Задание 5.

Сформировать массив *A*, содержащий *N* строк и *M* столбцов. Преобразовать его в одномерный массив *B*.

Существуют два способа решения этой задачи:

1. Задать независимый счетчик *k* для результирующего одномерного массива;

2. Вычислять значение очередного элемента массива по формуле *B(M * i+j) = A(i,j)*.

Фрагменты кодов программы:

Использование независимого счетчика *k* для формирования одномерного массива	Вычисление очередного элемента одномерного массива по формуле *B(M * i+j) = A(i,j)*
var i,j,k: integer; b: array of integer; begin {Очистка текстовых окон} Edit3.Text:=''; SetLength(b,n*m); k:=0; {Вложенный цикл для формирования элементов нового массива} For i:=0 to n-1 do for j:=0 to m-1 do begin b[k]:=a[i,j]; Edit3.Text:= Edit3.Text+ IntToStr(b[k])+' '; k:=k+1; end; end;	var i,j,k: integer; b: array of integer; begin {Очистка текстовых окон} Edit3.Text:=''; k:=n*m; SetLength(b,k); {Вложенный цикл для формирования элементов нового массива} for i:=0 to n-1 do for j:=0 to m-1 do begin b[m*i+j]:=a[i,j]; for i:=0 to k-1 do Edit3.Text:= Edit3.Text+ IntToStr(b[i])+' '; end;

Задание 6.

Заполнить двухмерный массив *N* * *N* случайными целыми числами из диапазона [–40, 40]. Определить:

1. Минимальное значение для элементов, расположенных на главной диагонали, и максимальное значение для элементов, расположенных на побочной диагонали;

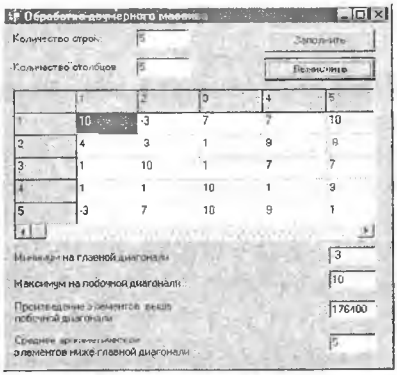
2. Произведение элементов, расположенных выше побочной диагонали;

3. Среднее арифметическое элементов, расположенных ниже главной диагонали.

Примечание. Способ отбора нужных элементов для решения такого рода задач заключается в следующем:

• для элементов, расположенных *на* главной диагонали, справедливо $i = j$ (i – индекс строки, j – индекс столбца), *ниже* $i > j$, *выше* $i < j$.

для элементов, расположенных *на* побочной диагонали, справедливо $i = N - j - 1$ (I – индекс строки, j – индекс столбца), *ниже* $i > N - j - 1$, *выше* $i < N - j - 1$ (индексы элементов начинаются с 0!).



Фрагмент программного кода:

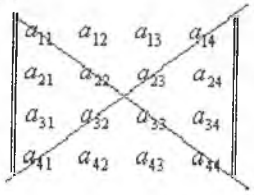
```
var
i,j,s,max,min,k,p: integer;
begin
{Очистка текстовых окон}
Edit3.Text:=''; Edit4.Text:='';
Edit5.Text:=''; Edit6.Text:='';
min:=a[0,0];
max:=a[0,n-1];
for i:=0 to n-1 do
begin
if a[i,i]<min then min:=a[i,i]; {Минимум на главной}
if a[i,n-i-1]> max then max:=a[i,n-i-1]; {Максимум на побоч-
ной}
end;
Edit3.Text:=IntToStr(min);
Edit4.Text:=IntToStr(max);
p:=1;
end;
{Вложенный цикл для нахождения произведения элементов выше
побочной}
for i:=0 to n-1 do
for j:=0 to n-1 do
if i < n-j-1 then p:=p*a[i,j];
Edit5.Text:=IntToStr(p);
s:=0; k:=0;
{Вложенный цикл для нахождения суммы и количества элементов
ниже главной}
for i:=0 to n-1 do
for j:=0 to n-1 do
if i > j then begin s:=s+a[i,j];k:=k+1; end;
Edit6.Text:=IntToStr(s/k);
end;
```

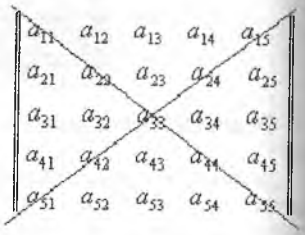
Задание 7.

Сформировать массив из $N * N$ элементов. Зеркально отразить (п вернуть) его относительно:

1. главной диагонали;
2. побочной диагонали.

Дополнительных массивов не создавать.





Важно! При решении задачи внешний и внутренний циклы органи зуются так, чтобы проход элементов осуществлялся *только* до нужной диагонали (до i-1 при повороте относительно главной диагонали и *n-i-1* при повороте относительно побочной), иначе смена произойдет дважды и все останется на своих местах.

Фрагменты кодов программы:

```
{Зеркальное отражение
относительно главной
диагонали}
for i:=1 to n-1 do
for j:=0 to i-1 do
begin
 d:=a[i,j];
 a[i,j]:=a[j,i];
 a[j,i]:=d;
end;
```

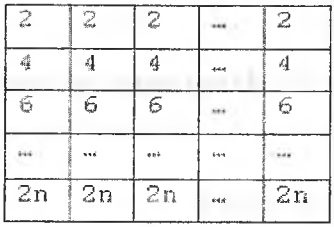
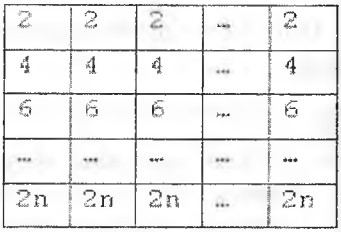
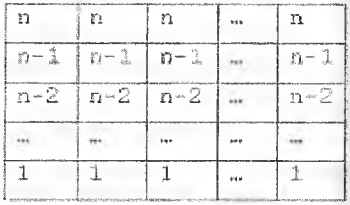
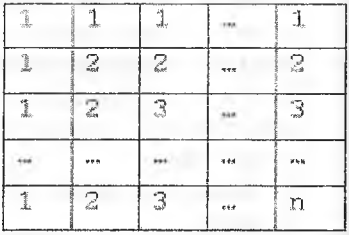
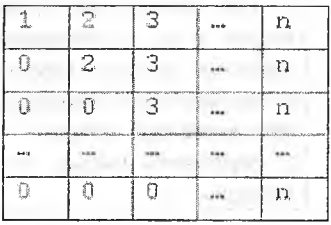
```
{Зеркальное отражение
относительно побочной
диагонали}
for i:=1 to n-2 do
for j:=0 to n-i-1 do
begin
 d:=a[i,j];
 a[i,j]:=a[n-j-1,n-i-1];
 a[n-j-1,n-i-1]:=d;
end;
```


Задания для самостоятельного выполнения

Задание 8.

Выполните индивидуальное задание из предложенных задач первого уровня.

№	Задание	
1.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить, есть ли в данном массиве строка, в которой имеются два элемента массива, имеющие наибольшие значения. 2. Заполнить массив n*n по правилу:	<table><tr><td>1</td><td>1</td><td>1</td><td>…</td><td>1</td></tr><tr><td>2</td><td>2</td><td>2</td><td>…</td><td>2</td></tr><tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr><tr><td>n</td><td>n</td><td>n</td><td>…</td><td>n</td></tr></table>
2.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить в нем разность между средним арифметическим элементов массива и средним арифметическим максимального и минимального элементов. 2. Заполнить массив n*n по правилу:	<table><tr><td>1</td><td>1</td><td>1</td><td>…</td><td>1</td></tr><tr><td>2</td><td>2</td><td>2</td><td>…</td><td>2</td></tr><tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr><tr><td>n</td><td>n</td><td>n</td><td>…</td><td>n</td></tr></table>
3.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Найти строку с минимальной суммой и в ней максимальный элемент. 2. Заполнить массив n*n по правилу:	<table><tr><td>1</td><td>2</td><td>3</td><td>…</td><td>n-1</td><td>n</td></tr><tr><td>0</td><td>1</td><td>2</td><td>…</td><td>n-2</td><td>n-1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>…</td><td>n-3</td><td>n-2</td></tr><tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr><tr><td>0</td><td>0</td><td>0</td><td>…</td><td>0</td><td>1</td></tr></table>

4.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить, есть ли в данном массиве столбец, в котором имеются одинаковые элементы. 2. Заполнить массив n*n по правилу:	
5.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить, есть ли в данном массиве строка, в которой ровно два отрицательных элемента. 2. Заполнить массив n*n по правилу:	
6.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить, есть ли в данном массиве строка, содержащая больше положительных элементов, чем отрицательных. 2. Заполнить массив n*n по правилу:	
7.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить в нем столбец с максимальной суммой и в нем минимальный по величине элемент. 2. Заполнить массив n*n по правилу:	
8.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить, есть ли в данном массиве столбец, в котором равное количество положительных и отрицательных элементов. 2. Заполнить массив n*n по правилу:	

1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить номера строк массива, содержащих только положительные элементы и найти среди них наибольший.
2. Заполнить массив n*n по правилу:

1	1	1	...	1
0	2	2	...	2
0	0	3	...	3
...	...	...	...	...
0	0	0	...	n

1. Дан двумерный массив размером n*m, заполненный случайными числами. Найти среднее арифметическое элементов, принадлежащих первой строке, последней строке, первому столбцу и последнему столбцу.
2. Заполнить массив n*n по правилу:

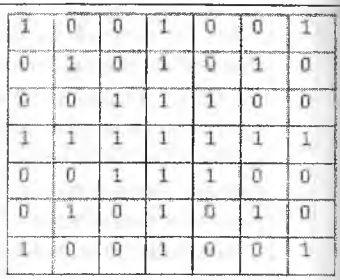
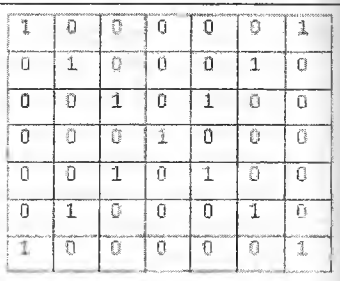
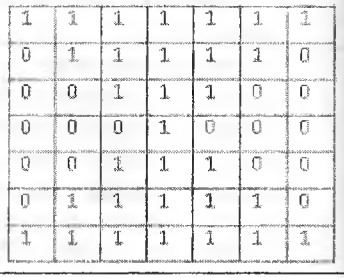
2	2	2	...	2
0	4	4	...	4
0	0	6	...	8
...	...	...	...	...
0	0	0	...	2^n

1. Дан двумерный массив размером n*m, заполненный случайными числами. Изменить массив путем деления всех его элементов на максимальный по модулю элемент.
2. Заполнить массив 6*6 по правилу:

1	2	3	4	5	6
2	3	4	5	6	1
3	4	5	6	1	2
4	5	6	1	2	3
5	6	1	2	3	4
6	1	2	3	4	5

1. Дан двумерный массив размером n*m, заполненный случайными числами. Найти сумму его элементов, расположенных между максимальным и минимальным элементами (включая оба этих числа).
2. Заполнить массив 6*6 по правилу:

1	1	1	1	1	1
1	2	3	4	5	6
1	3	6	10	15	21
1	4	10	20	35	56
1	5	15	35	70	126
1	6	21	56	126	252

13.	1. Дан двумерный массив размером n*m, заполненный случайными натуральными числами в диапазоне от 1 до 100. Определить, сколько чисел в массиве равны произведению своих индексов i*j. 2. Заполнить массив 7*7 по правилу:	
14.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Определить в нем строку с максимальной и столбец с минимальной суммой элементов. Задачу решить за один проход. 2. Заполнить массив 7*7 по правилу:	
15.	1. Дан двумерный массив размером n*m, заполненный случайными числами. Найти в каждой строке массива максимальный и минимальный элементы и поменять их с первым и последним элементом соответственно. 2. Заполнить массив 7*7 по правилу:	

Задание 9.

Выполните индивидуальное задание из предложенных задач второго уровня.

№	Задание
1.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить максимальный по модулю элемент каждой строки на противоположный по знаку. 2. Вставить после каждой четной строки первую строку. 3. Удалить все строки, содержащие ноль. 4. Поменять местами средние столбцы.

Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить минимальный по модулю элемент каждого столбца нулем. 2. Вставить после каждой строки, содержащей минимальное значение, строку из нулей. 3. Удалить все столбцы, в которых первый элемент больше последнего. 4. Поменять местами первый и последний столбцы.
Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить все элементы первых трех столбцов на их квадраты. 2. Вставить после каждой нечетной строки первую строку. 3. Удалить все столбцы, в которых первый элемент больше последнего. 4. Поменять местами средние строки с первой и последней.
Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить минимальный по модулю элемент каждого столбца на противоположный. 2. Вставить после каждого столбца, содержащего значение равное нулю, столбец из нулей. 3. Удалить все строки, содержащие максимальные элементы. 4. Поменять местами первый и последний столбцы.
Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить максимальный элемент каждой строки на противоположный по знаку. 2. Вставить после всех столбцов, содержащих максимальный элемент, столбец из нулей. 3. Удалить все столбцы, в которых есть отрицательный элемент. 4. Поменять местами первый и последний столбцы.
Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить максимальный элемент каждой строки нулем. 2. Вставить перед всеми строками, первый элемент которых делится на 3, строку из нулей. 3. Удалить самый левый столбец, в котором встретится четный отрицательный элемент. 4. Поменять местами второй и предпоследний столбцы.

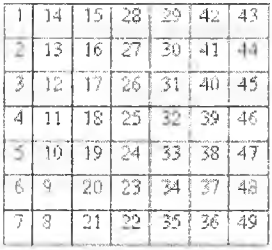
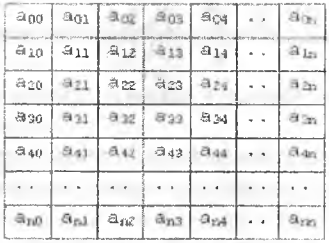
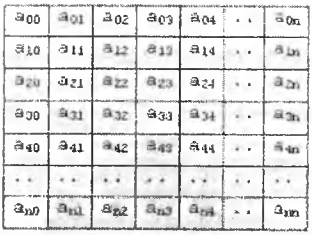
7.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить максимальный элемент каждой строки нулем. 2. Вставить после каждого столбца, содержащего максимальный элемент массива, столбец из нулей. 3. Удалить все столбцы, в которых встретится нечетный положительный элемент. 4. Поменять местами первый и предпоследний столбцы.
8.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить максимальный элемент каждого столбца нулем. 2. Вставить после всех строк, содержащих максимальный по модулю элемент, первую строку. 3. Удалить из него строку и столбец, на перекрестье которых находится максимальный по модулю элемент. 4. Поменять местами последний и предпоследний столбцы.
9.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить нулевой элемент каждого столбца максимальным по модулю элементом массива. 2. Вставить после каждой строки, содержащей максимальный по модулю элемент, последнюю строку. 3. Удалить из него каждую строку, содержащую нулевой элемент. 4. Поменять местами два средних столбца.
10.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить отрицательный элемент каждого столбца нулем. 2. Вставить после каждого столбца, содержащего максимальный по модулю элемент, строку из нулей. 3. Удалить из него каждую строку, содержащую элемент, кратный трем. 4. Поменять местами первый и последний столбцы.
11.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить элемент, кратный трем каждого столбца, нулем. 2. Вставить после каждого столбца, начиная со второго первый столбец. 3. Удалить из него каждый столбец, содержащий элемент, кратный пяти. 4. Поменять местами третий и последний столбцы.

12.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить четный элемент каждого столбца максимальным по модулю. 2. Вставить после столбцов, содержащих минимальное значение второй столбец. 3. Удалить все строки, в которых второй элемент больше предпоследнего. 4. Поменять местами первый и средний столбцы.
13.	Дан двумерный массив размером n*m, заполненный случайным образом. 5. Заменить нечетный элемент каждой строки нулем. Вставить после всех строк, содержащих минимальное значение, строку 1, 2, 3,… 7. Удалить все столбцы, в которых первый элемент четный. 8. Поменять местами первый и последний столбцы.
14.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить максимальный элемент каждой строки номером столбца, в которой он находится. 2. Вставить после всех столбцов, содержащих нулевой элемент, первый столбец. 3. Удалить все строки, в которых встретится четный отрицательный элемент. 4. Поменять местами первый и предпоследний столбцы.
15.	Дан двумерный массив размером n*m, заполненный случайным образом. 1. Заменить четный элемент каждой строки нулем. 2. Вставить после всех строк, содержащих минимальный элемент массива, строку 2, 4, 6,… 3. Удалить все столбцы, в которых встретится элемент по модулю больший левого верхнего ($a_{1,1}$). 4. Поменять местами третий и последний столбцы.

Задание 10.

Выполните индивидуальное задание из предложенных задач третьего уровня.

№	Задание
1.	1. Определить, является ли целочисленная матрица симметричной относительно главной диагонали. При отсутствии симметрии заменить элементы на минимальный из них.

	2. Сформировать матрицу 7x7 элементов, заполненную согласно схеме. Вывести ее на экран, развернув на 90 градусов по часовой стрелке. 
2.	1. Определить является ли целочисленная матрица симметричной относительно побочной диагонали. При отсутствии симметрии заменить элементы на минимальный из них. 2. Сформировать квадратную матрицу (N≥4). Найти сумму отмеченных элементов. 
3.	1. Определить, является ли целочисленная матрица симметричной относительно главной диагонали. При отсутствии симметрии заменить элементы нулями. 2. Сформировать квадратную матрицу (N≥4). Найти произведение отмеченных элементов. 

4. 1. Подсчитать количество одинаковых чисел над главной диагональю матрицы.
2. Сформировать квадратную матрицу (N≥4). Найти среднее арифметическое отмеченных элементов.

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	. .	a_{0n}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	. .	a_{1n}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	. .	a_{2n}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	. .	a_{3n}
a_{40}	a_{41}	a_{42}	a_{43}	a_{44}	. .	a_{4n}
. .	. .	. .	. .	. .	. .	. .
a_{n0}	a_{n1}	a_{n2}	a_{n3}	a_{n4}	. .	a_{nn}

5. 1. Подсчитать количество разных чисел над главной диагональю матрицы.
2. Сформировать квадратную матрицу (N≥4). Найти сумму четных элементов из отмеченных.

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	. .	a_{0n}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	. .	a_{1n}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	. .	a_{2n}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	. .	a_{3n}
a_{40}	a_{41}	a_{42}	a_{43}	a_{44}	. .	a_{4n}
. .	. .	. .	. .	. .	. .	. .
a_{n0}	a_{n1}	a_{n2}	a_{n3}	a_{n4}	. .	a_{nn}

6. 1. Подсчитать количество одинаковых чисел под главной диагональю матрицы.
2. Сформировать квадратную матрицу (N>4). Найти максимальный из отмеченных элементов.

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	. .	a_{0n}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	. .	a_{1n}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	. .	a_{2n}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	. .	a_{3n}
a_{40}	a_{41}	a_{42}	a_{43}	a_{44}	. .	a_{4n}
. .	. .	. .	. .	. .	. .	. .
a_{n0}	a_{n1}	a_{n2}	a_{n3}	a_{n4}	. .	a_{nn}

7.	1. Подсчитать количество различных чисел над побочной диагональю матрицы. 2. Расположить столбцы матрицы D[M,N] в порядке возрастания элементов k–ой строки ($1 \le k \le M$).
8.	1. Подсчитать количество одинаковых чисел под побочной диагональю матрицы. 2. Сформировать массив Z(N,N). Повернуть его на 270 градусов.
9.	1. Определить, является ли заданная матрица симметричной относительно главной диагонали. При отсутствии симметрии вывести месторасположения несимметричных элементов. 2. Сформировать двухмерный массив. Преобразовать его в одномерный, упорядочить одномерный массив по убыванию и снова преобразовать его в двухмерный.
10	1. Определить, является ли заданная матрица симметричной относительно побочной диагонали. При отсутствии симметрии вывести месторасположения несимметричных элементов. 2. Сформировать одномерный массив из целых чисел x_1, x_2, .., x_{n-1}, x_n. Получить квадратную матрицу порядка n. Вывести ее на экран, зеркально отразив относительно горизонтальной оси. $\begin{array}{\|c\|c\|c\|c\|c\|} \hline x_1 & x_2 & .. & x_{n-1} & x_n \\ \hline x_1^2 & x_2^2 & .. & x_{n-1}^2 & x_n^2 \\ \hline .. & .. & .. & .. & .. \\ \hline x_1^n & x_2^n & .. & x_{n-1}^n & x_n^n \\ \hline \end{array}$
11.	1. Определить, является ли заданная матрица симметричной относительно горизонтальной оси. При отсутствии симметрии вывести месторасположения несимметричных элементов. 2. Сформировать матрицу размером N x M. Переставляя ее строки и столбцы, добиться того, чтобы наибольший элемент (или один из них) оказался в верхнем левом углу.
12.	1. Определить, является ли заданная матрица симметричной относительно вертикальной оси. При отсутствии симметрии вывести месторасположения несимметричных элементов. 2. Дана квадратная матрица, сформировать одномерный массив из ее диагональных элементов. Найти след матрицы, суммируя элементы одномерного массива. Преобразовать исходную матрицу по правилу: четные строки разделить на полученное значение, нечетные оставить без изменения.

1. Определить, является ли целочисленная матрица симметричной относительно горизонтальной оси. При отсутствии симметрии заменить элементы на минимальный из них. 2. Квадратная матрица, симметричная относительно главной диагонали, задана верхним треугольником в виде одномерного массива. Восстановить исходную матрицу и вывести ее по строкам.
1. Определить, является ли целочисленная матрица симметричной относительно вертикальной оси. При отсутствии симметрии заменить элементы на минимальный из них. 2. Задана матрица порядка n и число k. Разделить элементы k-строки на диагональный элемент, расположенный в этой строке.
1. Определить, является ли заданная целочисленная квадратная матрица магическим квадратом (магический квадрат – это квадратная матрица, у которой суммы чисел в каждой строке, в каждом столбце и диагоналях равны между собой). 2. Дана квадратная матрица порядка $2n$. Получить новую матрицу, переставляя ее блоки размера $n \times n$ крест-накрест.

Лист самооценки выполнения лабораторной работы № 5.

Каждый пункт оценивается в 2 балла, если с уверенностью отвечаете «да». Максимальное количество баллов – 10.

1) Самостоятельно ответили на все вопросы из раздела «Вопросы для самоконтроля».
2) Самостоятельно выполнили задания № 1–8.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 6

Работа со строками в Delphi

4 часа

Вы научитесь:

- использовать функции Delphi для работы с данными строкового типа;
- использовать процедуры Delphi для работы с данными строкового типа;
- решать задачи с использованием функции Delphi для работы с данными строкового типа.

Теоретическая часть

В реальных задачах часто встречаются объекты символьного типа – строки.

Вспомните, каким образом трактуется строковая переменная в языке программирования Паскаль.

В состав строки могут входить буквы латинского алфавита, кириллица, цифры, всевозможные знаки, скобки, пробел и др. Каждый символ строковой величины занимает 1 байт памяти (десятичный код от 0 до 255, зафиксированный в кодовой таблице ASCII).

Фрагмент таблицы символьной кодировки ASCII:

32 -	48 - 0	64 - @	80 - P	96 - `	112 - p
33 - !	49 - 1	65 - A	81 - Q	97 - a	113 - q
34 - "	50 - 2	66 - B	82 - R	98 - b	114 - r
35 - #	51 - 3	67 - C	83 - S	99 - c	115 - s
36 - $	52 - 4	68 - D	84 - T	100 - d	116 - t
37 - %	53 - 5	69 - E	85 - U	101 - e	117 - u
38 - &	54 - 6	70 - F	86 - V	102 - f	118 - v
39 - '	55 - 7	71 - G	87 - W	103 - g	119 - w
40 - (	56 - 8	72 - H	88 - X	104 - h	120 - x
41 -)	57 - 9	73 - I	89 - Y	105 - i	121 - y
42 - *	58 - :	74 - J	90 - Z	106 - j	122 - z
43 +	59 ;	75 K	91 [	107 k	123 {
44 ,	60 <	76 L	92 \	108 l	124 |
45	61 =	77 M	93]	109 m	125 }
46 .	62 >	78 N	94 ^	110 n	126 ~
47 /	63 ?	79 O	95 _	111 o	127 □

Количество символов в строке называется ее *длиной*. Длина строки может динамически изменяться от 0 до 255. Пустая строка имеет нулевую длину.

Строковая переменная описывается в разделе описания переменных:

`Var <имя> : string [<максимальная длина строки>]`. Если максимальная длина не указана, то она принимается равной 255 (по умолчанию).

Для хранения и обработки *отдельных* символов используют переменные типа `char`. Значением переменной такого типа может быть любой один символ.

Выражения, в которых операндами служат строковые данные, называются *строковыми*. Над строковыми данными допустимы операции сцепления и операции отношения.

Операция сцепления (конкатенации) (+) применяется для соединения нескольких строк в одну результирующую строку. Сцеплять можно как строковые константы, так и строковые переменные.

Операции отношения `=, <, >, <=, >=, <>` позволяют произвести сравнения двух строк, в результате чего получается логическое значение (True или False). Операции отношения имеют более низкий приоритет, чем операции сцепления. Сравнение строк производится слева направо до первого несовпадающего символа, и та строка считается больше, в которой первый несовпадающий символ имеет больший номер в таблице символьной кодировки. Если строки имеют различную длину, но в общей части символы совпадают, считается, что более короткая строка меньше, чем более длинная. Строки равны, если они полностью совпадают по длине и содержат одни и те же символы.

Все остальные действия над строками и символами в Delphi реализуются с помощью встроенных процедур и функций.

Функции для работы с данными строкового типа:

Обращение к функции	Действие	Пример
`Copy (S, Poz, N)`	Выделяет из строки `S` подстроку длиной `N` символов, начиная с позиции `Poz`. Здесь `N` и `Poz` – целочисленные выражения.	`S:='IBM-PC';` `S1:=Copy(S,5,2);` Результат: `S1='PC'`
`Concat (S1,…,Sn)`	Выполняет сцепление (конкатенацию) строк `S1,S2,..,Sn` в одну строку.	`S1:='Test';` `S2:='-';` `S3:='5';` `S:=Concat(S1,S2,S3);` Результат: `S:='Test-5'`

Length(S)	Определяет длину строки S. Результат – значение целого типа.	S:='Test-5'; n:=Length(S); Результат: n=5
LowerCase(S)	Возвращает копию строки S, в которой все символы преобразованы в символы нижнего регистра. Работает только с буквами латинского алфавита.	S:= 'STUDENT'; S:=LowerCase(S); Результат:S='student'
UpperCase(S)	Возвращает копию строки S, в которой все символы преобразованы в символы верхнего регистра. Работает только с буквами латинского алфавита.	S:= 'student'; S:=UpperCase(S); Результат:S='STUDENT'
Trim(S)	Удаляет из строки S лидирующие и завершающие пробелы, а также управляющие символы.	S:=' Студент'; S:=Trim(S); Результат: S='Студент'
TrimLeft(S)	Возвращает копию строки S с удаленными лидирующими пробелами и управляющими символами.	S:=' Студент'; S:=TrimLeft(S); Результат: S='Студент '
TrimRight(S)	Возвращает копию строки S с удаленными завершающими пробелами и управляющими символами.	S:=' Студент'; S:=TrimRight(S); Результат: S=' Студент'
Chr(X)	Возвращает символ, указанный его ASCII-кодом. Здесь X – значение целого типа. Результат –тип Char.	for i:=65 to 71 do S:=S+Chr(i); Результат: S='ABCDEFG '
Ord(S)	Возвращает ASCII-код указанного символа. Здесь S – переменная типа Char. Результат – целый тип.	S:='F'; Cod:=Ord(S); Результат: Cod=70
Pos(S1, S2)	Обнаруживает первое появление в строке S2 подстроки S1. Результат – целое число, равное номеру позиции, где находится первый символ подстроки S1. Если такое появление не обнаружено, то результат равен 0.	S:='abcdef'; n:=Pos('cd',S); Результат: n=3

Процедуры для работы с данными строкового типа:

Обращение к процедуре	Действие	Пример
`Delete(S, Poz, N);`	Удаление `N` символов из строки `S`, начиная с позиции `Poz`.	`S:='abcdef'; Delete(S,3,2);` Результат: `S='abef'`
`Insert(S1, S2,Poz);`	Вставка строки `S1` в строку `S2`, начиная с позиции `Poz`.	`S:='ЭВМ PC'; Insert('IBM-',S,5);` Результат: `S='ЭВМ IBM-PC'`
`Str(X[:N[:M]],S);`	Преобразовывает числовое значение `X` в строку `S`. Параметр `X` является выражением целочисленного или действительного типа, а параметры `N` и `M` – это целочисленные выражения, определяющие форматирование строки `S`.	`x:= -23.60; Str(x:6:1,S);` Результат: `S='-23.6'` `a:=-23.60; b:=20.5; Str(a+b:6:1,S);` Результат: `S='-3.1'`
`Val(S,x, Cod);`	Преобразовывает строку `S` в числовое значение. Параметр `S` должен содержать последовательность символов, которая может быть воспринята как действительное число со знаком. Параметр `x` может быть как действительным, так и целочисленным. В параметр `Cod` заносится код выполнения преобразования: 0, если преобразование выполнено успешно, или номер позиции, в которой произошла ошибка.	`S:='-23.60'; val(S,x,Cod);` Результат: `x=-23,6`

Вопросы для самоконтроля:

1. Как описываются переменные символьного типа?
2. Опишите стандартные функции для работы с символьными переменными.
3. Как описываются строковые переменные?
4. Как записывается строковая константа?
5. Какова максимальная длина строки?
6. Что общего между строкой и символьным массивом?
7. Какие операции применимы к строковым переменным и константам?
8. Опишите с примерами стандартные функции обработки строковых данных.
9. Опишите работу стандартных процедур обработки строковых данных.
10. Как можно обратиться к элементам строки?

Основная часть

Задание 1.

Ввести строку символов. Определить:

1. Сколько цифр в строке.
2. Есть ли в заданной строке одинаковые символы.
3. Составить перечень всех гласных латинских букв.

Программный код:

```
procedure TForm1.Button1Click(Sender: TObject);
var
S,S1: string;
i: integer;
begin
S:=Edit1.Text;       {Введенная строка}
Edit2.Text:='';
{Цикл по всем символам строки для поиска цифр}
```

```
for i:=1 to length(S) do
if(S[i]>='0') and (S[i]<='9') then
Edit2.Text:= Edit2.Text + S[i]+' ';
Edit4.Text:='Нет';
{Цикл по всем символам строки для поиска одинаковых
символов}
for i:=1 to length(S) do
if Pos(Copy(S,i,1), Copy(S,i+1, length(S)-i)) <>0 then
begin
Edit4.Text:='Да';
Break; {Немедленный выход из цикла, если найден хотя бы один
одинаковый символ}
end;
Edit3.Text:='';
{Цикл по всем символам строки для поиска гласных латинских
букв}
for i:=1 to length(S) do
case S[i] of
'A','E', 'I','O','U','Y': Edit3.Text:= Edit3.Text+S[i]+' ';
'a','e', 'i','o','u','y': Edit3.Text:= Edit3.Text+S[i]+' ';
end;
if Edit3.Text='' then Edit3.Text:='Латинских гласных нет';
end;
end.
```

Задание 2.

Заменить во введенной строке строчные буквы кириллицы на прописные.

Программный код:

```
procedure TForm1.Button1Click(Sender: TObject);
var
S,S1: string;
i: integer;
begin
{S - заданная строка, S1 - итоговая строка после преобразо-
вания}
S:=Edit1.Text;
S1:='';
{Цикл по всем символам строки}
for i:=1 to length(S) do
case S[i] of
{Преобразование встреченной строчной буквы в прописную и до-
```

```
бавление ее в итоговую строку}
'а'..'я': S1:=S1+chr(ord(S[i])-32);
{Добавление символа в итоговую строку без преобразования}
else S1:=S1+S[i];
end;
Edit2.Text:=S1;
end;
end.
```

Задание 3.

Ввести текст в виде строки, состоящей из слов, разделенных пробелами (одним или более). Преобразовать строку так, чтобы между словами был только один пробел. Вывести сообщение о количестве удаленных пробелов.

Фрагмент кода программы:

```
S:=Trim(Edit1.Text); {Введенная строка без лидирующих и за-
вершающих пробелов}
S1:=''; {Выходная строка}
Edit2.Text:='';
{Цикл по всем символам строки для поиска пробелов}
for i:=1 to length(S) do
if (S[i]<>' ') then S1:=S1+S[i] {Переписывание не-пробелов}
else
{Запись в выходную строку только одного пробела, независимо
от их количества}
if (S[i+1]<>' ') then S1:=S1+'';
k:=length(S)-length(S1); {Подсчет количества удаленных про-
белов}
{Вывод новой строки}
Edit2.Text:=S1;
```

Задание 4.

Дана строка длиной до 254 символов. Удалить все знаки «+» перед символами, не являющимися цифрами.

Фрагмент программного кода:

```
S:=Edit1.Text+' ';
{К исходной строке добавляется пробел - для работы с последним символом}
S1:=''; {Результирующая строка}
Edit2.Text:='';
{Цикл по всем символам строки для поиска +. Рассматривается пара символов - текущий и следующий (за исключением последнего: это тот пробел, который был добавлен к исходной строке)}
for i:=1 to length(S)-1 do
if (S[i]='+') then
begin
if (S[i+1]>='0') and (S[i+1]<='9') then
S1:=S1+S[i] {Переписывание + перед цифрой}
end
else
{Переписывание любого символа, если он не +}
S1:=S1+S[i];
{Вывод новой строки}
   Edit2.Text:=S1;
```

Задание 5.

В заданной строке заменить все слова «всегда» на «часто».

Фрагмент программного кода:

```
S:=Edit1.Text; {Исходная строка}
a:=' '+Edit2.Text; {Слово, которое меняем}
b:=' '+Edit3.Text; {Слово, на которое меняем}
n:=pos(a,s); {Номер позиции первого появления нужного слова}
while n<>0 do {Условие выполнение цикла: слово найдено, т.е. номер позиции не равен 0}
begin
delete(s,n,length(a)); {Удаление из строки найденного слова}
insert(b,s,n); {Вставка нового слова на место старого}
n:=pos(a,s); {Номер позиции следующего появления нужного слова}
end;
```

Задание 6.

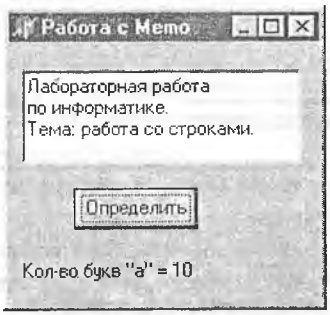
Дана строка, содержащая слова, разделенные пробелами. Преобразовать строку в массив слов (лексем).

Фрагмент программного кода:

```
s:=Edit1.Text; {Исходная строка}
Memo1.Text:='';
j:=0; {Задание индекса первого элемента массива}
{Цикл по всем символам строки для поиска очередного пробела}
for i:=0 to length(s) do
begin
if s[i]<>' ' then m[j]:=m[j]+s[i] {Заполнение символами
строки очередного элемента массива}
else
j:=j+1; {Вычисление очередного индекса массива}
end;
{Вывод полученного массива строк в объект Memo}
for i:=0 to j do
Memo1.Text:=Memo1.Text+m[i]+chr(13)+chr(10);
```

Задание 7.

Ввести с помощью компонента Memo текст из букв кириллицы, состоящий из нескольких строк. Сосчитать, сколько в этом тексте букв 'а'.



Программный код:

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

```
{В свойстве lines компонента Memo содержатся строки текста,
свойство lines.count содержит количество строк}
with Memo1 do
begin
for n:=1 to lines.count-1 do
begin
s:=lines[n];
for i:=1 to length(s) do
if copy(s,i,1)='a' then kol:=kol+1;
end;
end;
label1.caption:='Количество букв "a" = '+IntToStr(kol);
end;
end.
```

Задания для самостоятельного выполнения

Задание 8.

Выполните индивидуальное задание из предложенных задач первого уровня.

№	Задание
1.	Дана строка, заканчивающаяся точкой. Подсчитать, сколько в ней слов.
2.	Дана строка, содержащая текст на латинице. Найти количество слов, начинающихся с буквы b.
3.	Дана строка. Определить, сколько в ней символов «*», «;», «:».
4.	Дана строка символов, среди которых есть двоеточие (:). Определить, сколько символов ему предшествует.
5.	Дана строка. Преобразовать ее, удалив каждый символ «*».
6.	Дана строка. Определить, сколько раз входит в нее группа букв *abc*.
7.	Дана строка. Подсчитать количество букв k в последнем ее слове.
8.	Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобка. Вывести на экран все символы, расположенные внутри этих скобок.
9.	В строке заменить все двоеточия (:) точкой с запятой (;). Подсчитать количество замен.

10.	В строке между словами вставить вместо пробела запятую и пробел.
11.	Удалить часть символьной строки, заключенной в скобки (вместе со скобками).
12.	Определить, сколько раз в строке встречается заданное слово.
13.	Проверить, одинаковое ли число открывающихся и закрывающихся скобок в данной строке.
14.	Строка содержит произвольный текст на кириллице. Проверить, каких букв в нем больше: гласных или согласных.
15.	В строке имеется одна точка с запятой (;). Подсчитать количество символов до точки с запятой и после нее.

Задание 9.

Выполните индивидуальное задание из предложенных задач второго уровня.

№	Задание
1.	Дан текст, записанный прописными буквами кириллицы. Получить тот же текст, записанный строчными буквами.
2.	Дан произвольный текст. Выяснить, чего в нем больше: букв кириллицы или цифр.
3.	Дан текст на кириллице. Выяснить, входит ли данное слово в указанный текст, и если да, то сколько раз.
4.	Дан текст на кириллице. Определить сколько раз встречается в нем самое длинное слово.
5.	Дан текст на кириллице. Выбрать из него только те символы, которые встречаются в нем только один раз, в том порядке, в котором они встречаются в тексте.
6.	Дан текст на кириллице. Определить, сколько раз встречается в нем самое короткое слово.
7.	Дан текст на кириллице и некоторая буква. Подсчитать, сколько слов начинается с указанной буквы.
8.	Дан текст. Сколько слов в тексте? Сколько цифр в тексте?
9.	Дан текст и некоторое слово. Вывести те предложения текста, которые содержат данное слово.
10.	Дан текст, в котором встречаются арифметические выражения вида a ⊕ b, где ⊕ – один из знаков +, -, *, /. Выписать все арифметические выражения и вычислить их значения.

11.	Дан текст на кириллице. Подсчитать количество слов, начинающихся и заканчивающихся на одну и ту же букву.
12.	Дан текст на кириллице и некоторая буква. Найти слово, содержащее наибольшее количество указанных букв.
13.	Дан произвольный текст. Проверить, правильно ли в нем расставлены круглые скобки.
14.	Дан текст на кириллице и некоторые два слова. Определить, сколько раз они входят в текст и сколько раз они входят непосредственно друг за другом.
15.	Дан зашифрованный текст на кириллице. Каждая буква заменяется на следующую за ней (буква я заменяется на а). Получить новый текст, содержащий расшифровку данного текста.

Задание 10.

Выполните индивидуальное задание из предложенных задач третьего уровня.

Выполнить индивидуальное задание следующим образом:

1. Текст ввести с помощью компонента Memo.
2. Вывести слова исходного текста в алфавитном порядке.

№	Задание
1.	Дан зашифрованный текст на кириллице. Каждая буква заменяется на следующую за ней (буква *я* заменяется на *а*). Получить новый текст, содержащий расшифровку данного текста.
2.	Дан текст на кириллице и некоторые два слова. Определить, сколько раз они входят в текст и сколько раз они входят непосредственно друг за другом.
3.	Дан произвольный текст. Проверить, правильно ли в нем расставлены круглые скобки.
4.	Дан текст на кириллице и некоторая буква. Найти слово, содержащее наибольшее количество указанных букв.
5.	Дан текст на кириллице. Подсчитать количество слов, начинающихся и заканчивающихся на одну и ту же букву.
6.	Дан текст, в котором встречаются арифметические выражения вида a ⊕ b, где ⊕ – один из знаков +, -, *, /. Выписать все арифметические выражения и вычислить их значения.

7.	Дан текст и некоторое слово. Вывести те предложения текста, которые содержат данное слово.
8.	Дан текст. Сколько слов в тексте? Сколько цифр в тексте?
9.	Дан текст на кириллице и некоторая буква. Подсчитать, сколько слов начинается с указанной буквы.
10.	Дан текст на кириллице. Определить, сколько раз встречается в нем самое короткое слово.
11.	Дан текст на кириллице. Выбрать из него только те символы, которые встречаются в нем только один раз, в том порядке, в котором они встречаются в тексте.
12.	Дан текст на кириллице. Определить, сколько раз встречается в нем самое длинное слово.
13.	Дан текст на кириллице. Выяснить, входит ли данное слово в указанный текст, и если да, то сколько раз.
14.	Дан произвольный текст. Выяснить, чего в нем больше: русских букв или цифр.
15.	Дан текст, записанный прописными буквами кириллицы. Получить тот же текст, записанный строчными буквами.

Лист самооценки выполнения лабораторной работы № 6.

Каждый пункт оценивается в 1 балл, если с уверенностью отвечаете «да». Максимальное количество баллов – 5.

1) Самостоятельно ответили на все вопросы.
2) Самостоятельно выполнили задания № 1–7.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 7

Работа с файлами в Delphi

4 часа

Вы научитесь:

- работать с текстовыми файлами в среде Delphi;
- использовать процедуры Delphi для открытия и закрытия текстовых файлов;
- использовать процедуры Delphi для чтения и записи текстовых файлов;
- использовать компоненты группы Dialogs для решения задач с использованием файлов.

Теоретическая часть

Часть 1. Работа с текстовыми файлами.

По способу доступа к информации, записанной в файл, различают файлы прямого и *последовательного доступа*. Файлом последовательного доступа называется файл, к элементам которого обеспечивается доступ в такой же последовательности, в какой они записывались. Как правило – это текстовые файлы. Файлом *прямого доступа* называется файл, доступ к элементам которого осуществляется по адресу элемента. Как правило – это файлы баз данных.

Turbo Pascal поддерживает три типа файлов: текстовые, типизированные, нетипизированные.

К *типизированным* файлам относятся файлы строго определенного типа. Чаще всего это файлы, состоящие из записей. Они применяются для создания различных баз данных. Содержимое такого файла рассматривается как последовательность записей определенной структуры. Это файлы прямого доступа. Единицей измерения такого набора данных является сама запись.

Нетипизированные файлы не имеют строго определенного типа, их можно рассматривать как совокупность символов или байтов. Внутренняя реализация поддержки таких файлов наиболее близка к аппаратной поддержке работы с внешними носителями. За счет этого достигается максимальная скорость доступа к наборам данных. При работе с такими файлами не тратится время на преобразование типов и поиск управляющих последовательностей, достаточно считать содержимое файла в определенную область памяти. Это файлы прямого доступа, самым важным параметром служит длина записи в байтах.

Текстовый файл можно рассматривать как последовательность символов, разбитую на строки длиной от 0 до 256 символов. Это файлы последовательного доступа. Структурной единицей текстовых файлов является строка. Данные в таких файлах хранятся в виде цепочки ASCII кодов и могут обрабатываться любым текстовым редактором. Каждая строка завершается маркером конца строки. На практике такой маркер представляет собой последовательность из двух символов: перевод строки chr(10) и возврат каретки chr(13).

Эти два символа задают стандартные действия по управлению текстовыми файлами.

Текстовые файлы описываются в разделе описания переменных:

```
Var
<файловая переменная>: TextFile;
```

Файловая переменная – это имя переменной, которое используется в программном коде для работы с файлом.

Перед тем, как записать данные в файл или прочитать данные из файла, необходимо сначала открыть этот файл. Открытие текстового файла на запись, чтение или дозапись осуществляется с помощью разных процедур. Но прежде чем их использовать, необходимо во всех случаях присвоить файлу на магнитном носителе имя, т.е. установить соответствие между *файловой переменной* в программе и *именем файла* на диске. Это делается с помощью процедуры **AssignFile**:

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

Здесь имя файла – любое выражение строкового типа, которое строится по правилам определения имен в операционной системе.

Процедуры для открытия текстовых файлов:

Обращение к процедуре	**Действие** **В таблице F – имя файловой переменной**
`Rewrite(F)`	Открывает (создает) новый файл. Имя файла предварительно определяется в процедуре `AssignFile`. Если на диске уже был файл с таким именем, то он уничтожается.
`Reset(F)`	Открывает уже существующий файл. Файл считывается последовательно. Если эта процедура применена к несуществующему файлу, то возникает ошибка ввода-вывода.
`Append(F)`	Открывает уже существующий файл для дозаписи. Запись производится в конец файла.

У текстовых файлов есть своя специфика. Специальные расширения стандартных процедур чтения (**Read**) и записи (**Write**), описанных ниже, разрешают работать со значениями несимвольного типа. Другими словами, последовательность символов автоматически преобразуется к значению того типа переменной, которая используется в файловых операциях.

Вызов `Read(F,Ww)`, где `Ww` – переменная типа `word`, осуществляет чтение из файла F последовательности цифр, которая затем интерпре-

тируется в число, значение которого и будет присвоено переменной Ww. В случае, если вместо последовательности цифр идет любая другая последовательность символов, использование такого оператора приводит к ошибке выполнения программы.

В таблице F – имя файловой переменной. V1,V2,…,Vn – переменные разных типов.

Обращение к процедуре	Действие
Read (F,V1[,V2,…,Vn]);	Считывает из дискового файла строки символов в переменные V1, V2,…,Vn.
Readln F,V1[,V2,…,Vn]);	Выполняет те же действия, что и **Read**, и дополнительно – чтение до маркера конца строки и переход к новой строке **Readln(F)** без списка переменных позволяет пропустить строку в файле и перейти на новую строку.
Write F,V1[,V2,…,Vn]);	Записывает значения переменных V1,V2,…,Vn в файл на диске.
Writeln F,V1[,V2,…,Vn]);	Выполняет те же действия, что и **Write**, но обеспечивает запись всех величин с обязательной установкой маркера конца строки в файл **Writeln(F)** без списка переменных. Записывает в файл пустую строку.

После работы с файлом его нужно обязательно закрыть, иначе информация в файле может быть потеряна. Это делается с помощью процедуры CloseFile(F).

Функции для работы с файлами:

Функция	Действие
Eoln(F)	Возвращает булевское значение True, если текущая файловая позиция находится на маркере конца строки или вызов Eof(F) вернул значение True. Во всех остальных случаях – False.
Eof(F)	Возвращает булевское значение True, если указатель конца файла находится сразу за последним компонентом, и False – в противном случае.

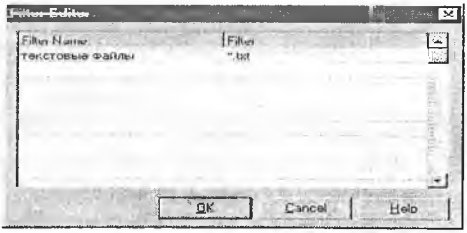
SeekEoln(F)	Возвращает булевское значение True при достижении маркера конца строки, причем указатель файла пропускает все пробелы и знаки табуляции, предшествующие маркеру. В противном случае возвращает значение False.
SeekEof(F)	Возвращает булевское значение True, если указатель файла находится на маркере конца файла. Эта функция также пропускает все пробелы и знаки табуляции, предшествующие маркеру, и выполняет автоматический пропуск маркера конца строки.

Краткая характеристика компонентов группы Dialogs

В состав среды Delphi входят десять компонентов группы Dialogs, реализующих работу со стандартными диалоговыми панелями Windows. Эти компоненты облегчают разработку стандартных фрагментов приложения, таких, как выбор нужного файла, цвета, шрифта, работу с принтером. Для определения, какая из кнопок диалоговой панели была нажата, используется метод Execute. Если нажата кнопка «ОК», то он возвращает значение True, если кнопка «Отмена», то значение False.

Компонент OpenDialog позволяет выбрать файл для открытия из стандартного диалогового окна Windows.

Свойства компонента OpenDialog:

DefaultExt	Создает расширение файла автоматически.
Filter	Позволяет задать шаблон имен открываемых файлов в окне редактора фильтров. Можно одновременно определить несколько фильтров. 
InitialDir	В этом свойстве хранится название рабочего каталога.

FileName	В этом свойстве хранится название выбранного файла.	
Options →	В этом свойстве находится ряд опций открываемой панели – свойств типа Boolean. Опция включена, если ее значение *True*, и отключена, если – *False*.	
	OfAllowMultiSelect	Позволяет выбрать несколько файлов одновременно.
	ofCreatePromt	Выдает запрос на создание файла, если его нет.
	ofFileMustExist	Позволяет выбирать только существующие файлы.
	ofNoChangeDir	Запрещает изменение каталога.
	ofOwerwritePromt	Запрашивает подтверждение на сохранение существующего файла.
	ofReadOnly	Помечает флажок Read Only (только для чтения).
	ofNoReadOnlyReturn	Запрещает выбор файла только для чтения.
Title	Задает текст заголовка панели.	

Пример использования в программе:

```
{Открыть диалог и запомнить имя файла с диска в переменной
Name}
if OpenDialog1.Execute then Name:= OpenDialog1.FileName;
```

Компонент SaveDialog позволяет выбрать файл для сохранения из стандартного диалогового окна Windows. По внешнему виду и порядку работы практически аналогичен компоненту OpenDialog. Различаются только надписи и некоторые свойства.

Компонент FontDialog позволяет вызвать стандартную диалоговую панель выбора шрифтов и их характеристик.

Свойства компонента FontDialog:

Font	Определяет, какой шрифт был выбран для последующей работы.	
Device	Выбор списка используемых шрифтов: только для принтера, только для дисплея или для того и другого устройства.	
Options	Задает опции диалоговой панели.	
	FdAnsiOnly	Работа только со шрифтами, поддерживаемыми Windows.
	fdEffects	Использование для шрифтов эффектов и цветов.
	fdFixedPitchOnly	Работа только с непропорциональными шрифтами.
	fdForceFontExist	Выдает сообщение при выборе несуществующего шрифта.
	fdTrueTypeOnly	Работа только со шрифтами TrueType.
	fdLimitSize -	Ограничивает размер выбираемого шрифта.

Пример использования в программе:

```
if FontDialog1.Execute then
Memo1.Font:= FontDialog1.Font;
```

Компонент ColorDialog  позволяет вызвать стандартную диалоговую панель настройки цветов.

Свойства компонента ColorDialog:

Color	Сохраняет выбранный цвет	
Options	Свойство состоит из следующих опций:	
	CdFullOpen	Выполнить полный показ диалога
	cdPreventFullOpen	Запретить полный показ диалога

Пример использования в программе:

```
if ColorDialog1.Execute then
Form1.Color:=ColorDialog1.Color;
```

Компонент `FindDialog`  позволяет вызвать стандартную диалоговую панель поиска текста. Позволяет ввести текст для поиска. Сам поиск программируется самостоятельно.

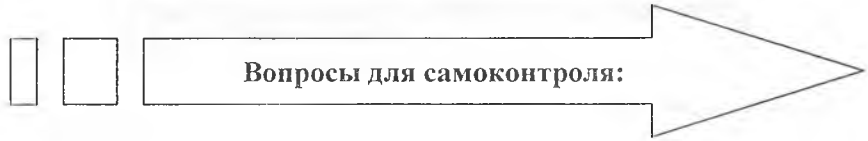
Свойства компонента `FindDialog`:

`FindText`	Хранит текст для последующего поиска.
`Options`	В этом свойстве заданы опции панели. Эти опции на включение в панель дополнительных сервисных кнопок.

Компонент `ReplaseDialog` позволяет вызвать стандартную диалоговую панель поиска и замены текста. Позволяет ввести текст для поиска. Сам поиск программируется самостоятельно.

Свойства компонента ReplaseDialog:

`FindText`	Хранит текст для поиска
`ReplaceText`	Хранит текст для замены



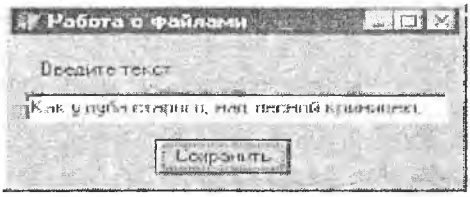
1. Как объявить файловую переменную?
2. Как файловую переменную связать с физическим файлом?
3. Опишите назначение процедур и функций файлового ввода-вывода: `Append`, `AssignFile`, `CloseFile`, `Eof`, `IOResult`, `Read`, `Reset`, `Rewrite`, `Seek`, `Write`.
4. Каково назначение директивы компилятора `{$I-}`?
5. Опишите назначение компонентов `OpenDialog`, `SaveDialog`, `FontDialog` страницы `Dialogs` палитры компонентов. Каковы особенности их размещения на форме приложения?
6. Каково назначение свойства `OpenDialog1.Filter`? Какими способами можно изменять значение этого свойства?
7. Какой метод используется для вызова диалоговой панели?
8. В каком свойстве находится имя файла?
9. Описать разницу между текстовыми, типизированными и нетипизированными файлами.

Основная часть

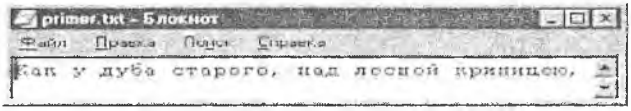
Задание 1.

Записать в файл текст, введенный в окне Edit.

Окно формы проекта:



Полученный файл в блокноте:



Программный код:

```
var
F: TextFile; {Описание файловой переменной}
begin
AssignFile(F, 'primer.txt'); {Связь файловой переменной с
файлом}
Rewrite(F);                  {Создать новый файл}
Writeln(F,Edit1.Text);       {Записать в файл}
CloseFile(F);                {Закрыть файл}
end;
```

Задание 2.

Открыть текстовый файл для чтения, считать из него текст в окно Memo. Перед открытием файла проверить его наличие, в случае отсутствия выдать сообщение.

Фрагмент кода программы:

```
var
F: TextFile; {Описание файловой переменной}
Ch: char;    {Описание переменной, в которую будет
считываться символ из файла}
begin
AssignFile(F, 'primer.txt'); {Связь файловой переменной с
файлом}
{$I-}  {Директива компилятора: отключить проверку ошибок
ввода-вывода}
Reset(F); {Открыть файл для чтения}
{$I+}  {Директива компилятора: включить проверку ошибок
ввода-вывода}
if IOResult=0 then {Если операция «Открыть файл» выполнена
успешно}
begin
while not Eof(F) do {Пока не конец файла}
begin
Read(F,Ch); {Прочитать из файла символ}
Memo1.Text:=Memo1.Text+Ch; {Вывести символ в поле Memo}
end;
CloseFile(F); {Закрыть файл}
end
else {IOResult<>0 - операция «Открыть файл» не выполнена}
ShowMessage('Нет такого файла');
end;
```

Задание 3.

Открыть текстовый файл для дополнения, добавить в него текст из окна Edit.

Фрагмент кода программы:

```
var
F: TextFile; {Описание файловой переменной}
begin
AssignFile(F, 'primer.txt'); {Связь файловой переменной с
файлом}
Append(F); {Открыть существующий файл для добавления текста
в его конец}
Writeln(F,Edit1.Text);      {Записать в файл}
CloseFile(F);               {Закрыть файл}
end;
```

Задание 4.

Стихотворный текст (в строке не более 80 символов) записать «лесенкой» (по одному слову в строке).

Программный код:

```
var
Form1: TForm1;
f: TextFile; {Описание файловой переменной}
implementation
{$R *.dfm}
Procedure TForm1.Button1Click(Sender: TObject);
var
m: array[0..100] of string[80];
i,j,n: integer;
s: string;
begin
j:=0;
{Деление строк, записанных в Memo на слова и запись слов в
строковый массив m}
with Memo1 do
begin
for n:=0 to lines.count-1 do
begin
s:=lines[n]+'';
for i:=1 to length(s) do
begin
m[j]:=m[j]+copy(s,i,1);
if (copy(s,i,1)='') then j:=j+1;
end;
end;
end;
{Запись полученного строкового массива в файл text.txt}
AssignFile(f, 'text.txt'); {Связь файловой переменной с фай-
лом}
Rewrite(f); {Создать новый файл}
for i:=0 to j do
Writeln(f,m[i]);     {Запись слов в файл}
CloseFile(f);                {Закрыть файл}
end;
procedure TForm1.Button2Click(Sender: TObject);
var
s: string;
```

```
begin
Reset(f);     {Открыть файл для чтения}
Readln(f,s);
Memo2.Lines[0]:=s;
while not Eof(f) do {Пока не конец файла}
begin
Readln(f,s); {Прочитать из файла строку}
{Вывести прочитанную строку в Memo}
Memo2.Lines.Add(s);
end;
CloseFile(f);{Закрыть файл}
end;
end.
```

Задание 5.

Ввести двумерную матрицу из файла. Для открытия нужного файла использовать компонент `OpenDialog`.

Программный код:

```
var
i,j: integer;
F: TextFile;  {Описание файловой переменной}
Name: string;
begin
name:='';
{Открыть диалог и запомнить имя файла с диска в переменной
Name}
If OpenDialog1.Execute then Name:= OpenDialog1.FileName;
AssignFile(F, Name); {Связать файловую переменную с файлом
на диске}
{$I-}  {Директива компилятора: отключить проверку ошибок
ввода-вывода}
Reset(F); {Открыть файл для чтения}
{$I+}
if IOResult=0 then {Если операция «Открыть файл» выполнена
успешно}
begin
Readln(F,n,m); {Считывание из файла размерности матрицы}
for i:=0 to n-1 do
begin
for j:=0 to m-1 do
Read(f,a[I,j]); {Считывание из файла элементов строки матрицы}
```

```
Readln(f); {Перевод на новую строку файла}
end;
CloseFile(F); {Закрыть файл}
end
else
ShowMessage('Нет такого файла');
end;
```

Задания для самостоятельного выполнения

Задание 6.

Выполните индивидуальное задание из предложенных задач первого уровня.

При выполнении индивидуального задания необходимо текст считать из текстового файла. Результат выполнения программы записать в исходный файл.

№	Задание
1.	В тексте имеется одна точка с запятой (;). Подсчитать количество символов до точки с запятой и после нее.
2.	Дан произвольный русский текст. Проверить, каких букв в нем больше: гласных или согласных.
3.	Проверить, одинаковое ли число открывающихся и закрывающихся скобок в данном тексте.
4.	Определить, сколько раз в тексте встречается заданное слово.
5.	Удалить часть текста, заключенного в скобки (вместе со скобками).
6.	В тексте между словами вставить вместо пробела запятую и пробел.
7.	В тексте заменить все двоеточия (:) точкой с запятой (;). Подсчитать количество замен.
8.	Дан текст, в котором есть одна открывающаяся и одна закрывающаяся скобка. Вывести на экран все символы, расположенные внутри этих скобок.

9.	Дан текст. Подсчитать количество букв *k* в последнем слове.
10.	Дан текст. Определить, сколько раз входит в него группа букв ***abc***.
11.	Дан текст. Преобразовать его, удалив каждый символ «*».
12.	Дан текст, среди символов которого есть двоеточие (:). Определить, сколько символов ему предшествует.
13.	Дан текст. Определить, сколько в нем символов «*», «;», «:».
14.	Дан текст, содержащий английский текст. Найти количество слов, начинающихся с буквы b.
15.	Дан текст, заканчивающийся точкой. Подсчитать, сколько в нем слов.

Задание 7.

Выполните индивидуальное задание из предложенных задач второго уровня.

№	Задание
1.	Дан двумерный массив размером n*m. 1. Заменить четный элемент каждой строки нулем 2. Вставить после всех строк, содержащих минимальный элемент массива, строку 2, 4, 6,… 3. Удалить все столбцы, в которых встретится элемент, по модулю больший левого верхнего ($a_{1,1}$). 4. Поменять местами третий и последний столбцы.
2.	Дан двумерный массив размером n*m. 1. Заменить максимальный элемент каждой строки номером столбца, в которой он находится. 2. Вставить после всех столбцов, содержащих нулевой элемент, первый столбец. 3. Удалить все строки, в которых встретится четный отрицательный элемент. 4. Поменять местами первый и предпоследний столбцы.
3.	Дан двумерный массив размером n*m. 1. Заменить нечетный элемент каждой строки нулем 2. Вставить после всех строк, содержащих минимальное значение, строку 1, 2, 3,… 3. Удалить все столбцы, в которых первый элемент четный. 4. Поменять местами первый и последний столбцы.

4.	Дан двумерный массив размером n*m. 5. Заменить четный элемент каждого столбца максимальным по модулю. 6. Вставить после столбцов, содержащих минимальное значение, второй столбец. 7. Удалить все строки, в которых второй элемент больше предпоследнего. 8. Поменять местами первый и средний столбцы.
5.	Дан двумерный массив размером n*m. 1. Заменить элемент, кратный трем каждого столбца, нулем. 2. Вставить после каждого столбца, начиная со второго, первый столбец. 3. Удалить из него каждый столбец, содержащий элемент, кратный пяти. 4. Поменять местами третий и последний столбцы.
6.	Дан двумерный массив размером n*m. 1. Заменить отрицательный элемент каждого столбца нулем. 2. Вставить после каждого столбца, содержащего максимальный по модулю элемент, строку из нулей. 3. Удалить из него каждую строку, содержащую элемент, кратный трем. 4. Поменять местами первый и последний столбцы.
7.	Дан двумерный массив размером n*m. 1. Заменить нулевой элемент каждого столбца максимальным по модулю элементом массива. 2. Вставить после каждой строки, содержащей максимальный по модулю элемент, последнюю строку. 3. Удалить из него каждую строку, содержащую нулевой элемент. 4. Поменять местами два средних столбца.
8.	Дан двумерный массив размером n*m. 1. Заменить максимальный элемент каждого столбца нулем. 2. Вставить после всех строк, содержащих максимальный по модулю элемент, первую строку. 3. Удалить из него строку и столбец, на перекрестье которых находится максимальный по модулю элемент. 4. Поменять местами последний и предпоследний столбцы.
9.	Дан двумерный массив размером n*m. 1. Заменить максимальный элемент каждой строки нулем. 2. Вставить после каждого столбца, содержащего максимальный элемент массива, столбец из нулей.

	3. Удалить все столбцы, в которых встретится нечетный положительный элемент. 4. Поменять местами первый и предпоследний столбцы.
10.	Дан двумерный массив размером n*m. 1. Заменить максимальный элемент каждой строки нулем. 2. Вставить перед всеми строками, первый элемент которых делится на 3, строку из нулей. 3. Удалить самый левый столбец, в котором встретится четный отрицательный элемент. 4. Поменять местами второй и предпоследний столбцы.
11.	Дан двумерный массив размером n*m. 1. Заменить максимальный элемент каждой строки на противоположный по знаку. 2. Вставить после всех столбцов, содержащих максимальный элемент, столбец из нулей. 3. Удалить все столбцы, в которых есть отрицательный элемент. 4. Поменять местами первый и последний столбцы.
12.	Дан двумерный массив размером n*m. 1. Заменить минимальный по модулю элемент каждого столбца на противоположный. 2. Вставить после каждого столбца, содержащего значение равное нулю, столбец из нулей. 3. Удалить все строки, содержащие максимальные элементы. 4. Поменять местами первый и последний столбцы.
13.	Дан двумерный массив размером n*m. 1. Заменить все элементы первых трех столбцов на их квадраты. 2. Вставить после каждой нечетной строки первую строку. 3. Удалить все столбцы, в которых первый элемент больше последнего. 4. Поменять местами средние строки с первой и последней.
14.	Дан двумерный массив размером n*m. 1. Заменить минимальный по модулю элемент каждого столбца нулем. 2. Вставить после каждой строки, содержащей минимальное значение, строку из нулей. 3. Удалить все столбцы, в которых первый элемент больше последнего. 4. Поменять местами первый и последний столбцы.

15.	Дан двумерный массив размером n*m. 1. Заменить максимальный по модулю элемент каждой строки на противоположный по знаку. 2. Вставить после каждой четной строки первую строку. 3. Удалить все строки, содержащие ноль. 4. Поменять местами средние столбцы.

Задание 8.

Выполните индивидуальное задание из предложенных задач третьего уровня. В проект, выполненный по индивидуальному заданию второго уровня, добавить вывод результатов программы в исходный текстовый файл с комментариями.

Лист самооценки выполнения лабораторной работы № 7.

Каждый пункт оценивается в 1 балл, если с уверенностью отвечаете «да». Максимальное количество баллов – 5.

1) Самостоятельно ответили на все вопросы.
2) Самостоятельно выполнили задания № 1–7.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 8

Графические возможности Delphi

6 часов

Вы научитесь:

- использовать объект *Canvas* для рисования;
- использовать методы *Pen, Brush, LineTo и MoveTo, Ellipse, Arc, Pie, Rectangle, Polyline* и их свойства для рисования различных геометрических фигур;
- работать с графическими объектами в среде Delphi.

Теоретическая часть

Рисовать в Delphi можно непосредственно на поверхности формы. Поверхности формы соответствует объект **Canvas** (canvas переводится как «поверхность», «холст для рисования»). Холст состоит из отдельных точек – пикселей. Положение пикселя характеризуется его горизонтальной (X) и вертикальной (Y) координатами. Левый верхний пиксель имеет координаты (0, 0). Координаты возрастают сверху вниз и слева направо. Значения координат правой нижней точки холста зависят от размера холста.

Размер холста можно получить, обратившись к свойствам формы: ClientHeight и Clientwidth.

Карандаш используется для вычерчивания точек, линий, контуров геометрических фигур. Карандашу соответствуют свойства **canvas. Pen** (карандаш). Вид линии, которую оставляет карандаш на поверхности холста, определяет ее свойства.

Свойства объекта `Pen`:

Свойство	Определяет
`Color`	Задает цвет линии, вычерчиваемой карандашом.
`Width`	Задает толщину линии (в пикселях).
`Style`	Определяет вид (стиль) линии, которая может быть непрерывной или прерывистой, состоящей из штрихов различной длины.
`Mode`	Определяет, как будет формироваться цвет точек линии в зависимости от цвета точек холста, через которые эта линия прочерчивается. По умолчанию вся линия вычерчивается цветом, определяемым значением свойства Pen.Color.

Значения свойства Color:

Константа	Цвет	Константа	Цвет
clBlack	Черный	clSilver	Серебристый
clMaroon	Каштановый	clRed	Красный
clGreen	Зеленый	clLime	Салатный
clOlive	Оливковый	clBlue	Синий
clNavy	Темно-синий	clFuchsia	Ярко-розовый
clPurple	Розовый	clAqua	Бирюзовый
clTeal	Зелено-голубой	clWhite	Белый
clGray	Серый		

Значение свойства Type (определяет вид линии):

Константа	Вид линии
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 (используется, если не надо изображать границу области, например, прямоугольника)

Кисть (canvas.Brush) используется методами, обеспечивающими вычерчивание замкнутых областей, например геометрических фигур, для заливки (закрашивания) этих областей.

Свойства объекта **TBrush** (кисть):

Свойство	Определяет
Color	Цвет закрашивания замкнутой области, т.е. область полностью перекрывает фон.
Style	Стиль (тип) заполнения области, т.е. сквозь не заштрихованные участки области будет виден фон.

Значения свойства Style (определяют тип закрашивания):

Константа	Тип заполнения (заливки) 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

Вычерчивание **прямой линии** осуществляет метод **Canvas.LineTo(x,y).** Этот метод вычерчивает прямую линию от текущей позиции карандаша в точку с координатами, указанными при вызове метода. Начальную точку линии можно задать при помощи метода **MoveTo**, указав в качестве параметров координаты нового положения карандаша.

Метод **polyline** вычерчивает ломаную линию. В качестве параметра метод получает массив типа **TPoint**. Каждый элемент массива представляет собой запись, поля x и y которой содержат координаты точки перегиба ломаной.

Метод **Ellipse** вычерчивает **эллипс** или **окружность**, в зависимости от значений параметров. Инструкция вызова метода в общем виде выглядит следующим образом: `Объект.Canvas.Ellipse(x1, y1, x2, y2),`

где: объект – имя объекта (компонента), на поверхности которого выполняется вычерчивание;

x1, y1, x2, y2 – координаты прямоугольника, внутри которого вычерчивается эллипс или окружность, если прямоугольник является квадратом.

Вычерчивание **дуги** выполняет метод **Arc**, инструкция вызова которого в общем виде выглядит следующим образом:

`Объект.Canvas.Arc(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.

Начальная (конечная) точка – это точка пересечения границы эллипса и прямой, проведенной из центра эллипса в точку с координатами x3 и y3 (x4, y4). Дуга вычерчивается против часовой стрелки от начальной точки к конечной.

Метод **pie** вычерчивает **сектор** эллипса или круга. Инструкция вызова метода в общем виде выглядит следующим образом:

`Объект.Canvas.Pie(x1, y1, x2, y2, x3, y3, x4, y4),`

где: x1, y1, x2, y2 – параметры, определяющие эллипс (окружность), частью которого является сектор;

x3, y3, x4, y4 – параметры, определяющие координаты конечных точек прямых, являющихся границами сектора.

Начальные точки прямых совпадают с центром эллипса (окружности). Сектор вырезается против часовой стрелки от прямой заданной точкой с координатами (x3, y3), к прямой заданной точкой с координатами (x4, y4).

Прямоугольник вычерчивается методом **Rectangle**, инструкция вызова которого в общем виде выглядит следующим образом:

```
Объект.Canvas.Rectangle(x1, y1, x2, y2),
```

где: объект – имя объекта (компонента), на поверхности которого выполняется вычерчивание;

x1, y1 и x2, y2 – координаты левого верхнего и правого нижнего углов прямоугольника.

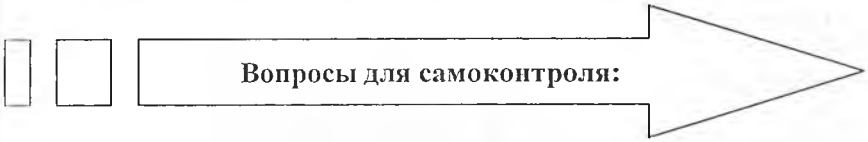
Метод **RoundRec** тоже вычерчивает **прямоугольник**, но **со скругленными углами**. Инструкция вызова метода RoundRec выглядит так:

```
Объект.Canvas.RoundRec(x1, y1, x2, y2, x3, y3),
```

где: x1, y1, x2, y2 – параметры, определяющие положение углов прямоугольника, в который вписывается прямоугольник со скругленными углами;

x3 и y3 – размер эллипса, одна четверть которого используется для вычерчивания скругленного угла.

Есть еще два метода, которые вычерчивают прямоугольник, используя в качестве инструмента только кисть (Brush). Метод **FillRect** вычерчивает **закрашенный прямоугольник**, а метод **FrameRect** – только **контур**. У каждого из этих методов лишь один параметр – структура типа TRect. Поля структуры TRect содержат координаты прямоугольной области, они могут быть заполнены при помощи функции Rect.



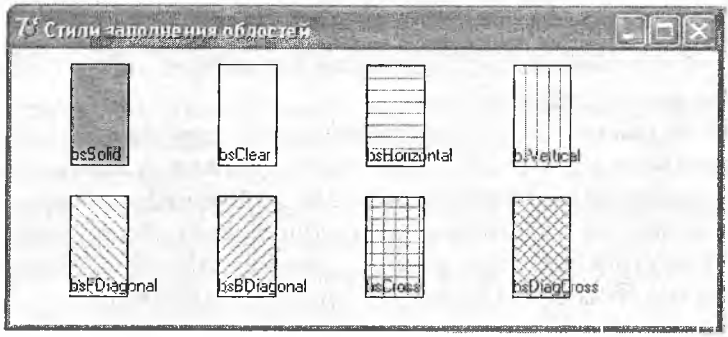
Вопросы для самоконтроля:

1. Какие геометрические фигуры можно рисовать в Delphi?
2. Каким образом нарисовать окружность, эллипс, дугу, сектор?
3. Каким образом нарисовать многоугольник?
4. Каким образом нарисовать ломаную линию?
5. В чем разница между методами **FillRect** и **FrameRect**?
6. Что такое массив типа **TPoint**?
7. Какие свойства имеет объект **TBrush**?
8. Какие свойства у объекта **Pen**?

Основная часть

Задание 1.

Составить программу под названием «Стили заполнения областей», которая выводит восемь прямоугольников, закрашенных зеленым цветом с использованием разных стилей.



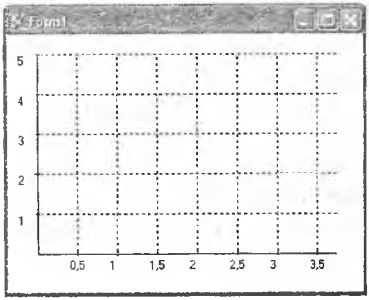
Код программы:

```
   procedure TForm1.FormPaint(Sender: TObject);
   const
   bsName: array[1..8] of string = ("bsSolid",'bsC
lear','bsHorizontal', "bsVertical", "bsFDiagonal",
"bsBDiagonal", "bsCross",'bsDiagCross');
     var
   x, y, w, h, k, i, j: integer;
   bs: TBrushStyle;
     begin
   w:=40; h:=30; y:=80;
   for I:=1 to 2 do
     begin
   x:=40;
   for j:=1 to 4 do
     begin
   k:=j+(i-1)*4;
   case k of
```

```
1:bs:=bsSolid;
2:bs:=bsClear;
3:bs:=bsHorizontal;
4:bs:=bsVertical;
5:bs:=bsFDiagonal;
6:bs:=bsBDiagonal;
7:bs:=bsCross;
8:bs:=bsDiagCross;
  end;
Canvas.Brush.Color:= clGreen;
Canvas.Brush.Style:= bs;
Canvas.Rectangle (x, y, x+w, y-w-h);
Canvas.Brush.Style:= bsClear;
Canvas.TextOut(x, y-15, bsName[k]);
x := x+w+60;
  end;
y:=y+h+60;
end; end; end.
```

Задание 2.

Составить программу, которая изображает на фоне координатные оси и оцифрованную сетку.



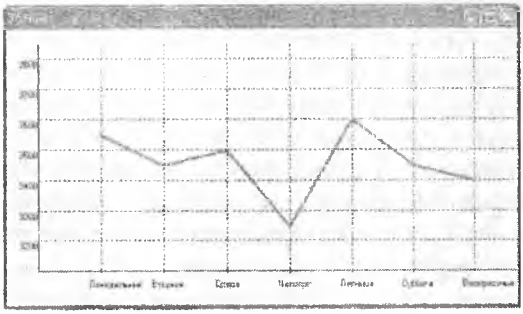
Код программы:

```
procedure TForm1.FormPaint(Sender: TObject);
var
x0,y0:integer;
dx,dy:integer;
```

```
h,w:integer;
x,y:integer;
lx,ly:real;
dlx,dly:real;
cross:integer;
dcross:integer;
begin
x0:=30; y0:=220; dx:=40; dy:=40; dcross:=1; dlx:=0.5;
dly:=1.0; h:=200; w:=300;
with Form1.Canvas do
begin
cross:=dcross;
MoveTo(x0,y0); LineTo(x0,y0-h);
MoveTo(x0,y0); LineTo(x0+w, y0);
x:=x0+dx; lx:=dlx;
repeat
MoveTo(x,y0-3);LineTo(x,y0+3);
cross:=cross-1;
if cross = 0 then begin
TextOut(x-8,y0+5,FloatToStr(lx));
cross:=dcross ; end;
Pen.Style:=psDot;
MoveTo(x,y0-3);LineTo(x,y0-h);
Pen.Style:=psSolid;
lx:=lx+dlx; x:=x+dx;
until (x>x0+w);
y:=y0-dy;
ly:=dly;
repeat
MoveTo(x0-3,y);LineTo(x0+3,y);
TextOut(x0-20,y,FloatToStr(ly));
Pen.Style:=psDot; MoveTo(x0+3,y); LineTo(x0+w,y);
Pen.Style:=psSolid; y:=y-dy;
ly:=ly+dly; until (y<y0-h);
end; end; end.
```

Задание 3.

Составить программу, отображающую динамику изменения стоимости данных бумаг с течением времени.



Код программы:

```
procedure TForm1.FormPaint(Sender: TObject);
  const
lbll: array [1..7] of string = ('Понедельник','Вторн
ик','Среда', 'Четверг','Пятница','Суббота','Воскресень
е');
Var
x0,y0:integer;
dx,dy:integer;
x,y:integer;
_height:integer;
_width:integer;
lbl_y:real;
dlbl_y:integer;
i:integer;
price: array [1..7] of TPoint;
begin
x0:=40;y0:=320;
dx:=80; dy:=40;
lbl_y:=3200;
dlbl_y:=100;
_height:=300;
_width:=900;
for i:=1 to 7 do price[i].x:=40+80*i;
price[1].y:=140;
price[2].y:=180; price[3].y:=160; price[4].y:=260;
price[5].y:=120; price[6].y:=180; price[7].y:=200;
with Form1.Canvas do
```

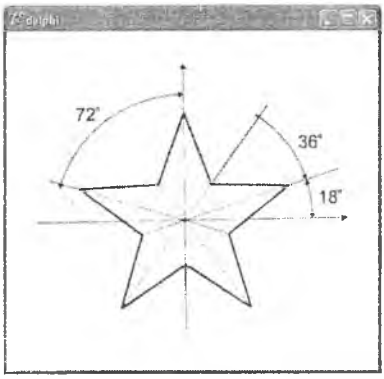
```
begin
Pen.Style:=psSolid;
Pen.Width:=1;
Pen.Color:=clBlack;
MoveTo(x0,y0);
LineTo(x0,y0-_height);
MoveTo(x0,y0);
LineTo(x0+_width,y0);
x:=x0+dx;
for i:=1 to 7 do
begin
MoveTo(x,y0-3); LineTo(x,y0+3);
TextOut(x-15,y0+10,lbll[i]);
Pen.Style:=psDot;
MoveTo(x,y0-3); LineTo(x,y0-_height);
Pen.Style:=psSolid;
x:=x+dx;
end;
y:=y0-dy;
for i:=1 to 7 do
begin
MoveTo(x0-3,y); LineTo(x0+3,y);
TextOut(x0-25,y,FloatToStr(lbl_y));
Pen.Style:=psDot;
MoveTo(x0+3,y); LineTo(x0+_width,y);
Pen.Style:=psSolid;
y:=y-dy;
lbl_y:=lbl_y+dlbl_y;
end;
Pen.Width:=3;
Pen.Color:=clRed;
MoveTo(x0+dx,y0-dy);
For i:=1 to 7 do PolyLine(price);
end;
end;
end.
```

Задание 4.

Нарисовать звезду в точке нажатия кнопки мыши.

Примечание. Цвет, которым вычерчивается звезда, зависит от того,

какая из кнопок мыши была нажата. Процедура обработки нажатия кнопки мыши (событие MouseDown) вызывает процедуру рисования звезды starLine и передает ей в качестве параметра координаты точки, в которой была нажата кнопка. Звезду вычерчивает процедура starLine, которая в качестве параметров получает координаты центра звезды и холст, на котором звезда должна быть выведена. Сначала вычисляются координаты концов и впадин звезды, которые записываются в массив p. Затем этот массив передается в качестве параметра методу Polyline. При вычислении координат лучей и впадин звезды используются функции sin и cos. Так как аргумент этих функций должен быть выражен в радианах, то значение угла в градусах умножается на величину pi/180, где pi – это стандартная именованная константа, равная числу π = 3,14.



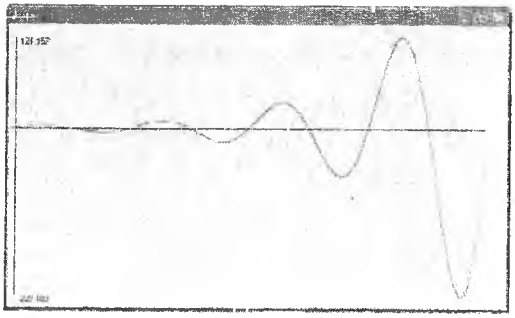
Код программы:

```
  procedure StarLine(x0,y0,r: integer; Canvas:
TCanvas);
  p: array [1.. 11] of TPoint;
  a: integer;
  i: integer;
  begin
  a := 18;
  for i:=1 to 10 do
  begin
  if (i mod 2=0) then
  begin
  p[i].x:= x0+Round(r/2*cos(a*pi/180));
```

```
p[i] .y:=y0-Round(r/2*sin(a*pi/180));
end;
else
begin
p [i] .x:=x0+Round(r*cos (a*pi/180));
p [i] .y:=y0-Round(r*sin(a*pi/180));
end;
a:= a+36;
end;
p[ll].X:= p[l].X;
Canvas. Polyline (p);
end;
procedure TForm1 . FormMouseDown {Sender : TObject;
Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
if Button = mbLeft
then Form1. Canvas . Pen . Color: = clRed
else Form1. Canvas. Pen. Color:= clGreen;
StarLine(x, y, 30, Form1. Canvas );
end;
end.
```

Задание 5.

Изобразить график функции $y = 2 \sin x \, e^{x/5}$



Код программы:

```
unit grfunc_;
interface
Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs;
type
TForm1 = class(TForm)
procedure FormPaint(Sender: TObject);
procedure FormResize(Sender: TObject);
private
{ Private declarations }
public
{Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
Function f(x:real):real;
begin
f:=2*Sin(x)*exp(x/5) ;
end;
procedure GrOfFunc;
var
x1,x2:real;
y1,y2:real;
x:real;
y:real;
dx:real;
l,b:integer;
w,h:integer;
mx,my:real;
x0,y0:integer;
begin
l:=10;
b:=Form1.ClientHeight-20;
h:=Form1.ClientHeight-40;
w:=Form1.Width-40;
x1:=0;
x2:=25;
dx:=0.01;
```

```
y1:=f(x1);
y2:=f(x1);
x:=x1;
repeat
y:= f (x);
If y < y1 then y1: =y;
If y > y2 then y2: =y;
x:=x+dx; until (x >= x2);
my:=h/abs(y2-y1);
mx:=w/abs(x2-x1);
x0:=1;
y0:=b-Abs(Round(y1*my));
with form1.Canvas do
begin // оси
MoveTo(1,b);LineTo(1,b-h);
MoveTo(x0,y0);LineTo(x0+w,y0);
TextOut(1+5,b-h,FloatToStrF(y2,ffGeneral,6,3));
TextOut(1+5,b,FloatToStrF(y1,ffGeneral,6,3));
x:=x1;
repeat
y:=f(x);
Pixels[x0+Round(x*mx),y0-Round(y*my)]:=clRed;
x:=x+dx;
until (x >= x2);
end;
end;
procedure TForm1.FormPaint(Sender: TObject);
begin
GrOfFunc;
end;
procedure TForm1.FormResize(Sender: TObject);
begin
form1.Canvas.FillRect(Rect(0,0,ClientWidth,
ClientHeight));
GrOfFunc;
end;
end.
```

Основную работу выполняет процедура GrOfFunc, которая сначала вычисляет максимальное (y2) и минимальное (y1) значения функции на от-

резке [x1, x2]. Затем, используя информацию о ширине (Form1.Clientwidth – 40) и высоте (Form1. ClientHeight – 40) области вывода графика, вычисляет масштаб по осям X (mx) и Y (my).

Высота и ширина области вывода графика определяются размерами рабочей (клиентской) области формы, т.е. без учета области заголовка и границ. После вычисления масштаба процедура вычисляет координату у горизонтальной оси (y0) и вычерчивает координатные оси графика. Затем выполняется непосредственное построение графика.

Вызов процедуры GrOfFunc выполняют процедуры обработки событий onPaint и onFormResize. Процедура TForm1. FormPaint обеспечивает вычерчивание графика после появления формы на экране в результате запуска программы, а также после появления формы во время работы программы, например, в результате удаления или перемещения других окон, полностью или частично перекрывающих окно программы. Процедура TForm1. FormResize обеспечивает вычерчивание графика после изменения размера формы.

Приведенная программа довольно универсальна. Заменив инструкции в теле функции f(x), можно получить график другой функции. Причем независимо от вида функции ее график будет занимать всю область, предназначенную для вывода.

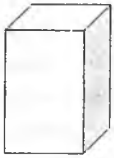
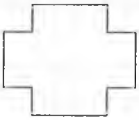
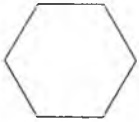
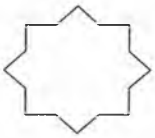
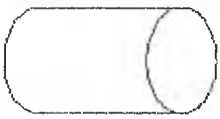
Задания для самостоятельного выполнения

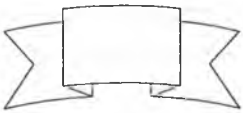
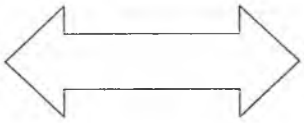
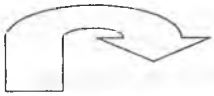
Задание 6.

Выполните индивидуальное задание из предложенных задач первого уровня.

Нарисуйте предложенный рисунок и дополните его своим рисунком.

№	Рисунок
1.	

2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	

10.	
11.	
12.	
13.	
14.	
15.	

Задание 7.

Выполните индивидуальное задание из предложенных задач второго уровня. Из заданий первого уровня составьте орнамент.

Задание 8.

Выполните индивидуальное задание из предложенных задач третьего уровня. Нарисовать графики функций.

№	Функция
1.	$y = 3\sin\sqrt{x} + 0{,}35x - 3{,}8$
2.	$y = 0{,}25x^3 + x - 1{,}2502$
3.	$y = x + \sqrt{x} + \sqrt[3]{x} - 2{,}5$
4.	$y = \sin(\ln x) - \cos(\ln x) + 2\ln x$
5.	$y = \cos\frac{2}{x} - 2\sin\frac{1}{x} + \frac{1}{x}$
6.	$y = 3x - 4\ln x - 5$
7.	$y = \sqrt{1-x} - \cos\sqrt{1-x}$
8.	$y = tgx - \frac{1}{3}tg^3x + \frac{1}{5}tg^5 - \frac{1}{3}$
9.	$y = 0{,}1x^2 - x\ln x$
10.	$y = x - 1/(3 + \sin 3{,}6x)$
11.	$y = x + \cos(x^{0{,}5x} + 2)$
12.	$y = 3\ln^2 x + 6\ln x - 5$
13.	$y = 3x - 14 + e^x - e^{-x}$
14.	$y = \sqrt{1-x} - tgx$
15.	$y = \cos x - e^{-\frac{x^2}{2}} + x - 1$

Лист самооценки выполнения лабораторной работы № 8.

Каждый пункт оценивается в 2 балла, если с уверенностью отвечаете «да». Максимальное количество баллов – 10.

1) Самостоятельно ответили на все вопросы из раздела «Вопросы для самоконтроля».
2) Самостоятельно выполнили задания № 1–8.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Лабораторная работа № 9

Работа с мультимедиа в Delphi

6 часов

Вы научитесь:

- использовать компонент *Animate* для вывода простой анимации;
- использовать компонент *MediaPlayer* для воспроизведения видеороликов, звука;
- создавать мультипликацию с помощью графических возможностей программы Delphi.

Теоретическая часть

Большинство современных программ, работающих в среде Windows, являются мультимедийными. Такие программы обеспечивают просмотр видеороликов и мультипликации, воспроизведение музыки, речи, звуковых эффектов. Типичными примерами мультимедийных программ являются игры и обучающие программы.

Delphi предоставляет в распоряжение программиста два компонента, которые позволяют разрабатывать мультимедийные программы:

• `Animate` – обеспечивает вывод простой анимации (подобной той, которую видит пользователь во время копирования файлов);

• `MediaPlayer` – позволяет решать более сложные задачи, например, воспроизводить видеоролики, звук, сопровождаемую звуком анимацию.

Компонент Animate

Компонент `Animate`, значок которого находится на вкладке Win32 (рис. 9.1), позволяет воспроизводить простую анимацию, кадры которой находятся в AVI-файле.

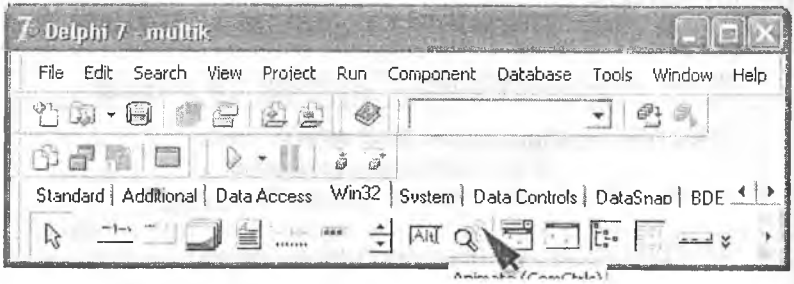


Рис. 9.1. Значок компонента Animate.

Примечание!

Хотя анимация, находящаяся в AVI-файле может сопровождаться звуковыми эффектами (так ли это – можно проверить, например, при помощи стандартной программы Проигрыватель Windows Media), компонент `Animate` обеспечивает воспроизведение только изображения. Для

полноценного воспроизведения сопровождаемой звуком анимации следует использовать компонент MediaPlayer.

Компонент Animate добавляется к форме обычным образом. После добавления компонента к форме следует установить значения его свойств. Свойства компонента Animate перечислены в табл. 9.1.

Таблица 9.1. Свойства компонента Animate Рис. 9.1. Значок компонента Animate.

Свойство	Определяет
Name	Имя компонента. Используется для доступа к свойствам компонента и управлением его поведением
FileName	Имя AVI-файла в котором находится анимация, отображаемая при помощи компонента
StartFrame	Номер кадра, с которого начинается отображение анимации
stopFrame	Номер кадра, на котором заканчивается отображение анимации
Activate	Признак активизации процесса отображения кадров анимации
Color	Цвет фона компонента (цвет "экрана"), на котором воспроизводится анимация
Transparent	Режим использования "прозрачного" цвета при отображении анимации
Repetitions	Количество повторов отображения анимации

Следует еще раз обратить внимание, что компонент Animate предназначен для воспроизведения AVI-файлов, которые содержат только анимацию. При попытке записать в свойство FileName имя файла, который содержит звук, Delphi выводит сообщение о невозможности открытия указанного файла (Cannot open AVI). Чтобы увидеть, что находится в AVI-файле: анимация и звук или только анимация, нужно из Windows раскрыть нужную папку, выделить AVI-файл и из контекстного меню выбрать команду **Свойства**. В результате этого откроется окно **Свойства**, на вкладке **Сводка** (рис. 9.2) которого будет выведена подробная информация о содержимом выбранного файла.

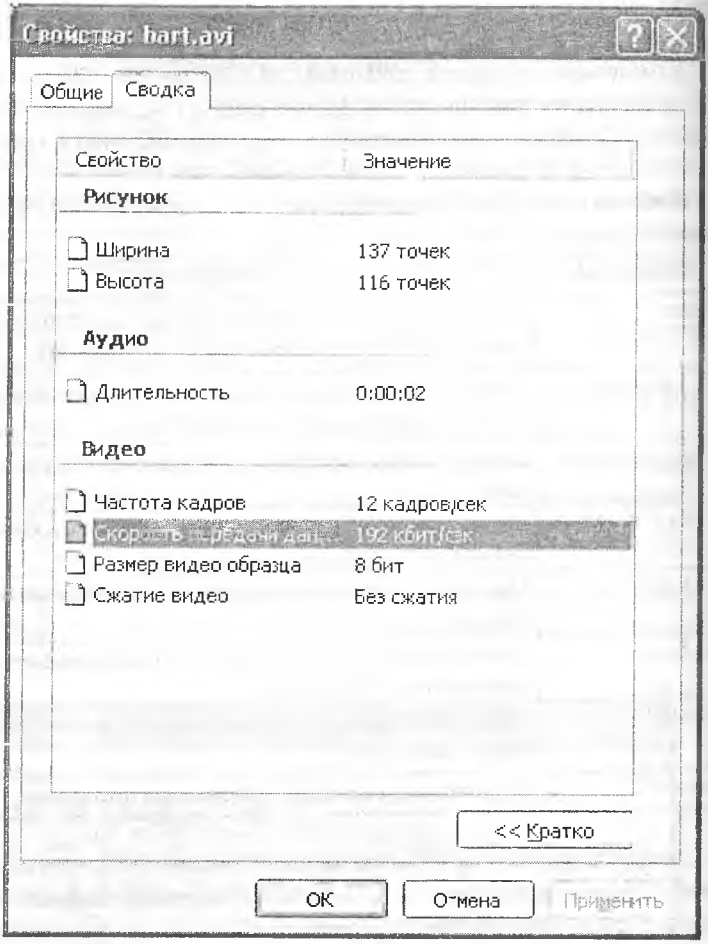


Рис. 9.2. На вкладке Сводка отражается информация об AVI-файле.

Компонент `Animate` позволяет программисту использовать в своих программах стандартные анимации Windows. Вид анимации определяется значением свойства `CommonAVI`. Значение свойства задается при помощи именованной константы. В табл. 9.2 приведены некоторые значения констант, вид анимации и описание процесса, для иллюстрации которого используются эти анимации.

Таблица 9.2. Значение свойства `CommonAVI` определяет анимацию

Значение	Анимация	Процесс
`aviCopyFiles`		Копирование файлов
`AviDeleteFile`		Удаление файла
`aviRecycleFile`		Удаление файла в корзину

Компонент MediaPlayer

Компонент `MediaPlayer`, значок которого находится на вкладке System (рис. 9.3), позволяет воспроизводить видеоролики, звук и сопровождаемую звуком анимацию.

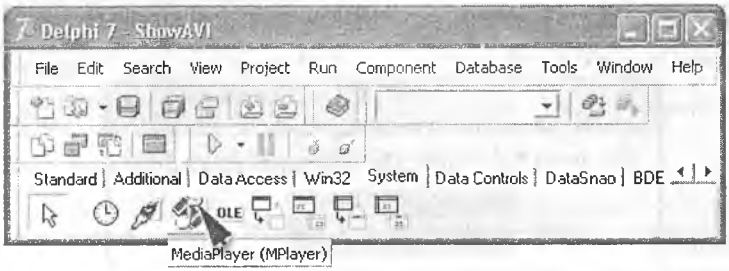


Рис. 9.3. Значок компонента MediaPlayer.

В результате добавления к форме компонента `MediaPlayer` на форме появляется группа кнопок (рис. 9.4), подобных тем, которые можно видеть на обычном аудио- или видеоплеере. Назначение этих кнопок пояснено в табл. 9.3. Свойства компонента `MediaPlayer` приведены в табл. 9.4.

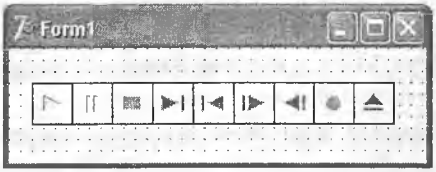


Рис. 9.4. Компонент MediaPlayer.

Таблица 9.3. Кнопки компонента MediaPlayer

Кнопка	Обозначение	Действие
Воспроизведение	btPlay	Воспроизведение звука или видео
Пауза	btPause	Приостановка воспроизведения
Стоп	btStop	Остановка воспроизведения
Следующий	btNext	Переход к следующему кадру
Предыдущий	btPrev	Переход к предыдущему кадру
Шаг	btStep	Переход к следующему звуковому фрагменту, например, к следующей песне на CD
Назад	btBack	Переход к предыдущему звуковому фрагменту, например, к предыдущей песне на CD
Запись	btRecord	Запись
Открыть/Закрыть	btEject	Открытие или закрытие CD-дисковода компьютера

Таблица 9.4. Свойства компонента MediaPiayer

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента и управлением работой плеера.
DeviceType	Тип устройства. Определяет конкретное устройство, которое представляет собой компонент MediaPiayer. Тип устройства задается именованной константой: dtAutoSelect — тип устройства определяется автоматически; dtVaweAudio — проигрыватель звука; dtAVivideo — видеопроигрыватель; dtCDAudio — CD-проигрыватель.
FileName	Имя файла, в котором находится воспроизводимый звуковой фрагмент или видеоролик.
AutoOpen	Признак автоматического открытия сразу после запуска программы, файла видеоролика или звукового фрагмента.

`Display`	Определяет компонент, на поверхности которого воспроизводится видеоролик (обычно в качестве экрана для отображения видео используют компонент Panel).
`VisibleButtons`	Составное свойство. Определяет видимые кнопки компонента. Позволяет сделать невидимыми некоторые кнопки.

Запись звука

В некоторых случаях программисту могут потребоваться специфические звуки или музыкальные фрагменты, которые не представлены на диске компьютера в виде WAV-файла. В этом случае возникает задача создания или, как говорят, записи WAV-файла.

Наиболее просто получить представление нужного звукового фрагмента в виде WAV-файла можно при помощи входящей в состав Windows программы **Звукозапись**. Программа **Звукозапись**, вид ее диалогового окна приведен на рис. 9.5, запускается из главного меню Windows при помощи команды **Пуск | Программы | Стандартные | Развлечения | Звукозапись.**

Рис. 9.5. Диалоговое окно программы **Звукозапись**.

Источником звука для программы **Звукозапись** может быть микрофон, аудио-CD или любое другое подключенное к линейному входу звуковой платы компьютера устройство, например аудиомагнитофон. Кроме того, возможно микширование (смешение) звуков различных источников.

Создается WAV-файл следующим образом. Сначала нужно определить источник (или источники) звука. Чтобы это сделать, надо открыть **Регулятор громкости** (для этого надо щелкнуть на находящемся на па-

нели задач изображении динамика и из появившегося меню выбрать команду **Регулятор громкости** и из меню **Параметры** выбрать команду **Свойства**. Затем в появившемся окне **Свойства** (рис. 9.6) выбрать переключатель **Запись** и в списке **Отображаемые регуляторы громкости** установить флажки, соответствующие тем устройствам, сигнал с которых нужно записать. После щелчка на кнопке ОК на экране появляется окно **Уровень записи** (рис. 9.7), используя которое можно управлять уровнем сигнала (громкостью) каждого источника звука в общем звуке и величиной общего, суммарного сигнала, поступающего на вход программы **Звукозапись**. Величина сигнала задается перемещением движков соответствующих регуляторов. Следует обратить внимание на то, что движки регуляторов группы **Уровень** доступны только во время процесса записи звука. На этом подготовительные действия заканчиваются. Теперь можно приступить непосредственно к записи звука.

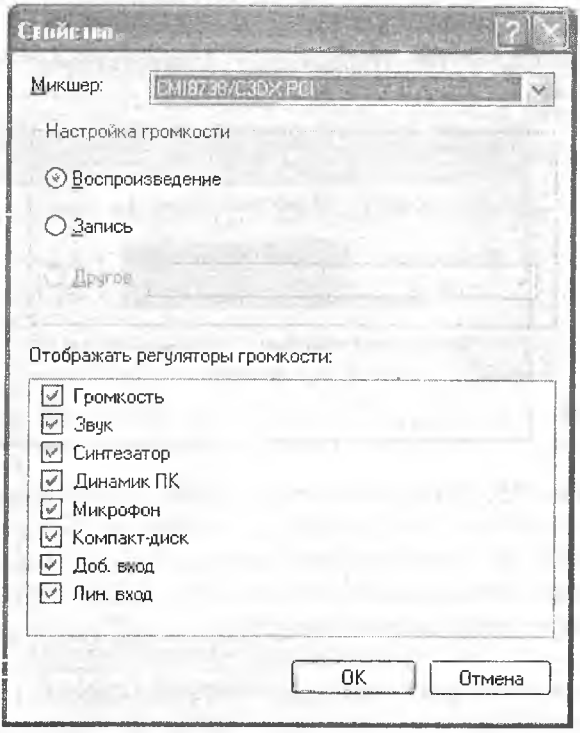


Рис. 9.6. Диалоговое окно **Свойства.**

Рис. 9.7. Диалоговое окно **Уровень записи** позволяет управлять записываемым сигналом.

Чтобы записать музыкальный или речевой фрагмент, надо запустить программу **Звукозапись**, активизировать диалоговое окно **Уровень**, выбрать устройство-источник звука, инициировать процесс звучания (если запись осуществляется, например, с CD) и в нужный момент времени щелкнуть на кнопке **Запись**.

Во время записи в диалоговых окнах можно наблюдать изменение сигнала на выходе микшера (индикатор **Громкость** диалогового окна **Уровень**) и на входе программы записи. На рис. 9.8 в качестве примера приведен вид диалогового окна **Звукозапись** во время записи звука.

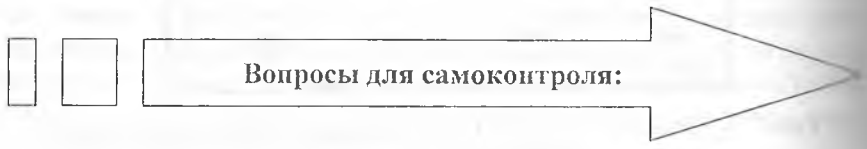
Рис. 9.8. Диалоговое окно **Звукозапись** во время записи.

Для остановки процесса записи следует щелкнуть на кнопке **Стоп**.

Сохраняется записанный фрагмент в файле обычным образом, т. е. выбором из меню **Файл** команды **Сохранить** или **Сохранить как**. При

выборе команды **Сохранить как** можно выбрать формат, в котором будет сохранен записанный звуковой фрагмент.

Существует несколько форматов звуковых файлов. В частности, тут можно сохранение звука с различным качеством как стерео, так и моно. Здесь следует понимать, что чем выше качество записи, тем больше места на диске компьютера требуется для хранения соответствующего WAV-файла. Считается, что для речи приемлемым является формат «22050 Гц, 8 бит, моно», а музыки – «44100 Гц, 16 бит, моно» или «44100 Гц, 16 бит, стерео».



1. Какие компоненты Delphi позволяют разрабатывать мультимедийные программы?
2. Для чего предназначен компонент Animate?
3. Какие данные позволяет воспроизводить компонент MediaPlayer?
4. Охарактеризуйте свойства компонента Animate.
5. Охарактеризуйте свойства компонента MediaPlayer.

Основная часть

Задание 1.

Создать программу, которая после запуска в форме выводит первый кадр анимации. Программа обеспечивает два режима просмотра анимации: непрерывный и покадровый. Вид формы программы приведен на рис. 9.10, значения свойств компонента Animate1 — в таблице 9.5.

Таблица 9.5. Значения свойств компонента `Animate1`

Свойство	Значение
`FileName`	`bart.avi`
`Active`	`False`
`Transparent`	`True`

Рис. 9.10. Форма программы **Просмотр анимации.**

Кнопка `Button1` используется как для инициализации процесса воспроизведения анимации, так и для его приостановки. Процесс непрерывного воспроизведения анимации инициирует процедура обработки события `Onclick` на кнопке Пуск, которая присваивает значение `True` свойству `Active`. Эта же процедура заменяет текст на кнопке `Button1` с Пуск на Стоп. Режим воспроизведения анимации выбирается при помощи переключателей `RadioButton1` и `RadioButton2`. Процедуры обработки события Onclick на этих переключателях изменением значения свойства Enabled блокируют или, наоборот, делают доступными кнопки управления: активизации воспроизведения анимации (`Button1`), перехода к следующему (`Buttor.2`) и предыдущему (`Buttons`) кадру. Во время непрерывного воспроизведения анимации процедура обработки события `OnCkick` на кнопке Стоп (`Button1`) присваивает значение `False` свойству Active и тем самым останавливает процесс воспроизведения анимации.

Код программы:

```
unit ShowAVI_; interface
uses
Windows, Messages, SysUtils,
Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, ComCtrls, ExtCtrls;
type
TForm1 = class(TForm)
Animate1: TAnimate; // Компонент Animate
Button1: TButton; // Кнопка Пуск-Стоп
Button2: TButton; // Следующий кадр
Button3: TButton; // Предыдущий кадр
RadioButton1: TRadioButton; // Просмотр всей анимации
RadioButton2: TRadioButton; // Покадровый просмотр
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure RadioButton1Click(Sender: TObject);
procedure RadioButton2Click(Sender: TObject);
private
{ Private declarations } public
{ Public declarations ) end;
var
Form1: TForm1; // Форма
CFrame: integer; // Номер отображаемого кадра
// в режиме покадрового просмотра
implementation {$R *.DFM}
// К следующему кадру
procedure TForm1.Button2Click(Sender: TObject);
begin
if CFrame = 1 then Button2.Enabled := True;
if CFrame < Animate1.FrameCount then begin
CFrame := CFrame + 1;
// Вывести кадр
Animate1.StartFrame := CFrame;
Animate1.StopFrame := CFrame;
Animate1.Active := True;
if CFrame = Animate1.FrameCount // Текущий кадр -
//последний
then Button2.Enabled:=False;
```

```
end;
end;
// К предыдущему кадру
procedure TForm1.Button3Click(Sender: TObject);
begin
if CFrame = Animate1.FrameCount
then Button2.Enabled := True;
if CFrame > 1 then begin
CFrame := CFrame - 1;
// Вывести кадр
Animate1.StartFrame := CFrame;
Animate1.StopFrame := CFrame;
Animate1.Active := True;
if CFrame = 1 // Текущий кадр - первый
then Form1.Button3.Enabled := False;
end;
end;
// Активизация режима просмотра всей анимации
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
Button1.Enabled:=True; //доступна кнопка Пуск
// Сделать недоступными кнопки покадрового просмотра
Form1.Button3.Enabled:=False ;
Form1.Button2.Enabled:=False;
end;
// Активизация режима покадрового просмотра
procedure TForm1.RadioButton2Click(Sender: TObject);
begin
Button2.Enabled:=True; // Кнопка Следующий кадр до-
ступна
Buttons.Enabled:=False; // Кнопка Предыдущий кадр
//недоступна
// Сделать недоступной кнопку Пуск - вывод всей анима-
ции
Button1.Enabled:=False; end;
// Пуск и остановка просмотра анимации
procedure TForm1.Button1Click(Sender: TObject);
begin
if Animate1.Active = False // В данный момент анимация
//не выводится
then begin
```

```
Animate1.StartFrame:=1; // Вывод с первого по послед-
ний // кадр
Animate1.StopFrame:=Animate1.FrameCount;
Animate1.Active:=True;
Button1.caption:='Стоп';
RadioButton2.Enabled:=False;
end
else // Анимация отображается
begin
Animate1.Active:=False; // Остановить отображение
Button1.caption:='Пуск';
RadioButton2.Enabled:=True;
end; end; end.
```

Задание 2 (Звуки Microsoft Windows).

Создать программу, в которой после появления диалогового окна воспроизводится «Звук Microsoft», затем пользователь может из списка выбрать любой из находящихся в каталоге C:\Windows\Media звуковых файлов и после щелчка на кнопке **Воспроизведение** услышать, что находится в этом файле.

Звуковые фрагменты находятся в файлах с расширением WAV. Например, в каталоге C:\Winnt\Media можно найти файлы со стандартными звуками Windows.

Данная программа (вид ее диалогового окна приведен на рис. 9.11) демонстрирует использование компонента MediaPiayer для воспроизведения звуковых фрагментов, находящихся в WAV-файлах.

Помимо компонента `MediaPiayer` на форме находится компонент `ListBox` и два компонента `Label`, первый из которых используется для вывода информационного сообщения, второй – для отображения имени WAV-файла, выбранного пользователем из списка.

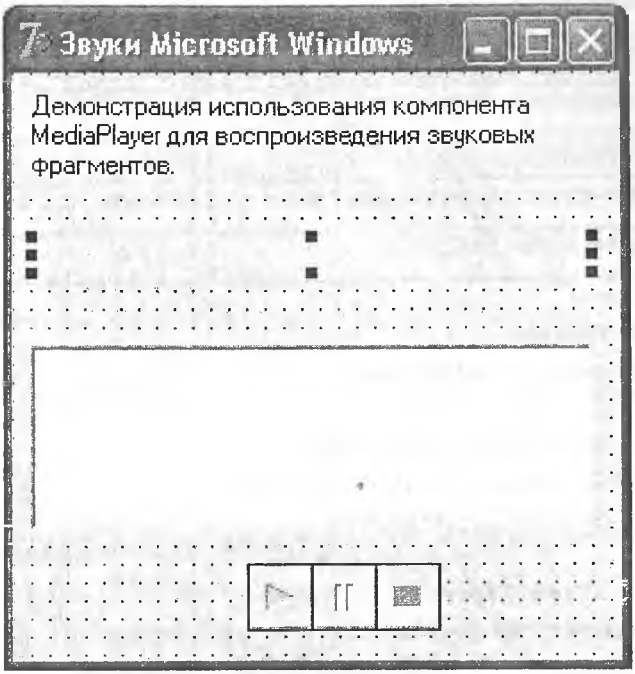


Рис. 9.11. Форма программы **Звуки Microsoft Windows.**

Значения измененных свойств компонента MediaPlayer1 приведены в табл. 9.6, значения остальных свойств оставлены без изменения.

Таблица 9.6. Значения свойств компонента MediaPlayer1

Компонент	Значение
DeviceType	DtAutoSelect
FileName	C:\Winnt\Media\Звук Microsoft.wav
AutoOpen	True

VisibleButtons.btNext	False
VisibleButtons.btPrev	False
VisibleButtons.btStep	False
VisibleButtons.btBack	False
VisibleButtons.btRecord	False
VisibleButtons.btEject	False

Код программы:

```
unit WinSound_; interface
uses
Windows, Messages, SysUtils,
Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, MPlayer;
type
TForm1 = class(TForm)
MediaPlayer1: TMediaPlayer; // Медиаплеер
Label1: TLabel; // Информационное сообщение
ListBox1: TListBox; // Список WAV-файлов
Label2: TLabel; // Выбранный из списка файл
procedure FormActivate(Sender: TObject);
procedure ListBox1Click(Sender: TObject);
procedure MediaPlayer1Click(Sender: TObject; Button:
TMPBtnType;
var DoDefault: Boolean); private
{ Private declarations } public
{ Public declarations } end;
const
SOUNDPATCH='c:\winnt\media\'; // Положение звуковых
// файлов
var
Form1: TForm1;
```

```
implementation
{$R *.DFM}
procedure TForm1.FormActivate(Sender: TObject);
var
SearchRec: TSearchRec; // Структура, содержащая
//информацию о файле, удовлетворяющем условию поиска
begin
Form1.MediaPlayer1.Play ;
// Сформируем список WAV-файлов, находящихся
// в каталоге c:\winnt\media
if FindFirst(SOUNDPATCH+'*.wav', faAnyFile, SearchRec)
=0 then
begin
// В каталоге есть файл с расширением WAV
// Добавим имя этого файла в список
Form1.ListBox1.Items.Add(SearchRec.Name) ;
// Пока в каталоге есть другие файлы с расширением WAV
while (FindNext(SearchRec) = 0) do
Form1.ListBox1.Items.Add(SearchRec.Name);
end;
end;
// Щелчок на элементе списка
procedure TForm1.ListBox1Click(Sender: TObject);
begin
// Вывести в поле метки Label2 имя выбранного файла
Label2.Caption:=ListBox1.Items[ListBox1.itemIndex];
end;
// Щелчок на кнопке компонента MediaPlayer
procedure TForm1.MediaPlayer1Click(Sender: TObject;
Button: TMPBtnType;
var DoDefault: Boolean); begin
```

```
if (Button = btPlay) and (Label2.Caption <> "") then
begin
// Нажата кнопка Play
with MediaPlayer1 do begin
FileName:=SOUNDPATCH+Label2.Caption; // имя выбранного
// файла
Open; // Открыть и проиграть звуковой файл
end; end; end; end.
```

Воспроизведение звука сразу после запуска программы активизирует процедура обработки события `onFormActivate` путем применением метода Play к компоненту `MediaPlayer1` (действие этого метода аналогично щелчку на кнопке **Воспроизведение**). Эта же процедура формирует список WAV-файлов, находящихся в каталоге C:\Winnt\Media. Для формирования списка используются функции `FindFirst` и `FindNext`, которые, соответственно, выполняют поиск первого и следующего (по отношению к последнему, найденному функцией `FindFirst` или `FindNext`) файла, удовлетворяющего указанному при вызове функций критерию. Обеим функциям в качестве параметров передаются маска WAV-файла (критерий поиска) и переменная структура `searchRec`, поле Name которой в случае успешного поиска будет содержать имя файла, удовлетворяющего критерию поиска.

Щелчок на элементе списка обрабатывается процедурой `TForm1.ListBox1Click`, которая выводит в поле метки Label2 имя файла, выбранного пользователем (во время работы программы свойство `ItemIndex` содержит номер элемента списка на котором выполнен щелчок).

В результате щелчка на одной из кнопок компонента MediaPiayeri активизируется процедура `TForm1.MediaPiayer1Click`, которая проверяет, какая из кнопок компонента была нажата. Если нажата кнопка Воспроизведение (`btPlay`), то в свойство `FileName` компонента MediaPiayer1 записывается имя выбранного пользователем файла, затем метод Open загружает этот файл и активизирует процесс его воспроизведения.

Задание 3.

Создать программу, которая пересчитывает вес из фунтов в кило-

граммы и сопровождает выдачу результата звуковым сигналом. В случае, если пользователь забудет ввести исходные данные или введет их неверно, программа должна вывести сообщение об ошибке, также сопровождаемое звуковым сигналом.

Наличие у компонента `MediaPiayer` свойства `visible` позволяет скрыть компонент от пользователя и при этом применять его для воспроизведения звука без участия пользователя. Вид диалогового окна программы во время ее разработки приведен на рис. 9.12, значения свойств компонента MediaPlayer в табл. 9.7.

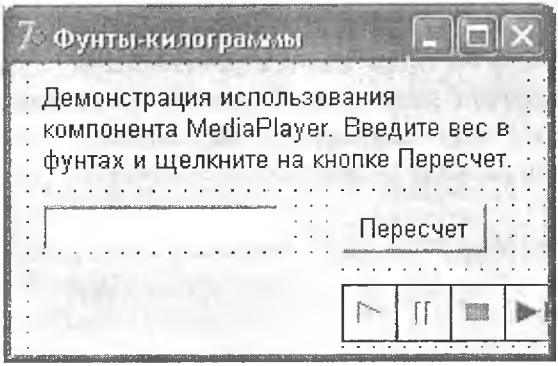


Рис. 9.12. Диалоговое окно программы **Фунты-килограммы.**

Таблица 9.7. Значения свойств компонента MediaPiayer1

Свойство	Значение
`Name`	`MediaPiayer1`
`DeviceType`	`dtAutoSelect`
`FileName`	`c:\winnt\media\ding.wav`
`AutoOpen`	`True`
`Visible`	`False`

Код программы:

```
unit FuntToKg1_; interface
uses
Windows, Messages, SysUtils,
Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls, MPlayer;
type
TForm1 = class(TForm)
Edit1: TEdit; // Поле ввода веса в фунтах
Button1: TButton; // Кнопка Пересчет
Label2: TLabel; // Поле вывода результата
Label1: TLabel; // Поле информационного сообщения
MediaPlayer1: TMediaPlayer; // Медиаплеер
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM)
// Щелчок на кнопке Пересчет
procedure TForm1.Button1Click(Sender: TObject);
var
f: real; // Вес в фунтах
k: real; // Вес в килограммах
begin
form1.Label2.Caption: =' ";
try // Возможна ошибка, если в поле
// Edit1 будет не число
f:=StrToFloat(Edit1.Text);
```

```
Form1.MediaPlayer1.Play;
// Звуковой сигнал
k:=f*0.4095;
Label2.caption:=Edit1.text+' ф. - это " +
FloatToStrF(k,ffGeneral,4,2}+' кг. ";
except
on EConvertError do // Ошибка преобразования
begin
// Определим и проиграем звук «Ошибка»
Form1.MediaPlayer1.FileName:=
"c:\windows\media\chord.wav";
Form1.MediaPlayer1.Open;
Form1.MediaPlayer1.Play; // Звуковой сигнал
ShowMessage("Ошибка! Вес следует ввести числом.");
form1.Edit1.SetFocus; // Курсор в поле ввода
// Восстановим звук
Form1.MediaPlayer1.FileName:=
"c:\windows\media\ding.wav";
Form1.MediaPlayer1.Open;
end; end; end;  end.
```

Задание 4. Просмотр видеороликов и анимации.

Составить программу, которая в результате щелчка на командной кнопке воспроизводит на поверхности формы простую, сопровождаемую звуковым эффектом, мультипликацию (в данном примере используется мультипликация со вращающимся по часовой стрелке словом Delphi).

Помимо воспроизведения звука, компонент MediaPiayer позволяет просматривать видеоролики и мультипликации, представленные как AVI-файлы (AVI – это сокращение от Audio Video Interleave, что переводится как чередование звука и видео, т.е. AVI-файл содержит как звуковую, так и видеоинформацию).

Вид диалогового окна программы приведен на рис. 9.13, а значения свойств компонента MediaPlayerl – в табл. 9.8.

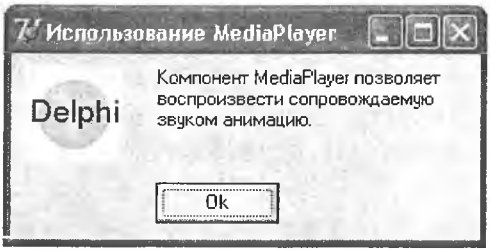


Рис. 9.13. Диалоговое окно программы Использование MediaPlayer.

Таблица 9.8. Значения свойств компонента MediaPlayer1

Свойство	Значение
Name	MediaPlayer1
FileName	delphi.avi
DeviceType	dtAVIVideo
AutoOpen	True
Display	Panel1
Visible	False

Создается форма приложения обычным образом. Компонент Panel1 используется в качестве экрана, на который осуществляется вывод анимации, и его имя принимается в качестве значения свойства Display компонента MediaPlayer1. Поэтому сначала к форме лучше добавить компонент Panel и затем – MediaPlayer. Такой порядок создания формы позволяет установить значение свойства Display путем выбора из списка.

Следует особо обратить внимание на то, что размер области вывода анимации на панели определяется не значениями свойств Width и Height панели (хотя их значения должны быть как минимум такими же, как ширина и высота анимации). Размер области определяется значением свойства.

DisplayRect компонента MediaPlayer. Свойство DisplayRect ВО время разработки программы недоступно (его значение не выводится в окне

Object Inspector). Поэтому значение свойства DisplayRect устанавливается во время работы программы в результате выполнения инструкции

```
MediaPlayer1.DisplayReet:=Rect(0,0,60,60).
```

Замечание

Чтобы получить информацию о размере кадров AVI-файла, надо, используя возможности Windows, открыть папку, в которой находится этот файл, щелкнуть правой кнопкой мыши на имени файла, выбрать команду **Свойства** и в появившемся диалоговом окне – вкладку Сводка, в которой выводится подробная информация о файле, в том числе и размер кадров.

Код программы:

```
uses
Windows, Messages, SysUtils,
Classes, Graphics, Controls,
Forms, Dialogs, MPlayer, StdCtrls, ExtCtrls;
type
TForm1 = class(TForm)
Label1: TLabel; // информационное сообщение
Panel1: TPanel; // панель, на которую выводится анима-
ция
Button1: TButton; // кнопка OK
MediaPlayer1: TMediaPlayer; { универсальный проигрыва-
тель }
procedure Button1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
{ Private declarations } public
{ Public declarations } end;
var
Form1: TForm1 ;
implementation
{$R *.DFM}
procedure TForm1.Button1Click(Sender: TObject);
begin
MediaPlayer1.Play; // воспроизведение анимации
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
// зададим размер области вывода анимации
// на поверхности формы
MediaPlayer1.DisplayRect:=Rect(0,0,60,60);
end; end.
```

Процесс воспроизведения анимации активизируется применением метода Play, что эквивалентно нажатию кнопки **Play** в случае, если кнопки компонента MediaPlayer доступны пользователю.

Задание 5. Мультипликация.

Составить программу, которая демонстрирует движение окружности от левой к правой границе окна программы.

Под мультипликацией обычно понимается движущийся и меняющийся рисунок. В простейшем случае рисунок может только двигаться или только меняться.

Рисунок может быть сформирован из графических примитивов (линий, окружностей, дуг, многоугольников и т.д.). Обеспечить перемещение рисунка довольно просто: надо сначала вывести рисунок на экран, затем через некоторое время стереть его и снова вывести этот же рисунок, но уже на некотором расстоянии от его первоначального положения. Подбором времени между выводом и удалением рисунка, а также расстояния между старым и новым положением рисунка (шага перемещения), можно добиться того, что у наблюдателя будет складываться впечатление, что рисунок равномерно движется по экрану.

Вид формы приведен на рис. 9.14.

Рис. 9.14. Форма программы **Движущаяся окружность**.

Код программы:

```
unit mcircle_;
interface
uses
Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms,
Dialogs, ExtCtrls, StdCtrls;
type
TForm1 = class(TForm) Timer1: TTimer;
procedure Timer1Timer(Sender: TObject};
procedure FormActivate(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
implementation
{$R *.DFM}
var
Form1: TForm1;
x,y: byte; // Координаты центра окружности
dx: byte; // Приращение координаты x при движении
//окружности
// Стирает и рисует окружность на новом месте
procedure Ris;
begin
// Стереть окружность
form1.Canvas.Pen.Color:=form1.Color;
form1.Canvas.Ellipse(x,y,x+10,y+10);
x:=x+dx;
// Нарисовать окружность на новом месте
form1.Canvas.Pen.Color:=clBlack;
form1.Canvas.Ellipse(x,y, x+10, y+10) ;
end;
// Сигнал от таймера
procedure TForm1.Timer1Timer(Sender: TObject);
begin Ris; end;
procedure TForm1.FormActivate(Sender: TObject);
begin
x:=0;
```

```
y:=10;
dx:=5;
timer1.Interval:=50;
// Период возникновения события OnTimer —0.5 сек
form1.canvas.brush.color:=form1.color;
end; end.
```

Основную работу выполняет процедура Ris, которая стирает окружность и выводит ее на новом месте. Стирание окружности выполняется путем перерисовки окружности поверх нарисованной, но цветом фона.

Для обеспечения периодического вызова процедуры Ris в форму программы добавлен невизуальный компонент Timer (таймер), значок которого находится на вкладке **System** палитры компонентов (рис. 9.15). Свойства компонента Timer перечислены в табл. 9.9.

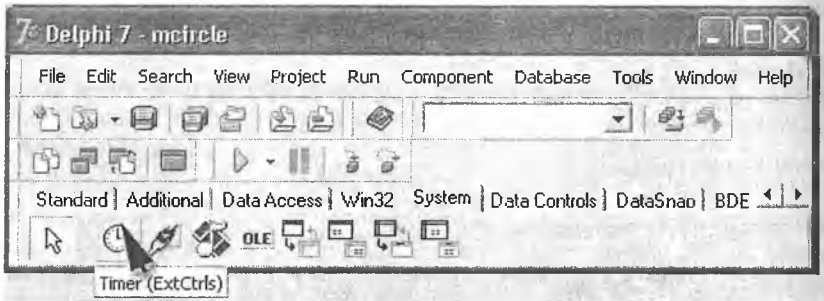


Рис. 9.15. Значок компонента Timer.

Таблица 9.9. Свойства компонента Timer

Свойство	Определяет
Name	Имя компонента. Используется для доступа к компоненту.
Interval	Период генерации события OnTimer. Задается в миллисекундах.
Enabled	Разрешение работы. Разрешает (значение True) или запрещает (значение False) генерацию события OnTimer.

Добавляется компонент Timer к форме обычным образом, однако, поскольку компонент Timer является невизуальным, т.е. во время работы пркграммы не отображается на форме, его значок можно поместить в любое место формы.

Компонент Timer генерирует событие OnTimer. Период возникновения события OnTimer измеряется в миллисекундах и определяется значением свойства Interval. Следует обратить внимание на свойство Enabled. Оно дает возможность программе «запустить» или «остановить» таймер. Если значение свойства Enabled равно False, то событие OnTimer не возникает.

Событие onTimer в рассматриваемой программе обрабатывается процедурой Timer1Timer, которая, в свою очередь, вызывает процедуру Ris. Таким образом, в программе реализован механизм периодического вызова процедуры Ris.

Примечание

Переменные x, y (координаты центра окружности) и dx (приращение координаты x при движении окружности) объявлены вне процедуры Ris, т.е. они являются глобальными. Поэтому надо не забыть выполнить их инициализацию (в программе инициализацию глобальных переменных реализует процедура FormActivate).

Задание 6. Мультипликация сложных изображений.

Составить программу, которая выводит на экран изображение перемещающегося кораблика.

При программировании сложных изображений, состоящих из множества элементов, используется метод, который называется ***методом базовой точки.*** Суть этого метода заключается в следующем:

1. Выбирается некоторая точка изображения, которая принимается за базовую.
2. Координаты остальных точек отсчитываются от базовой точки.
3. Если координаты точек изображения отсчитывать от базовой в относительных единицах, а не в пикселах, то обеспечивается возможность масштабирования изображения.

На рис. 9.16 приведено изображение кораблика. Базовой точкой является точка с координатами (X0 Y0). Координаты остальных точек отсчитываются именно от этой точки.

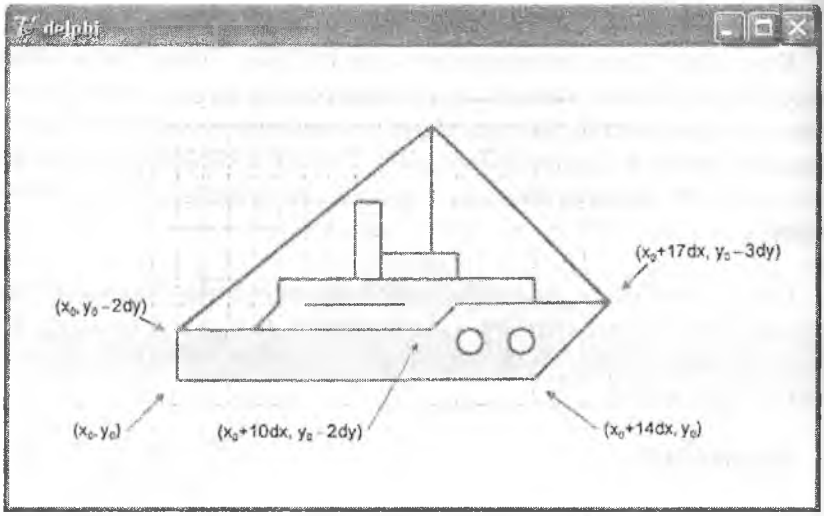


Рис. 9.16. Определение базовой точки изображения.

Код программы:

```
unit ship_;
interface
uses
Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
StdCtrls, ExtCtrls;
type
TForm1 = class(TForm)
Timer1: TTimer;
procedure Timer1Timer(Sender: TObject);
procedure FormActivate(Sender: TObject);
private
{ Private declarations } public
```

```
{ Public declarations } end;
var
Form1: TForm1;
x,y: integer; // Координаты корабля (базовой точки)
implementation
{$R *.DFM}
// Вычерчивает кораблик
procedure Titanik(x,y: integer; color: TColor);
const dx = 5; dy = 5;
var
buf: TColor;
begin
with form1.canvas do begin
buf:=pen.Color; // Сохраним текущий цвет
pen.Color:=color;
// Установим нужный цвет
// Рисуем . . .
// корпус MoveTo(x,y);
LineTo(x,y-2*dy) ;
LineTo (x+10*dx, y-2*dy);
LineTo (x+11*dx, y-3*dy);
LineTo (x+17*dx,y-3*dy);
LineTo (x+14*dx, y);
LineTo (x,y) ;
// надстройка
MoveTo(x+3*dx,y-2*dy);
LineTo (x+4*dx, y-3*dy);
LineTo (x+4*dx, y-4*dy);
LineTo (x+13*dx,y-4*dy);
LineTo (x+13*dx, y-3*dy);
MoveTo(x+5*dx,y-3*dy);
LineTo (x+9*dx, y-3*dy);
// Капитанский мостик
Rectangle (x+8*dx, y-4*dy, x+11*dx, y-5*dy);
```

```
// Труба
Rectangle (x+7*dx, y-4*dy, x+8*dx, y-7*dy);
// Иллюминаторы
Ellipse (x+11*dx,y-2*dy,x+12*dx,y-1*dy);
Ellipse (x+13*dx, y-2*dy, x+14*dx, y-1*dy);
// Мачта
MoveTo(.x+10*dx,y-5*dy) ; LineTo(x+10*dx,y-10*dy);
// Оснастка
MoveTo(x+17*dx,y-3*dy);
LineTo(x+10*dx,y-10*dy);
LineTo(x,y-2*dy);
pen.Color:=buf; // Восстановим старый цвет карандаша
end;
end;
// Обработка сигнала таймера
procedure TForm1.Timer1Timer(Sender: TObject);
begin
Titanik(x,y,form1.color); // Стереть рисунок
if x < Form1.ClientWidth
then x := x+5
else begin // Новый рейс
x := 0;
y := Random(50) + 100;
end;
Titanik(x,y,clWhite); // Нарисовать в новой точке
end;
procedure TForm1.FormActivate(Sender: TObject);
begin
x:=0; y:=100;
Form1.Color:=clNavy;
// Сигнал таймера каждые 50 миллисекунд
Timer1.Interval:= 50;
end; end.
```

Отрисовку и стирание изображения кораблика выполняет процедура Titanik, которая получает в качестве параметров координаты базовой точки и цвет, которым надо вычертить изображение кораблика. Если при вызове процедуры цвет отличается от цвета фона формы, то процедура рисует кораблик, а если совпадает – то «стирает». В процедуре Titanik объявлены константы dx и dy, определяющие шаг (в пикселах), используемый при вычислении координат точек изображения. Меняя значения этих констант, можно проводить масштабирование изображения.

Задания для самостоятельного выполнения

Задание 7.

Выполните индивидуальное задание первого уровня из лабораторной работы № 1, сопроводив выдачу результата звуковым сигналом. В случае если пользователь забудет ввести исходные данные или введет их неверно, программа должна вывести сообщение об ошибке, также сопровождаемое звуковым сигналом.

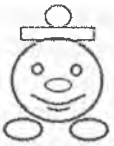
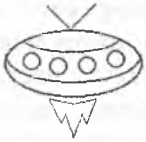
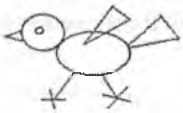
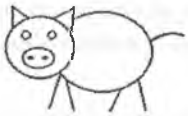
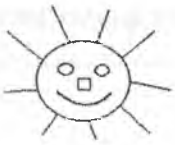
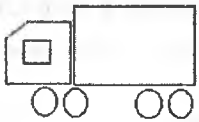
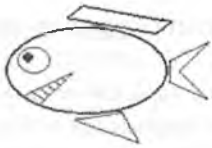
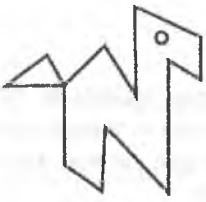
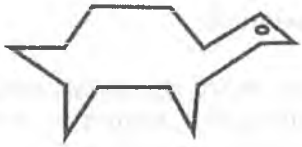
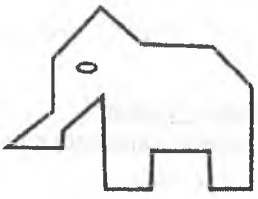
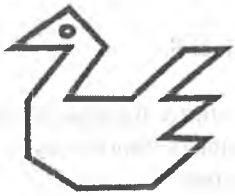
Задание 8.

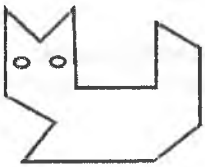
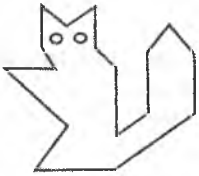
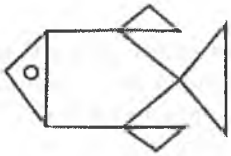
Выполните индивидуальное задание второго уровня из лабораторной работы № 1, в котором условие задачи записано в виде аудиофайла и воспроизводится по щелчку на соответствующей кнопке (подготовить аудиофайл с помощью программы Медиаплеер).

Задание 9.

Выполните индивидуальное задание третьего уровня.

Составить программу, которая выводит перемещающуюся по форме изображение.

№	Задание	№	Задание
1		2	
3		4	
5		6	
7		8	
9		10	
11		12	

13		14	
15		16	

Лист самооценки выполнения лабораторной работы № 9.

Каждый пункт оценивается в 2 балла, если с уверенностью отвечаете «да». Максимальное количество баллов – 10.

1) Самостоятельно ответили на все вопросы из раздела «Вопросы для самоконтроля».
2) Самостоятельно выполнили задания № 1–6.
3) Самостоятельно выполнили задание первого уровня.
4) Самостоятельно выполнили задание второго уровня.
5) Самостоятельно выполнили задание третьего уровня.

Тестовые задания

1. Объектно-ориентированный язык программирования:

a) Basic,
b) Algol,
c) Delphi,
d) Assembler.

2. Какая часть среды Delphi первоначально состоит из одного пустого окна, которое затем заполняется объектами?

a) Дизайнер форм,
b) Редактор текста,
c) Палитра компонента,
d) Инспектор объектов.

3. Где находятся различные готовые объекты в среде Delphi?

a) Дизайнер форм,
b) Редактор текста,
c) Палитра компонента,
d) Инспектор объектов.

4. Какая часть среды Delphi состоит из двух страниц, каждую из которых можно использовать для настройки вида и поведения выбранного компонента?

a) Дизайнер форм,
b) Редактор текста,
c) Палитра компонента,
d) Инспектор объектов.

5. Какое меню содержит команды для компиляции и сборки проектов?

a) Project,
b) Run,
c) File,
d) Edit.

6. Какое меню содержит команды для выполнения операций с проектами, модулями и файлами?

a) Project,
b) Run,

c) File,
d) Edit.

7. Какое меню содержит команды, осуществляющие операции редактирования?

a) Project,
b) Run,
c) File,
d) Edit.

8. Как открыть новый проект?

a) File – New Application,
b) File – New Data Module,
c) File – Open,
d) File – New.

9. Какое свойство формы устанавливает название окна формы?

a) Position,
b) Name,
c) Width,
d) Caption.

10. Какое свойство формы устанавливает имя компонента в программе?

a) Position,
b) Name,
c) Width,
d) Caption.

11. Какой компонент предназначен для ввода и отображения короткого текста?

a) TLabel,
b) TEdit,
c) TMemo,
d) TButton.

12. Компонент для вставки кнопки:

a) TLabel,
b) TEdit,
c) TMemo,
d) TButton.

13. К какому типу относится свойство Font?

a) Простое,
b) Перечисляемое,
c) Сложное,
d) Вложенное.

14. К какому типу относится свойство Caption?

a) Простое,
b) Перечисляемое,
c) Сложное,
d) Вложенное.

15. К какому типу относится свойство Height?

a) Простое,
b) Перечисляемое,
c) Сложное,
d) Вложенное.

16. Какая страница Палитры Компонент содержит объекты, позволяющие создать более красивый пользовательский интерфейс программы?

a) Standart,
b) Additional,
c) Dialogs,
d) System.

17. Какое расширение имеет главный файл проекта?

a) .dpr,
b) .dfm,
c) .pas,
d) .res.

18. Какое расширение имеет первый модуль программы?

a) .dpr,
b) .dfm,
c) .pas,
d) .res.

Список литературы

1. Каримов И.А. Мировой финансово-экономический кризис, пути и меры по его преодолению в условиях Узбекистана. – Ташкент: Узбекистан, 2009. – 48 с.
2. Каримов И.А. Гармонично развитое поколение – основа прогресса Узбекистана. – Ташкент: Узбекистан, 1998. – 64 с.
3. Закон Республики Узбекистан «Об образовании» // Халқ таълими. – Тошкент, 1997. – № 5. – С. 4–16.
4. Национальная программа по подготовке кадров // Халқ таълими. – Тошкент, 1998. – № 1. – С. 5–41.
5. Государственный стандарт Узбекистана. Требования к необходимому содержанию и уровню подготовленности бакалавра по направлению «математика-информатика». – Ташкент, ТДПУ, 2001. – 31 с.
6. Ашуров М., Мирмахмудов М., Сапаев Ш. Замонавий дастурлаш тиллари фанидан лаборатория ишлари / – Тошкент, 2008. – 67 с.
7. Ахраров Ш.С., Абдурахимов М., Мирзаахмедов А.М. Компьютерная технология реализации лабораторно-практических занятий в учебно-производственной практике // Ж. Наука. Образование. Техника. – 2004. – № 1. – С. 104–107.
8. Бакиев Р.Р. Узлуксиз таълим тизимида информатика ўқитишнинг концептуал асослари, мазмуни ва методикаси // Ж. Педагогик таълим. – Ташкент, 2005. – № 4. – С. 15–18.
9. Вирт Н. Алгоритмы и структуры данных / Пер. с англ. – М.: Мир, 1989. – 360 с.
10. Гринзоу Лу. Философия программирования для Windows 95/NT / Пер. с англ. – СПб.: Символ-Плюс, 1997. – 640 с.
11. Закирова Ф. Информационное обеспечение образовательного процесса и его структура // Педагогик таълим. – Ташкент, 2004. – № 5. – С. 34–35.
12. Информатика и ИТ. Учеб. пособие для педагогических вузов // Закирова Ф., Набиулина Л., Саратовская А., Ли О. – Ташкент: Аълокачи, 2007. – 173 с.
13. Замонавий дастурлаш тиллари // Дастур. Бакиев Р., Мамаражабов М., Закирова Ф., Ашуров М. – Ташкент, ТДПУ – 2008.
14. Культин Н. Основы программирования в Delphi 7. – Санкт-Петербург: BXБ-Петербург. 2003. – 598 с.
15. Набиулина Л.М. Развитие самостоятельности и активности студентов на лабораторных занятиях по информатике // Олима аёлларнинг фан-техника тараққиётида тутган ўрни. Республика илмий-амалий анжумани материаллари. – Тошкент, 2008. – С. 167–168.
16. Назиров Ш., Мусаев М., Неъматов А., Кобулов Р. Delphi tilida dasturlash asoslari. – Тошкент: Gulom. 2007–277 б.

17. Нормативные документы по реформированию системы высшего образования РУз. – Ташкент: Узбекистон, 1996.
18. Основные положения о лабораторных и практических занятиях// http://www.ta-bsatu.km.ru.
19. Практическое руководство по программированию / Пер. с англ. Б. Мик, П. Хит, Н. Рашби и др.; под ред. Б. Мика, П. Хит, Н. Рашби. – М.: Радио и связь, 1986. – 168 с.
20. Роберт И.В. Организация подготовки специалистов в области теории и методики информатизации образования //http:/ www.omsk.edu/ volume/2006/comp-edu/.
21. Узлуксиз таълим тизими учун ўқув адабиётларнинг янги авлодини яратиш концепцияси/ Каримов А.А., Имамов Э.З., Рузиев К.И., Бутаеров О. – Т.: Шарқ, 2002. – 16 с.
22. Юлдашев У.Ю. Информационные технологии. Ч. 1.– Ташкент: ТГПУ, 2007. – 72 с.
23. Юлдашев У.Ю. Информационные технологии. Ч. 2.– Ташкент: ТГПУ, 2007. – 96 с.
24. Юлдашев У., Закирова Ф. Роль и место учебно-методического комплекса нового поколения в информационно-учебном обеспечении образовательного процесса // Педагогик таълим. – Ташкент, 2004. – № 2. – С. 27–29.
25. www. istedod.uz
26. www.pedagog.uz
27. www.ziyonet.uz
28. www.e-darslik.net

Оглавление

Закирова Феруза Махмудовна
Набиулина Луиза Махмудовна

Современные языки программирования

Данное учебное пособие предназначено для обучения студентов высших педагогических образовательных учреждений направления специальности 5110700 – Методика преподавания информатики по учебной дисциплине «Современные языки программирования».

Рецензенты:
Абдукадиров А. – доктор педагогических наук, профессор,
Мамаражабов М. – кандидат педагогических наук, доцент.

Ташкент
Национальное общество
философов Узбекистана
100000, г. Ташкент, ул. Матбуотчилар, 32.
2012

Редактор В. Сайкина
Компьютерная верстка Б. Абдуназаров
Корректор В. Сайкина

№ лицензии: AI №110, 25.07.2012.
Подписано в печать 26.07.2012. Формат 60x84$^{1}/_{16}$. Печать офсетная.
Гарнитура Times New Roman. Кегль 10,5; 9. Усл. печ. л. 12,0. Уч.-изд. л. 11,5.
Тираж 500 экз. Заказ № 27

Отпечатано в типографии
ООО «START-TRACK PRINT»
Адрес: г. Ташкент, ул. 8 Марта, 57.

ISBN 978-9943-391-44-4

НАЦИОНАЛЬНОЕ ОБЩЕСТВО ФИЛОСОФОВ УЗБЕКИСТАНА