Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Колледж электроники и бизнеса

Н.А. Уйманова, М.Г. Таспаева

ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Часть I

Рекомендовано к изданию Редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Оренбургский государственный университет» в качестве методических указаний для студентов, обучающихся по программам среднего профессионального образования по специальности 09.02.03 Программирование в компьютерных системах

Оренбург
2015

УДК 681.3.06:004.4(075.3)
ББК 32.973 Я73
У 35

Рецензент – кандидат педагогических наук, доцент А.Е.Шухман

Уйманова, Н.А.

У 35 Основы объектно-ориентированного программирования : методические указания к лабораторным работам: в 3 ч. Часть 1/ Н.А. Уйманова, М.Г. Таспаева; Оренбургский гос. ун-т. – Оренбург : ОГУ, 2015. – 59 с.

Методические указания предназначены для выполнения лабораторных работ, обеспечивающих учебный процесс по дисциплине «Основы объектно-ориентированного программирования», студентами 3 курса очной формы обучения специальности 09.02.03 «Программирование в компьютерных системах».

Методические указания составлены с учетом федерального государственного образовательного стандарта среднего профессионального образования по направлению подготовки дипломированных специалистов, утвержденного приказом № 804 от 28 июля 2014 года Министерством образования и науки Российской федерации.

УДК 681.3.06:004.4(075.3)
ББК 32.973 Я73

Содержание

Введение.....5
1 Лабораторная работа №1. Создание простейшего класса объектов. Простое наследование. Родительские и дочерние классы.....7
1.1 Цель работы.....7
1.2 Ход работы.....7
1.3 Содержание отчета.....9
1.4 Контрольные вопросы.....9
1.5 Методические рекомендации.....9
1.5.1 Объектный тип данных. Описание класса.....9
1.5.2 Графические возможности Delphi. Свойство Canvas.....11
1.5.3 Создание простого класса фигур.....17
1.6 Варианты индивидуальных заданий.....20
2 Лабораторная работа №2. «Delphi – среда визуального программирования».....21
2.1 Цель работы.....21
2.2 Ход работы.....21
2.3 Содержание отчета.....23
2.4 Контрольные вопросы.....23
2.5 Методические рекомендации.....23
2.5.1 Создание простейшего проекта.....23
2.5.2. Работа со свойствами формы. Программный доступ к свойствам формы.....29
2.5.3 Организации реакции на события формы.....32
2.5.4 Создание заставки к приложению Delphi.....36
2.5.5 Настройка опций проекта. Сохранение формы в репозитории Delphi. Отладка проекта.....37
2.6 Варианты индивидуальных заданий.....41
2.7 Дополнительная литература.....43
3 Лабораторная работа №3. Прямая работа с файлами в среде Delphi.....43
3.1 Цель работы.....43

3.2 Ход работы.....43
3.3 Содержание отчета.....45
3.4 Контрольные вопросы.....45
3.5 Методические рекомендации.....46
3.5.1 Открытие файла.....47
3.5.2 Чтение и запись данных файла.....48
3.5.3 Ошибки открытия файла.....49
3.5.4 Запись чисел с формы Delphi в текстовый файл.....50
3.5.5 Чтение чисел из текстового файла, подсчет среднего арифметического значения 52
3.6 Варианты индивидуальных заданий.....53
Список использованных источников.....59

Введение

Учебная дисциплина «Основы объектно-ориентированного программирования» является дисциплиной из вариативной части ОПОП, обуславливающей знания для профессиональной деятельности выпускника.

Целью данного курса является формирование у будущего специалиста умений и навыков профессиональной направленности, написания программных продуктов, базирующихся на принципах объектно-ориентированного программирования, а так же сформировать представление о работе в среде визуального редактора.

У студентов необходимо сформировать такие умения и навыки, чтобы они могли в дальнейшем эффективно их использовать в своей профессиональной деятельности при программировании программных продуктов. Будущий специалист должен овладеть, прежде всего, базовыми принципами написания объектно-ориентированной программы, уметь работать в среде визуального программирования Delphi.

Задачами курса является:

- изучение основных понятий объектно-ориентированного программирования, их программное описание;
- изучение принципов объектно-ориентированного программирования;
- практическое освоение структуры написания классов с целью дальнейшего применения в профессиональной деятельности;
- изучение приемов организации распределения памяти для экземпляров класса;
- изучение технологии описания и применения виртуальных методов;
- выработка умений написания программных продуктов в среде визуального программирования Delphi.

Методические указания к лабораторным работам (часть 1) предназначены для проведения лабораторных занятий по дисциплине «Основы объектно-ориентированного программирования» по темам «Основные понятия объектно-ориентированного программирования», «Знакомство со средой Delphi» и «Прямая работа с файлами в

среде Delphi». Рассмотренные лабораторные работы позволят студентам понять практические основы проектирования, используя объектно-ориентированный синтаксис программного средства. А так же изучить основные принципы работы в среде программирования Delphi, что послужит базой для написания более сложных с точки зрения программирования и проектирования интерфейса программных продуктов.

1 Лабораторная работа №1. Создание простейшего класса объектов. Простое наследование. Родительские и дочерние классы

1.1 Цель работы

Научиться описывать классы объектов, организовывать механизм простого наследования.

1.2 Ход работы

Задание общего уровня:

1 Изучить методические рекомендации к лабораторной работе;

2 Создать проект в среде визуального программирования Delphi (запустив приложение Delphi или выполнить пункт меню в открытом окне среды File->New->Application);

3 В секции interface программного модуля unit unit1 описать класс Tfigura ниже описания типа Tform1 (см.методические рекомендации к лабораторной работе);

4 Аналогичным образом описать дочерний класс Tline;

5 В разделе объявления переменных var, описать объект типа Tline (см.методические рекомендации);

6 В секции implementation написать реализацию методов Draw, описанного в классе Tfigura и метода Resize, описанного в классе Tline (см. методические рекомендации);

7 Поместить на форму объект Button1;

8 Изменить свойство Caption данного объекта на «Нарисовать фигуру»;

9 Организовать для объекта Button1 событие OnClick;

10 Внутри образовавшегося метода осуществить вызов методов Draw и Resize (см. методические рекомендации);

11 В разделе var объявить переменные xn,yn типа integer и переменную cl типа Tcolor;

12 Сохранить проект в директории на диске с именем Lab1_1 (File->Save Project as…), тип фалов при сохранении не менять;

13 Провести отладку и компиляцию проекта (клавиша F9);

14 Создать проект в среде визуального программирования Delphi, реализовать описание класса согласно индивидуального задания (создать единственный класс объектов с именем фигуры, например для объекта окружность класс Tcircle. Класс содержит один из двух методов, указанных в индивидуальном задании);

15 Организовать реализацию данного метода в секции implementation;

16 Сохранить проект на диске в директории Lab1_2;

17 Провести отладку и компиляцию проекта;

18 Оформить отчет о проделанной работе;

Задание повышенного уровня:

19 Для своего индивидуального задания разработать родительский класс Tfigura;

20 Реализовать механизм простого наследования, где родительский класс имеет имя Tfigura, дочерний класс – класс описывающий объект индивидуального задания;

21 Выполнить порождение 10 объектов описанного дочернего класса (по щелчку на кнопку осуществить прорисовку объекта с измененными свойствами, согласно методов индивидуального задания. Например, методы изменение цвета контура и положения объекта по горизонтали – объект каждый раз меняет свое положение по оси X, изменяя при этом цвет контура);

22 Сохранить проект;

23 Выполнить отладку и компиляцию проекта;

24 Оформить отчет о проделанной работе;

25 Защитить работу.

1.3 Содержание отчета

1 Цель, ход работы;

2 Постановка задачи №1, листинг программного модуля, результат работы приложения;

3 Формулировка индивидуального задания;

4 Листинг программного модуля индивидуального задания, результат работы приложения;

5 Вывод.

1.4 Контрольные вопросы

1 Что называется объектно-ориентированным программированием? В чем основное отличие объектно-ориентированной программы от модульной или линейной?

2 В чем состоит отличие типа «объект» от типа «запись»?

3 Что называется объектом, свойством, методом, классом? Приведите примеры программного обращения к указанным понятиям.

4 Что такое инкапсуляция? Приведите пример описания объектного типа.

1.5 Методические рекомендации

1.5.1 Объектный тип данных. Описание класса

Тип объект (*Object*) можно рассматривать как усовершенствование типа запись (*Record*): описание свойств, параметры моделируемой сущности (они занимают поля объекта) дополняются методами – описаниями действий с объектом. Собственно, в описании объектного типа дают лишь заголовки соответствующих блоков, а сами блоки приводят ниже. Имеется и еще одно чисто формальное отличие от описания

записи: вместо *Record* используется слово *Object*. Переменные объектного типа называются экземплярами объекта. В отличие от типа «запись» объектный тип содержит не только поля, описывающие данные, но также процедуры и функции, описания которых содержаться в описании объекта. Эти процедуры и функции и являются методами. Методам объекта доступны его поля. Методы и их параметры определяются в описании объекта, а их реализация дается вне этого описания, в том месте программы, которое предшествует вызову данного метода. В описании объекта содержатся лишь шаблоны обращений к методам, которые необходимы компилятору для проверки соответствия количества параметров и их типов при обращении к методам.

Описание объектного типа осуществляется в секции *interface* программного модуля (рисунок 1).

```
unit Unit1;

interface
...
type
TFigura=class
SFigur:real;                    свойство объекта
procedure Draw (c:TColor);      метод объекта
```

Рисунок 1 – Описание объектного типа Tfigura

Реализация описанных в объектном типе методов происходит в секции *implementation* программного модуля. Имя метода является составным: вначале указывается тип объекта, а затем сам метод.

```
....
Implementation
Procedure Tfigura.Draw(c:Tcolor);
```

Begin

// тело метода;

End;

1.5.2 Графические возможности Delphi. Свойство Canvas

Работа с графикой в Delphi предполагает обращение к свойству *Canvas* компонентов. Для программиста *Canvas* в Delphi – это холст (канва), который дает доступ к каждому пикселю. Конечно, рисовать попиксельно для работы с графикой в Delphi не приходится, система Delphi предоставляет мощные средства работы с графикой, облегчающие задачу программиста.

В работе с графикой в Delphi в распоряжении находится канва (свойство *Canvas* Delphi компонентов), кисть(свойство *Brush*), карандаш(свойство *Pen*) компонента, на котором планируется рисовать. У *Pen* и *Brush* есть свойство *Color*(цвет) и *Style*(стиль). Чтоб получить доступ к шрифтам, предоставлено свойство холста(*Canvas*) *Font*.

Таблица 1 – Свойства объекта *Pen* (Карандаш)

Свойство	Описание
Color	Цвет линии
Width	Толщина линии (задается в пикселах)
Style	Вид линии: psSolid — сплошная; psDash — пунктирная, длинные штрихи; psDot – пунктирная, короткие штрихи; psDashDot — пунктирная, с чередованием длинного и короткого штрихов; psDashDotDot — пунктирная с чередованием одного длинного и двух коротких штрихов; psClear — линия не отображается

Таблица 2 – Свойства объекта *Brush*(Кисть)

Свойство	Описание
Color	Цвет закраски замкнутой области.
Style	Стиль закраски области:— сплошная заливка; штриховка: bsHorizontal—горизонтальная; bsVertical—вертикальная; bsFDiagonal — диагональная с наклоном линии вперед; bsBDiagonal — диагональная с наклоном линии назад; bsCross—диагональная клетка.

Таблица 3 – Основные свойства класса *Tfont*

Свойство	Описание
Name	Шрифт, который используется для отображения текста. В качестве значения следует брать название шрифта, например Arial.
Size	Размер шрифта
Style	Стиль начертания символов. Задается с помощью констант: fsBold(полужирный), fsItalic(курсив), fsUnderline(подчеркнутый), fsStrikeOut(перечеркнутый). Свойство Style является множеством, что позволяет комбинировать необходимые стили. Например, инструкция, которая устанавливает стиль «полужирный курсив», выглядит так: Font.Style := [fsBold, fsItalic]
Color	Цвет символов. В качестве значения можно использовать константу типа Tcolor.

Не все компоненты в Delphi имеют эти свойства. Свойство *Canvas* имеют, например, такие компоненты как *ListBox*, *ComboBox*, *StringGrid*, а также и сама форма, которая размещает компоненты.

Холст для рисования *Canvas*, представляет собой перевернутую систему координат XOY, где O эта верхняя левая точка компонента или формы.

Таблица 4 – Константы Tcolor

Цвет	Константа
Бирюзовый	clAqua
Черный	clBlack
Синий	clBlue
Ярко-розовый	clFuchsia
Зеленый	clGreen
Салатовый	cLime
Каштановый	clMaroon
Темно-синий	clNavy
Оливковый	clOlive
Фиолетовый	clPurple
Красный	clRed
Серебристый	clSilver
Зелено-голубой	clTeal
Белый	clWhite

Основное свойство такого объекта как Canvas Delphi – Pixels[i,j] типа *Tcolor*, то есть это двумерный массив точек (пикселов), задаваемых своим цветом.

Рисование на канве происходит в момент присвоения какой-либо точке канвы заданного цвета. Каждому пикселю может быть присвоен любой доступный для Windows цвет.

Form1.Canvas.Pixels[100, 100] := clRed;

приведёт к рисованию красной точки с координатами [100, 100].

Узнать цвет пиксела можно обратным присвоением:

Color :=Form1.Canvas.Pixels[100,100];

Таблица 5 – Свойства класса *Tcanvas*

Свойство	Описание
Pen	Карандаш. Определяет цвет, стиль и толщинц линии, которую рисует, например метод Lineto
PenPos	Положение (координаты) карандаша
Brush	Кисть. Определяет цвет и стиль закраски области, например

	прямоугольника, который рисует метод Rectangle.
Font	Шрифт. Определяет шрифт, который используется для вывода текста, например методом TextOut.

Таблица 6 – Методы класса *Tcanvas*

Метод	Действие
1	2
Lineto(x, y)	Рисует линию из текущей точки в точку с указанными координатами (перемещение указателя текущей точки в нужную обеспечивает метод Moveto). Цвет линии определяется свойством Pen.Color .
Rectangle(x1 ,y1 ,x2, y2)	Рисует прямоугольник. Параметры x1, y1указывают координаты верхней левой точки, ax2, y2 координаты нижней правой точки. Цвет границы прямоугольника определяет свойство Pen.Color , цвет закраски области — свойствоBrush.Color .

Продолжение таблицы 6

1	2
RoundRect(x1, y1, x2, y2, x3, y3)	Рисует прямоугольник со скругленными углами. Параметры x1, y1 указывают координаты верхней левой точки, x2, y2 координаты нижней правой точки, а x3, y3 радиус скругления. Цвет границы прямоугольника определяет свойство Pen.Color ,цвет закраски области — свойство Brush.Color .
Ellipse(x1, y1, x2, y2)	Рисует эллипс (окружность). Параметры x1, y1 указывают координаты верхней левой точки, ax2, y2 координаты нижней правой точки прямоугольника в который вписана окружность. Цвет границы прямоугольника определяет свойство Pen.Color , цвет закраски области — свойство Brush.Color .
Arc(x1, y1, x2, y2, x3, y3,x4, y4)	Рисует дугу. Параметры x1, y1, x2 и y2 задают эллипс, частью которого является дуга, параметры x3, y3, x4 и y4 — начальную и конечную точку дуги. Цвет дуги определяет свойство Pen.Color.
Pie(x1 ,y1 ,x2 ,y2 ,x3 ,y3 ,x4 ,y4)	Рисует сектор. Параметры x1, y1, x2 и y2 задают эллипс, частью которого является сектор, параметры x3, y3, x4 и y4 —границы сектора. Цвет границы сектора определяет свойствоPen.Color , цвет закраски сектора — свойствоBrush.Color .
FillRect(aRect)	Рисует закрашенный прямоугольник. Параметр aRect(тип Trect) определяет положение и размер прямоугольника. Цвет закраски области определяет свойство Brush.Color .

Продолжение таблицы 6

1	2
FrameRect(aRect)	Рисует контур прямоугольника. Параметра Rect (тип Trect) определяет положение и размер прямоугольника. Цвет контура определяет свойство Brush.Color .
Polyline(points, n)	Рисует ломаную линию. Points — массив типаTPoint. Каждый элемент массива представляет собой запись, поля x и y, которые содержат координаты точки перегиба ломаной. N —количество звеньев ломаной. Метод Polyline вычерчивает ломаную линию, последовательно соединяя прямые точки, координаты которых находятся в массиве: первую со второй, вторую с третьей, третью с четвертой и т.д.

Вычерчивание графических элементов на графической поверхности, например компонента *Image* (закладка *Additional*), выполняют соответствующие методы класса *Tcanvas*.

Инструкция, обеспечивающая вычерчивание графического элемента, в общем виде выглядит так:

Объект.Canvas.*Метод (Параметры)*;

Объект определяет объект, на поверхности которого нужно нарисовать графический элемент. В качестве объекта можно указать компонент *Image.*

Метод – это имя метода, который обеспечивает рисование нужного графического элемента.

Параметры, в большинстве случаев, определяют положение графического элемента на графической поверхности и его размер.

Например, в результате выполнения инструкции

Form1.Canvas.Rectangle(10,20,60,40);

В поле компонента *Form1* будет нарисован прямоугольник шириной 50 и

высотой 20 пикселей, левый верхний угол которого будет находиться в точке(10,20).

При записи инструкций, обеспечивающих вывод графики, удобно использовать инструкцию *with,* которая позволяет сократить количество набираемого кода.

Вывод строки текста на графическую поверхность объекта обеспечивает метод *TextOut*. Инструкция вызова метода *TextOut* в общем виде выглядит следующим образом:

Объект.Canvas.TextOut(x, y, '*Текст*');

Параметры *x, y* определяют координаты точки графической поверхности, от которой выполняется вывод текста.

Шрифт, используемый для отображения текста, определяет свойство Font графической поверхности, на которую текст выводится. Свойство *Font* представляет собой объект типа Tfont.

1.5.3 Создание простого класса фигур

Для начала работы над проектом необходимо запустить среду программирования Delphi.

Внутри программного модуля описать объектный тип данных Tfigura

```
type
Tfigura=class
procedure Draw (c:Tcolor);
end;
```

Организовать дочерний тип данных

```
Tline=class(Tfigura)
procedure Resize(xx,yy:integer);
end;
```

В глобальном разделе переменных объявить объект класса Tline

```
var
```

Form1: Tform1;

L:Tline;

В секции implementation организовать реализацию описанных методов

implementation

*{$R *.dfm}*

procedureTFigura.Draw(c:Tcolor);

begin

form1.canvas.pen.color:=c;

end;

procedureTLine.Resize(xx,yy:integer);

begin

form1.canvas.LineTo(xx,yy);

end;

Окно приложения Delphi состоит из нескольких окон. В окно формы необходимо поместить со вкладки *Standart* объект *Button1* (рисунок 2).

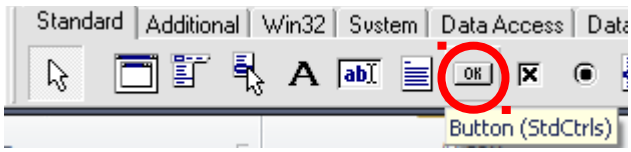

Рисунок 2 – Вкладка *Standart*

Для объекта *Button1*изменить свойство *Caption* на «*Нарисовать объект*»

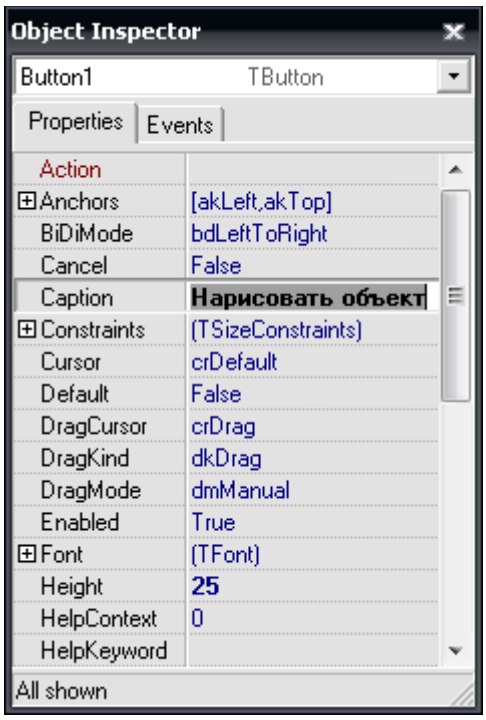

Рисунок 3 – Свойства объекта Button1

Для объекта Button1 организовать событие OnClick, прописать следующий код:

```
procedure Tform1.Button1Click(Sender: Tobject);
begin
xn:=40;
yn:=50;
cl:=clRed;
L.Draw(cl);
L.Resize(xn,yn);
end;
```

1.6 Варианты индивидуальных заданий

1 Класс объектов «Окружность». Методы изменение цвета контура и радиуса;

2 Класс объектов «Квадрат». Методы рисование квадрата, изменение его положения (по горизонтали);

3 Класс объектов «Прямоугольник». Методы изменение цвета контура и размера;

4 Класс объектов «Скругленный прямоугольник». Методы изменение размера и контура фигуры;

5 Класс объектов «Окружность». Методы изменение цвета заливки и положения (по вертикали);

6 Класс объектов «Прямоугольник». Методы изменение стиля заливки и положения (по горизонтали);

7 Класс объектов «Квадрат». Методы изменение стиля заливки и цвета контура;

8 Класс объектов «Сектор». Методы Изменение цвета контура и стиля заливки;

9 Класс объектов «Скругленный прямоугольник». Методы изменение размеров и цвета контура;

10 Класс объектов «Сектор». Методы изменение размеров и цвета заливки;

11 Класс объектов «Окружность». Методы изменение положения и стиля контура;

12 Класс объектов «Овал». Методы изменение горизонтального радиуса и цвета заливки;

13 Класс объектов «Квадрат». Методы изменение размера и цвета заливки;

14 Класс объектов «Овал». Методы изменение положения (по горизонтали) и изменение стиля контура;

15 Класс объектов «Прямоугольник». Методы изменение положения (по вертикали) и цвета заливки.

2 Лабораторная работа №2. «Delphi – среда визуального программирования»

2.1 Цель работы

Познакомиться с основными элементами визуальной среды программирования Delphi, овладеть приемами работы с формами.

2.2 Ход работы

Задание общего уровня:

1 Изучить методические рекомендации к лабораторной работе;

2 Задание 1

- создать проект в среде визуального программирования Delphi, демонстрирующий простейшие возможности среды (внешний интерфейс формы проекта представлен на рисунке 5);
- сохранить проект в директории на диске с именем Lab2_1;
- провести отладку и компиляцию проекта;
- оформить отчет о проделанной работе;

3 Задание 2

- создать проект в среде визуального программирования Delphi, демонстрирующий операцию сложения двух чисел (внешний интерфейс формы проекта представлен на рисунке 7);
- сохранить проект на диске в директории Lab2_2;
- провести отладку и компиляцию проекта;

4 Задание 3

- создать проект в среде визуального программирования Delphi, демонстрирующий программный доступ к свойствам формы, (внешний интерфейс формы проекта представлен на рисунке 8);
- сохранить проект в директории на диске с именем Lab2_3;
- провести отладку и компиляцию проекта;
- оформить отчет о проделанной работе;

5 Задание 4

– создать проект в среде визуального программирования Delphi, демонстрирующий реакцию на события формы (на форме разместить компонент Label);

– сохранить проект в директории на диске с именем Lab2_4;

– провести отладку и компиляцию проекта;

– оформить отчет о проделанной работе;

6 Задание 5

– создать заставку к существующему проекту согласно методическим рекомендациям п. 2.5.4;

– провести отладку и компиляцию проекта;

7 Задание 6

– создать иконку к существующему проекту согласно методическим рекомендациям, указанных в п.2.5.5;

– провести отладку и компиляцию проекта;

8 Задание 7. Добавить в репозиторий Delphi ранее созданную форму в соответствии с методическими рекомендациями п. 2.5.5;

9 Оформить отчет о проделанной работе;

Задание повышенного уровня:

1 Создать проект в среде визуального программирования Delphi, демонстрирующий программный доступ к свойствам формы, реализовать реакцию на события формы согласно своего индивидуального задания;

2 Сохранить проект;

3 Провести отладку и компиляцию проекта;

4 Оформить отчет о проделанной работе;

5 Защитить работу.

2.3 Содержание отчета

1 Цель, ход работы;

2 Постановка задачи заданий 1-7, листинг программного модуля, результат работы приложения;

3 Формулировка индивидуального задания;

4 Листинг программного модуля индивидуального задания, результат работы приложения;

5 Вывод.

2.4 Контрольные вопросы

1 Какие элементы включает интерфейс среды разработки Delphi?

2 Какие файлы входят в состав проекта Delphi?

3 Перечислите основные свойства формы. Как программно изменить свойства?

4 Как настроить опции проекта?

5 Что такое репозиторий Delphi? Как добавить форму в репозиторий?

2.5 Методические рекомендации

2.5.1 Создание простейшего проекта

Основные элементы интерфейса интегрированной среды разработки Delphi представлены на рисунке 4.

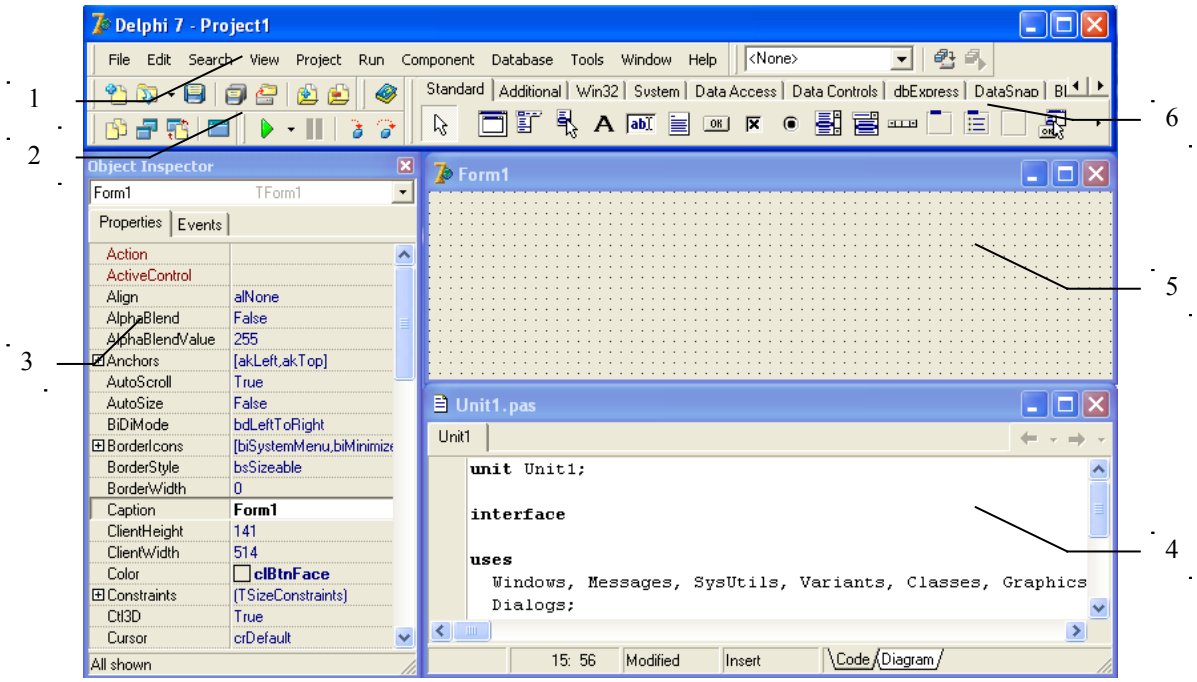

Рисунок 4 – Основные элементы интерфейса

1 – основное меню; 2 – панель инструментов;

3 – окно инспектора объектов; 4 – редактор кода программы;

5- окно формы; 6 – палитра компонентов

Основное меню содержит следующие команды: File, Edit, Search, View, Project, Run, Component, Database, Tools, Window, Help.

Палитра компонентов содержит множество компонентов, которые подразделяются на несколько групп. Каждая группа размещена на своей странице палитры компонентов.

Окно инспектора объектов предназначено для изменения свойств выбранных компонентов и состоит из двух страниц. Страница Properties (Свойства) предназначена для изменения необходимых свойств компонента, страница Events (События) – для определения реакции компонента или формы на то или иное событие (например, щелчок «мыши» на кнопке – событие OnClick, создание формы – OnCreate).

Окно формы представляет собой проект Windows-окна программы. На этом окне в процессе написания программы размещаются необходимые компоненты.

Редактор кода программы предназначен для просмотра, написания и редактирования текста программы. В системе Delphi используется язык программирования Object Pascal.

Программа в среде DELPHI составляется как описание алгоритмов, которые будут выполняться, если возникает определенное событие, связанное с формой или с каким-либо из размещенных на ней компонентов. Для каждого обрабатываемого события, с помощью страницы Events инспектора объектов в тексте программы организуется процедура (procedure), между ключевыми словами begin и end которой программист записывает на языке Object Pascal требуемый алгоритм.

Переключение между окном формы и окном редактора кода осуществляется с помощью клавиши F12.

Задание 1.

1 Создаем новый проект приложения, выбрав команду File-New, а затем Application;

2 Выберем команду *File-Save Project As* и в появившемся окне выберем новую папку, введем имя сохраняемого модуля MyUnit, а в следующем окне – имя сохраняемого проекта, т.е. будущей программы Program1;

3 Создадим простейшее приложение:

Разместим на форме 2 компонента *Label*  вкладки *Standart*. В окне инспектора объектов у объекта *Label1* изменим значение свойства *Caption* с «*Label1*» на *Меня зовут*. У объекта *Label2* удаляем значение свойства *Caption*.

Разместим на форме компонент *Edit*  вкладки *Standart*. В окне инспектора объектов у объекта *Edit1* свойстве Text удалите значение «*Edit1*» (т.е. задаем свойству *Text* значение, равное пустой строке).

Разместим на форме кнопку *Button*  вкладки *Standart*. Изменим свойство *Caption* компонентов *Button1* на *Привет!*

Форма должна принять следующий вид (рисунок 5):

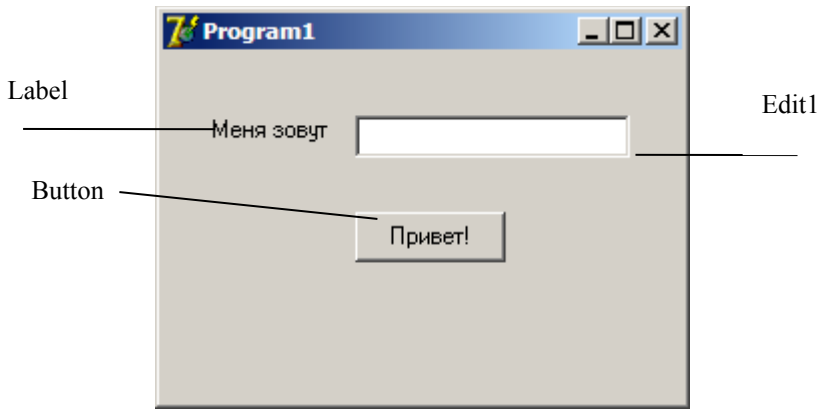

Рисунок 5 – Внешний интерфейс формы

Создадим обработчик события нажатия кнопки. При нажатии на кнопку «Привет!» программа должна вывести в метку *Label2* текст. Для создания обработчика события нажатия этой кнопки нужно выделить компонент Button1, в инспекторе объектов перейти на закладку Events, и дважды щелкнуть в строке напротив названия события **OnClick**. В модуле программы появится заготовка процедуры обработчика события:

```
procedure Tform1.Button1Click(Sender: Tobject);
begin
end;
```

А в описании класса формы добавится поле с заголовком процедуры:

```
procedure Button1Click(Sender: Tobject);
```

Приведем эту процедуру к следующему виду:

```
procedure Tform1.Button1Click(Sender: Tobject);
begin
  Label2.Caption:='Здравствуй, '+Edit1.Text+'!';
end;
```

4 Запустить программу можно выбрав команду *Run* в меню *Run*, или нажав клавишу F9. При этом происходит компиляция и, если нет ошибок, компоновка

программы и создание единого загружаемого файла с расширением .exe. На экране появляется главная форма программы;

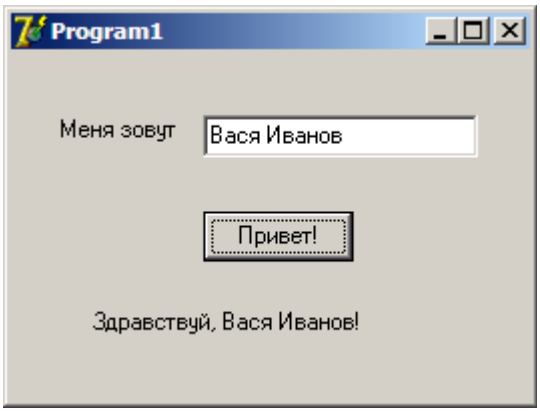

Рисунок 6 – Результат выполнения программы

Задание 2.

1 Создаем новый проект приложения, выбрав команду File-New, а затем Application;

2 Выберем команду *File-Save Project As* и в появившемся окне выберем новую папку, введем имя сохраняемого модуля MyUnit, а в следующем окне – имя сохраняемого проекта, т.е. будущей программы MyProgram;

3 Создадим приложение, которое будет суммировать два числа и выводить ответ в метку Label:

Разместим на форме 2 компонента *Label*  вкладки *Standart*. В окне инспектора объектов у объектов *Label1* и *Label2* изменим значение свойства Caption с «*Label1*» и «*Label2*» на *Число a* и *Число b* соответственно.

Разместим на форме 2 компонента *Edit*  вкладки *Standart*. В окне инспектора объектов у объектов *Edit1* и *Edit2* в свойстве Text удалите значения «*Edit1*» и «*Edit2*» (т.е. задаем свойству *Text* значение, равное пустой строке).

На форме также необходимо разместить кнопку *Button*  вкладки *Standart*. Изменим свойство *Caption* компонентов *Button1* на *Вычислить*.

Для вывода результата сложения двух чисел добавим на форму еще одну метку

Label (значение свойства Caption – *Сумма равна*)

Форма должна принять следующий вид:

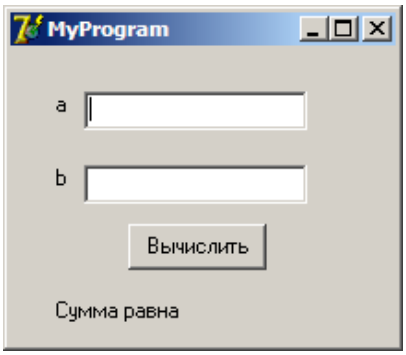

Рисунок 7 – Внешний интерфейс формы проекта

Создадим обработчик события нажатия кнопки. При нажатии на кнопку «Вычислить» программа должна производить необходимые вычисления и выводить результат. Для создания обработчика события нажатия этой кнопки нужно выделить компонент Button1, в инспекторе объектов перейти на закладку Events, и дважды щелкнуть в строке напротив названия события *OnClick*. В модуле программы появится заготовка процедуры обработчика события:

```
procedure Tform1.Button1Click(Sender: Tobject);
begin
end;
```

А в описании класса формы добавится поле с заголовком процедуры:

```
procedure Button1Click(Sender: Tobject);
```

Приведем эту процедуру к следующему виду:

```
procedure Tform1.Button1Click(Sender: Tobject);
  var a,b,c:integer;
begin
  a:=StrToInt(Edit1.Text);
  b:= StrToInt(Edit2.Text);
  c:=a+b;
```

Label3.Caption:=Label3.Caption+IntToStr©;

end;

Здесь после слова *var* перечисляются имена переменных величин и указывается их тип. Мы используем переменные *a, b* и *c*, которые имеют целый тип.

Функция *StrToInt()* выполняет преобразование строки (в данном случае хранящейся в поле *Edit*) в целое число (такое преобразование возможно, если строка состоит только из цифр).

Функция *IntToStr()* выполняет обратное преобразование, т.е. преобразует целое число в строку.

Запустить программу можно выбрав команду *Run* в меню *Run*, или нажав клавишу F9. При этом происходит компиляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется главная форма программы.

2.5.2. Работа со свойствами формы. Программный доступ к свойствам формы

Форма - это компонент Form класса Tform. На ее основе конструируется приложение. Существуют 2 класса форм:

1 Немодальные – те, которые позволяют переключаться в другую форму приложения без своего закрытия;

2 Модальные – те, которые требуют обязательного закрытия перед обращением к другой форме.

Каждое приложение имеет одну главную форму и, возможно, несколько 29ТОростепенных. Главная форма загружается автоматически при запуске приложения.

Каждая форма имеет две области:

1 Клиентская область – та часть формы, в которой размещаются визуальные компоненты;

2 Неклиентская область – занята рамкой, заголовком формы и строкой главного меню.

Ниже перечислены Delphi свойства формы (объекта Tform):

- *Name* – Имя формы. Испльзуется для управления формой и доступа к копонентам формы;
- *Caption* – Текст заголовка;
- *Widht* – Ширина Delphi свойств формы;
- *Height* – Высота формы;
- *Top* – Расстояние от верней части формы до верхней границы экрана;
- *Left* – Расстояние от левой границы формы до левой границы экрана;
- *Visible* – позволяет скрывать и отображать данную форму;
- *Active* – определяет активность формы;
- *Clientwidth* – возвращает ширину клиентской области;
- *Clientheight* – возвращает высоту клиентской области;
- *Windowstate* – определяет состояние отображаемой формы;
- *BorderStyle* – Вид границы окна формы. Она может быть обычной (bsSizeable), тонкой (bsSingle) или отсутствовать (bsNone);
- *BorderIcons* – Кнопки управления окном. Значение их свойств определяет, какие кнопки управления окном будут доступны пользователю во время работы программы;
- *Icon* – Значок в заголовке окна, обозначающего кнопку вывода системного меню;
- *Color* – Цвет фона. Его можно задать, указав название цвета или сделать привязку к цветовой схеме операционной системы;
- *Font* – Шрифт, то есть его можно выбрать из диалогового окна;
- *AlphaBlend* – использовать ли прозрачность формы;
- *AlphaBlendValue* — степень прозрачности формы (0-прозрачна полностью, 255-непрозрачна).

Задание 3

Создать приложение, программно изменяющее свойства формы и имеющее интерфейс, представленный на рисунке 8.

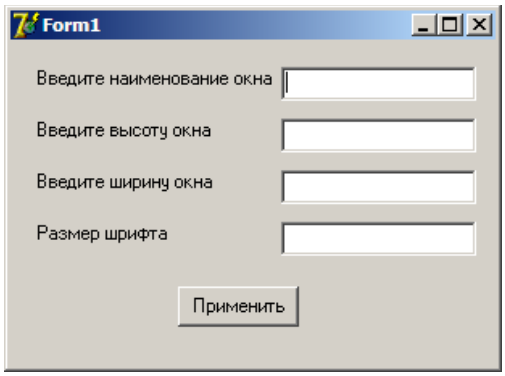

Рисунок 8 – Внешний интерфейс формы проекта

1 Создаем новый проект приложения, выбрав команду File-New, а затем Application;

2 Выберем команду *File-Save Project As*;

3 Размещаем на форме необходимые компоненты;

4 Создадим обработчик события нажатия кнопки. Приведем процедуру обработчика событий к следующему виду:

```
procedure Tform1.Button1Click(Sender: Tobject);
begin
  form1.Caption:=edit1.Text;
  form1.Height:=StrToInt(edit2.Text);
  form1.Width:=StrToInt(edit3.Text);
  form1.Font.Size:=StrToInt(edit4.Text);
end;
```

5 Запустить программу можно выбрав команду *Run* в меню *Run*, или нажав клавишу F9. При этом происходит компиляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется главная форма программы.

2.5.3 Организации реакции на события формы

События формы

- OnCreate — происходит сразу после создания формы. Обработчик этого события может установить начальные значения для свойств формы и ее компонентов, запросить у операционной системы необходимые ресурсы, создать служебные объекты, а также выполнить другие действия прежде, чем пользователь начнет работу с формой. Парным для события OnCreate является событие OnDestroy;
- OnDestroy — происходит непосредственно перед уничтожением формы. Обработчик этого события может освободить ресурсы, разрушить служебные объекты, а также выполнить другие действия прежде, чем объект формы будет разрушен;
- OnShow — происходит непосредственно перед отображением формы на экране. Парным для события OnShow является событие OnHide;
- OnHide — происходит непосредственно перед исчезновением формы с экрана. Парным для события OnHide является событие OnShow;
- OnActivate — происходит, когда пользователь переключается на форму, т.е. форма становится активной. Парным для события OnActivate является событие OnDeactivate;
- OnDeactivate — происходит, когда пользователь переключается на другую форму, т.е. текущая форма становится неактивной. Парным для события OnDeactivate является OnActivate;
- OnCloseQuery — происходит при попытке закрыть форму. Запрос на закрытие формы может исходить от пользователя, который нажал на рамке формы кнопку «Закрыть», или от программы, которая вызвала у формы метод Close. Обработчику события OnCloseQuery передается по ссылке булевский параметр CanClose, разрешающий или запрещающий действительное закрытие формы;
- OnClose — происходит после события OnCloseQuery, непосредственно перед закрытием формы;

- OnContextPopup — происходит при вызове контекстного меню формы;
- OnMouseDown — происходит при нажатии пользователем кнопки мыши, когда указатель мыши наведен на форму. После отпускания кнопки мыши в компоненте происходит событие OnMouseUp. При перемещении указателя мыши над формой периодически возникает событие OnMouseMove, что позволяет отслеживать позицию указателя;
- OnMouseWheelUp — происходит, когда колесико мыши проворачивается вперед (от себя);
- OnMouseWheelDown — происходит, когда колесико мыши проворачивается назад (на себя);
- OnMouseWheel — происходит, когда колесико мыши проворачивается в любую из сторон;
- OnStartDock — происходит, когда пользователь начинает буксировать стыкуемый компонент;
- OnGetSiteInfo — происходит, когда стыкуемый компонент запрашивает место для стыковки;
- OnDockOver — периодически происходит при буксировке стыкуемого компонента над формой;
- OnDockDrop — происходит при стыковке компонента;
- OnEndDock — происходит по окончании стыковки компонента;
- OnUnDock — происходит, когда пользователь пытается отстыковать компонент.
- OnDragDrop — происходит, когда пользователь опускает в форму буксируемый объект;
- OnDragOver — периодически происходит при буксировке объекта над формой;
- OnCanResize — происходит при попытке изменить размеры формы. Запрос на изменение размеров может исходить от пользователя. Обработчику события OnCanResize передается по ссылке булевский параметр Resize, разрешающий или запрещающий действительное изменение размеров формы;

- OnResize — происходит при изменении размеров формы;
- OnConstrainedResize — происходит при изменении размеров формы и позволяет на лету изменять минимальные и максимальные размеры формы;
- OnShortCut — происходит, когда пользователь нажимает клавишу на клавиатуре. Позволяет перехватывать нажатия клавиш еще до того, как они дойдут до стандартного обработчика формы;
- OnKeyDown – генерируется, когда нажата клавиша на клавиатуре;
- OnKeyPress – генерируется, когда нажата и отпущена клавиша на клавиатуре;
- OnKeyUp – генерируется, когда отпущена клавиша на клавиатуре.

Задание 4

Создать приложение, демонстрирующее реакцию на события формы.

1 Создаем новый проект приложения, выбрав команду File-New, а затем Application;

2 Выберем команду *File-Save Project As*;

3 Создадим обработчик события нажатия клавиши клавиатуры. Приведем процедуру обработчика событий к следующему виду:

```
procedure Tform1.FormKeyPress(Sender: Tobject; var Key: Char);
begin
  Label1.Caption:=’Вы нажали клавишу!’;
end;
```

4 Создадим обработчик события, происходящее при нажатии и отпускании кнопки мыши. Например, нарисуем прямоугольник координаты левого верхнего угла которого будут соответствовать точке нажатия мышки, а координатам левого верхнего угла точка, в которой пользователь отпустил кнопку. Приведем процедуру обработчика событий к следующему виду:

```
var
 Form1: Tform1;
 xx,yy:integer;
...
```

```
procedure Tform1.FormMouseDown(Sender: Tobject; Button: TmouseButton;
  Shift: TshiftState; X, Y: Integer);
begin
   xx:=x;
  yy:=y;
end;

procedure Tform1.FormMouseUp(Sender: Tobject; Button: TmouseButton;
  Shift: TshiftState; X, Y: Integer);
begin
   Form1.Canvas.Rectangle(XX,YY,X,Y);
end;
```

5 Запустить программу можно выбрав команду *Run* в меню *Run*, или нажав клавишу F9. При этом происходит компиляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется главная форма программы (рисунок 9).

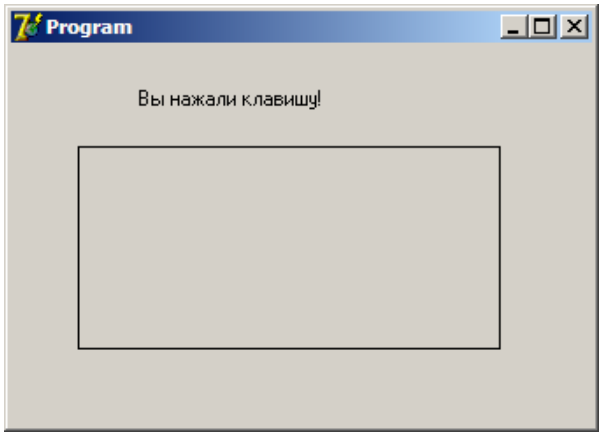

Рисунок 9 – Результат работы приложения

2.5.4 Создание заставки к приложению Delphi

Для создания заставки к приложению Delphi выполняем следующую последовательность действий, описанных в задании 5.

Задание 5.

1 Добавляем в существующий проект форму (*File – New Form*). Это окно и будет заставкой. У него нужно убрать рамку с полосой заголовка, установив свойство *BorderStyle* в *bsNone*;

2 Разрабатываем дизайн окна заставки;

3 Выполняем команду меню *Project- Options*. Во вкладке *Forms* форму *Form2* из списка автоматически создаваемых форм (*Auto-Create forms*) перенести в список доступных форм (*Available forms*);

4 На форме-заставке размещаем компонент *Timer* с вкладки *System*. В его свойстве *Interval* установить значение 5000, а в событии *OnTimer* написать:

Timer1.Enabled := false;

Это сделано для того, чтобы заставка была видна в период указанного времени – 5000 миллисекунд, т.е. 5 секунд;

5 Далее открываем файл проекта: *Project > View Source* и вносим изменения согласно примера ниже:

```
program Project1;
uses
Forms,
Unit1 in 'Unit1.pas'{Form1},
Unit2 in 'Unit2.pas'{IntroForm};
{$R *.res}
begin
Application.Initialize;
  Form2 := Tform2.Create(Application);
  Form2.Show;
  Form2.Update;
```

```
  while Form2.Timer1.Enabled do
    Application.ProcessMessages;
  Application.CreateForm(Tform1, Form1);
  Form2.Hide;
  Form2.Free;
  Application.Run;
end.
```

2.5.5 Настройка опций проекта. Сохранение формы в репозитории Delphi. Отладка проекта

Для установки параметров проекта используется окно параметров проекта (*Project Options*), открываемое командой *Project* → *Options* (Проект → Параметры).

Параметры разбиты на группы, каждая из которых располагается в окне параметров проекта на своей странице.

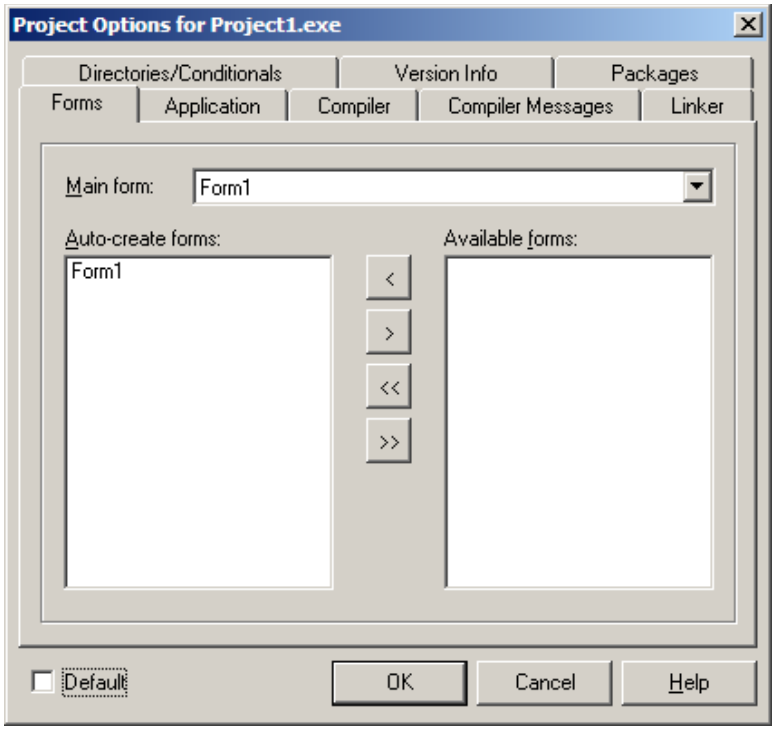

Рисунок 10 – Окно опций проекта

На вкладке *Forms* можно задать главную форму приложения (*Main form*) и в списке *Auto-create forms* указать формы, которые будут создаваться одновременно с главной формой.

На вкладке *Application* можно задать название (*Title*) программы. В среде Delphi дополнительно можно задать файл справки (*Help file*) и значок (*Icon*).

На вкладке *Compiler* настраиваются параметры компилятора. Наиболее интересными из них являются переключатели *Optimization* (включает оптимизацию генерируемого кода) и *Use Debug DCUs* (позволяет отлаживать исходный код системных библиотек). Оба этих переключателя полезны при отладке программы: первый следует выключить, а второй – включить.

На вкладке *Compiler Messages* настраивается чувствительность компилятора к подозрительному коду. Включив переключатели *Show hints* и *Show warnings*, можно получать от компилятора весьма полезные подсказки (*hints*) и предупреждения (*warnings*), а не только сообщения об ошибках.

На вкладке *Linker* настраиваются параметры сборки проекта. Следует обратить внимание на группу *Memory sizes*, особенно на два параметра: *Min stack size* и *Max stack size*. Они задают соответственно минимальный и максимальный размеры стека прикладной программы, т.к. может потребоваться увеличить значения этих параметров при написании приложения, активно использующего рекурсивные подпрограммы.

На вкладке *Directories/Conditionals* можно задать каталоги для различных файлов. Наиболее важные из них: *Output directory* – каталог, в который помещается выполняемый файл; *Unit output directory* – каталог, в который помещаются промежуточные объектные модули (DCU-файлы); *Search path* – список каталогов, в которых осуществляется поиск программных модулей.

На вкладке *Version Info* выводится информация о версии приложения. Для того чтобы эта информация помещалась в выполняемый файл, нужно включить переключатель *Include version information in project*.

На вкладке *Packages* можно управлять списком пакетов, используемых в проекте. Пакеты – это внешние библиотеки компонентов. Переключатель *Build with*

runtime packages позволяет существенно уменьшить размер выполняемого файла за счет использования внешних библиотек компонентов вместо встраивания их кода непосредственно в выполняемый файл. Этот режим выгоден при создании нескольких программ, построенных на базе большого количества общих компонентов.

Задание 6.

Создать иконку для существующего приложения Delphi. Для этого необходимо:

1 Запускаем редактор изображений *Image Editor* (меню *Tools- Image Editor*);

2 Чтобы начать работу по созданию нового значка, нужно из меню *File* выбрать команду *New*, а из появившегося списка — опцию *Icon File;*

3 После выбора типа создаваемого файла открывается окно *Icon Properties*, в котором необходимо выбрать характеристики создаваемого значка: *size* (размер) — 32x32 (стандартный размер значков Windows) и *Colors* (*Палитра*) — 16 цветов. В результате нажатия кнопки *OK* открывается окно *Icon1.ico*, в котором можно, используя стандартные инструменты и палитру, нарисовать нужный значок;

4 Сохраняем созданный значок (*File-Save*);

5 В среде Delphi выполняем команду *Project → Options*. Во вкладке *Application* по кнопке *Load Icon* загружаем файл созданного значка.

Задание 7.

Окно репозитория Delphi открывается при выборе *File – New – Other* и имеет вид, показанный на рисунке 2.

На его страницах расположены пиктограммы для выбора прототипов форм, модулей, проектов и экспертов построения форм и проектов. Зависимые переключатели Copy, Inherit и Use определяют режим связи между хранящимся в репозитории прототипом и его копией в проекте: *Copy* – выбранный элемент копируется в текущий каталог и автоматически подключается к проекту; между образцом и его копией нет никакой связи; *inherit* – в проекте создаются наследники выбранного элемента и всех его родителей; любые изменения образца проявляются во всех проектах, которые его унаследовали; изменения наследника не влияют на

образец; *Use* – выбранный элемент становится частью проекта; любые его изменения в проекте приводят к изменениям образца и сказываются во всех других проектах, которые его унаследовали или используют.

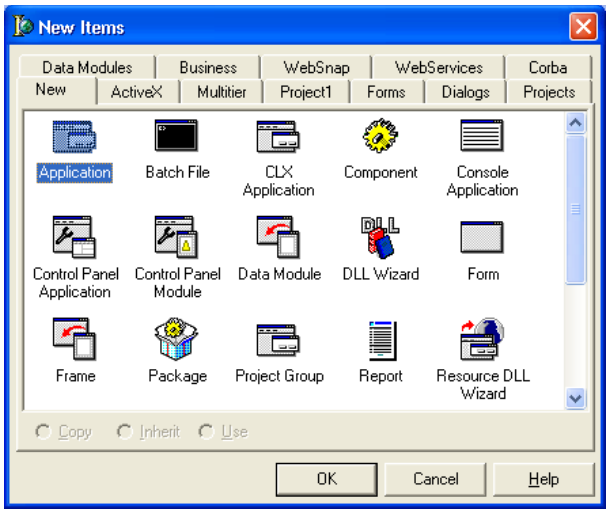

Рисунок 11 – Окно репозитория Delphi

Ниже кратко описывается методика размещения в репозитории разработанной программистом формы.

1 Разработаем форму. Разместим на ней все необходимые интерфейсные компоненты. Сохраним форму;

2 Щелкните по форме правой кнопкой мыши и в локальном меню выберите Add to Repository. Появится диалоговое окно регистрации формы (рисунок 12);

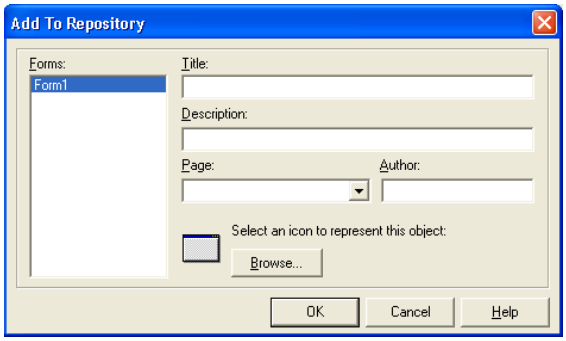

Рисунок 12 – Окно репозитория Delphi

3 В строке *Title* напишем имя, под которым форма будет зарегистрирована в репозитории (*WorkForm)*, в строке *Description* – описание формы, (*Основная форма для проекта MyProgram)*, в строке *Author* – свое имя. Раскроем список *Page* и выберем закладку страницы репозитория *(Forms)*, на которой разместим форму, после чего щелкнем по кнопке *OK* – форма помещена в репозитории.

2.6 Варианты индивидуальных заданий

1 Создать приложение, на форме которого расположить два текстовых поля, в них задать координаты вывода формы; при двойном нажатии левой кнопки мыши по форме окно выводится в заданном месте;

2 Создать приложение, обработать событие *OnShow,* при котором форма плавно увеличивает ширину до заданного значения. На форме разместить кнопку «Закрыть», при нажатии на которую форма закрывается (воспользоваться циклом for…to…do);

3 Создать приложение, осуществляющее вывод координат указателя мыши в надпись при движении мыши;

4 Создать приложение, при закрытии которого форма становится полупрозрачной и постепенно угасает, закрывая приложение (воспользоваться циклом for…downto…do);

5 Создать приложение. При нажатии правой кнопкой мыши в заголовке формы появляются координаты указателя мыши, при отпускании кнопки мыши заголовок формы очищается;

6 Создать приложение, в котором при изменении размеров формы в заголовок выводятся текущие высота и ширина;

7 Создать приложение. При нажатии клавиши управления курсором на клавиатуре форма смещается в левую, правую сторону, вверх и вниз на заданное расстояние;

8 Создать приложение. При нажатии левой кнопки мыши цвет формы меняется случайным образом. На форме разместить кнопку «Закрыть», при нажатии на которую появляется окно сообщения о подтверждении закрытии формы;

9 Создать приложение. Разместить на форме надпись. При нажатии на клавиатуре стрелки «вверх» размер шрифта надписи увеличивается, при нажатии стрелки «вниз» размер шрифта надписи уменьшается;

10 Создать приложение. Разместить на форме надпись. При нажатии левой кнопки мыши надпись становится невидимой, при отпускании мыши надпись становится видимой;

11 Создать приложение. При двойном щелчке левой кнопки мыши по форме меняется тип рамки формы. Перебрать все возможные варианты обрамления формы;

12 Создать приложение, в котором когда форма активна ее цвет выбирается случайным образом из 256 оттенков красного, при деактивации цвет формы выбирается случайным образом из 256 оттенков зеленого. Чтобы проверить активность формы необходимо в приложении создать еще одну форму (File->New->Form);

13 Создать приложение, в котором при прокручивании колесика мыши вперед размеры формы уменьшались, при прокручивании колесика мыши назад размеры формы увеличивались;

14 Создать приложение. При нажатии левой кнопки мыши по форме в ней появляется «отверстие» круглой формы, при отпускании форма восстанавливается;

15 Создать приложение при щелчке левой кнопки мыши форма плавно уменьшает высоту до фиксированного значения, при двойном щелчке плавно увеличивает высоту до фиксированного значения (воспользоваться циклом for…to…do и for…downto…do).

2.7 Дополнительная литература

1 Свойства и характеристики форм в Делфи [Электронный ресурс]. – Режим доступа: http://delphi-faq.ru/palitra-komponentov-delphi/vizyalnie-biblioteki-vcl/svojstva-i-xarakteristiki-formy-form-v-delfi-delphi.html;

2 Borland Delphi для начинающих – Работа с формой [Электронный ресурс]. – Режим доступа: http://www.codenet.ru/progr/delphi/learn/workwithform.php;

3 Функция Sleep в Delphi [Электронный ресурс]. – Режим доступа: http://serj.kz/content/21.

3 Лабораторная работа №3. Прямая работа с файлами в среде Delphi

3.1 Цель работы

Изучить основные принципы работы с процедурами записи и чтения данных из файла в среде визуального программирования Delphi.

3.2 Ход работы

Задание общего уровня

1 Изучить методические рекомендации к лабораторной работе;

2 Создать новый проект delphi (запустить приложение delphi или выполнить команду file->new->application);

3 Спроектировать форму delphi, согласно рисунка 13, представленного в методических рекомендациях;

4 Организовать событие onclick для объекта button1;

5 По щелчку на кнопку организовать запись в текстовый файл тех вводимых чисел, которые являются четными, имя текстового файла любое (см. методические рекомендации);

6 Сохранить проект на внешнем носителе в директории с именем lab3_1;

7 Осуществить отладку и компиляцию проекта (клавиша f9);

8 В отчет занести постановку задачи, листинг программно модуля unit unit1, скриншот окна результата работ проекта и содержимое текстового файла;

9 В имеющемся проекте организовать метод, осуществляющий чтение записанных чисел в файле и подсчет среднего арифметического числа. Результат подсчета вывести на форму в компонент label (см. методические рекомендации);

10 Оформить форму, подписать и аккуратно расположить все компоненты;

11 Сохранить проект;

12 Осуществить отладку и компиляцию проекта;

13 Оформить отчет;

14 Преобразовать форму имеющегося проекта в соответствии с рисунком 14;

15 Для объекта button2 организовать событие onclick;

16 В локальном разделе var образовавшегося метода объявить переменные srar (среднее арифметическое) типа real, sum типа integer, i типа integer;

17 В образовавшемся методе организовать чтение чисел из созданного текстового файла (см. методические рекомендации);

18 Оформить форму, подписать и аккуратно расположить все компоненты;

19 Сохранить проект;

20 Осуществить отладку и компиляцию проекта;

21 Оформить отчет;

22 В уже имеющемся проекте организовать проверку существования файла, используя процедуры fileexist, ioresult;

Задание повышенного уровня

23 Создать новый проект delphi, выполнить индивидуальное задание;

24 Сохранить проект в директории с именем lab3_2;

25 Осуществить отладку и компиляцию проекта;

26 Оформить отчет, в который поместить постановку задачи индивидуального задания, листинг программного модуля и результат работы проекта;

27 Защитить лабораторную работу.

3.3 Содержание отчета

1 Цель работы;
2 Ход работы;
3 Постановка задачи на запись данных в файл, постановка задачи на чтение данных из файла;
4 Листинг программного модуля;
5 Результат работы приложение на запись данных в файл, содержимое файла, результат работы программы на чтение данных из файла.

3.4 Контрольные вопросы

1 Основные понятия объектно-ориентированного программирования: объект, свойство, событие, метод, класс. Привести программные примеры обращения или описания основных понятий;
2 Назовите основные принципы объектно-ориентированного программирования, приведите программные примеры;
3 Приведите примеры объявления файловой переменной;
4 Перечислите последовательность процедур, используемых для записи данных в файл;
5 Перечислите последовательность процедур для чтения данных из файла;
6 Какие процедуры используются для проверки существования файла, как можно обработать ошибку открытия файла в проекте.

3.5 Методические рекомендации

Технология работы с файлами в системе Delphi требует определённого порядка действий:

1 Файл должен быть открыт. Система следит, чтобы другие приложения не мешали работе с файлом. При этом определяется, в каком режиме открывается файл - для изменения или только считывания информации. После открытия файла в программу возвращается его идентификатор, который будет использоваться для указания на этот файл во всех процедурах обработки;

2 Начинается работа с файлом. Это могут быть запись, считывание, поиск и другие операции;

3 Файл закрывается. Теперь он опять доступен другим приложениям без ограничений. Закрытие файла гарантирует, что все внесённые изменения будут сохранены, так как для увеличения скорости работы изменения предварительно сохраняются в специальных буферах операционной системы.

В Delphi реализовано несколько способов работы с файлами. Прямая работа с файлами осуществляется через процедуры языка Pascal, связанными с использованием файловых переменных. Файловая переменная вводится для указания на файл.

var F: File;

Описанная таким образом файловая переменная считается *нетипизированной*, и позволяет работать с файлами с неизвестной структурой. Данные считываются и записываются побайтно блоками, размер которых указывается при открытии файла, вплоть от 1 байт.

Но чаще используются файлы, состоящие из последовательности одинаковых записей. Для описания такого файла к предыдущему описанию добавляется указание типа записи:

var F: File ofmun_записи;

В качестве типа могут использоваться базовые типы, или создаваться свои. Важно только, чтобы для типа был точно известен фиксированный размер в байтах, поэтому тип String в чистом виде применяться не может.

Для обозначения файла, содержащего разноплановый набор символов используется следующие объявление:

var F: TextFile;

3.5.1 Открытие файла

Для открытия файла нужно указать, где он расположен. Для этого файловая переменная должна быть ассоциирована с нужным файлом, который определяется его адресом. Адрес файла может быть абсолютным, с указанием диска и каталогов ('C:\Мои документы\Мои рисунки\FileName.ini'), или относительным, тогда он создаётся в папке с .exe файлом программы. Для задания относительного адреса достаточно указать имя файла с нужным расширением. Делается это оператором AssignFile :

AssignFile(SaveF, 'C:\Мои документы\Мои рисунки\FileName.ini');

AssignFile(SaveF, 'FileName.ini');

Теперь файл должен быть открыт.

Открытие файла оператором **Rewrite** приведёт к воссозданию файла заново, т.е. существующий файл будет ***без предупреждения*** уничтожен, и на его месте будет создан новый пустой файл заданного типа, готовый к записи данных.Если же файла не было, то он будет создан.

Rewrite(SaveF);

Открытие файла оператором **Reset** откроет существующий файл к считыванию или записи данных, и его указатель будет установлен на начало файла:

Reset(SaveF);

Каждый из этих операторов может иметь второй необязательный параметр, имеющий смысл для нетипизированных файлов, и указывающий длину записи нетипизированного файла в байтах:

Rewrite(SaveF, 1);

Reset(SaveF, 1);

Открытие файла для добавления данных осуществляется оператором **Append**. Происходит добавление записей в конец файла.

Append(SaveF);

3.5.2 Чтение и запись данных файла

Чтение данных из файла можно осуществить, только, если он был открыт оператором **Reset.**

read(файловая_переменная, список_переменных)

readln(файловая_переменная, список_переменных)

Отличие между этими процедурами в том, что при вызове инструкции readln указатель чтения из файла автоматически перемещается в начало следующей строки файла.

Запись данных в файл осуществляется только в том случае, если файл открыт операторами **Rewrite** и **Append**.

Write(файловая_переменная, список_переменных)

Writeln(файловая_переменная, список_переменных)

Различие между инструкциями в том, что инструкция writeln после вывода всех значений, записывает в файл символ «новая строка».

При этом чтение и запись производится с текущей позиции указателя, затем указатель устанавливается на следующую запись. Можно проверить, существует ли нужный файл, оператором FileExists :

If FileExists('FileName.ini')thenRead(SaveF, SaveV);

Принудительно установить указатель на нужную запись можно оператором Seek(SaveF, N), где N - номер нужной записи, который, как и почти всё в программировании, отсчитывается от нуля:

Seek(SaveF, 49); - установка указателя на 50-ю запись.

При последовательном чтении из файла рано или поздно будет достигнут конец файла, и при дальнейшем чтении произойдёт ошибка. Проверить, не достигнут ли конец файла, можно оператором EOF (аббревиатура EndOfFile), который равен true, если указатель установлен в конец файла:

while (not EOF(SaveF)) do

Read(SaveF, SaveV);

Оператор Truncate(SaveF) позволяет отсечь (стереть или удалить!) все записи файла, начиная от текущей позиции указателя, и до конца файла.

В конце работы с файлом его необходимо закрыть. Это делается оператором CloseFile(SaveF) ;

3.5.3 Ошибки открытия файла

Попытка открыть файл может завершиться неудачей и вызвать ошибку выполнения программы. Причин неудачи открытия файлов может быть несколько: программа может попытаться открыть файл на гибком диске, который не готов к работе или отсутствие открываемого в режиме добавления файла (файла нет — добавлять некуда).

Если программа запущена из Delphi, то при возникновении ошибки во время открытия файла, возникает исключение и на экране появляется диалоговое окно с сообщением об ошибке.

Программа может взять на себя задачу контроля результата выполнения инструкции открытия файла. Сделать это можно, проверив значение функции IOResult (Input-OutputResult — результат ввода-вывода). Функция IOResult возвращает 0, если операция ввода-вывода завершилась успешно; в противном случае — не ноль. Чтобы программа могла проверить результат выполнения операции ввода-вывода, нужно разрешить ей это делать. Для этого надо перед инструкцией вызова процедуры открытия файла поместить директиву компилятору — строку {$I-}, которая запрещает автоматическую обработку ошибок ввода-вывода. После инструкции открытия файла следует поместить директиву {$I+}, восстанавливающую режим обработки ошибок ввода/вывода.

AssignFile (f , filename) ;

{$I-}

Append (f)

{$I +}

if IOResult<>0 then Rewrite (f)

3.5.4 Запись чисел с формы Delphi в текстовый файл

Для организации записи данных с формы Delphi используются компоненты, в которые можно осуществить текстовый ввод данных. Например, компонент Edit, класса TEdit, Memoкласс TMemo, MaskEdit класс TMaskEdit и т.п.

Организовать форму в новом проекте Delphi, согласно следующего вида (рисунок 13).

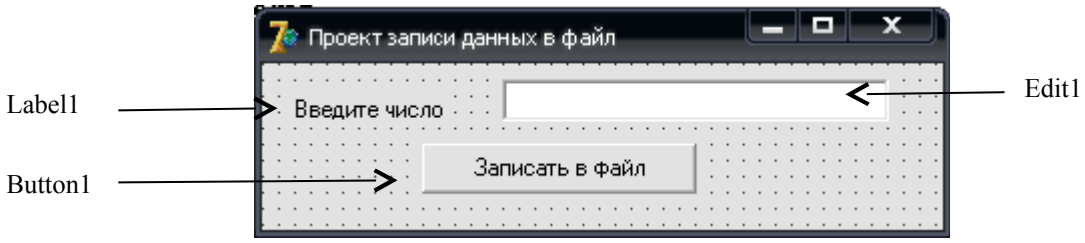

Рисунок 13 – Форма проекта в Delphi

Запись данных в текстовый файл организовать по щелчку. Ниже представлен обработчик события OnClick компонента Button1.

Procedure TForm1.Button1Click (Sender:TObject);

Var

F:TextFile;

a:integer;

Begin

AssignFile (F,**'E:\1.txt'**); *//связываем переменную с физическим файлом, имя файла задать свое*

Append (F); *//открыть файл для добавления*

a:=StrToInt(Edit1.Text); //*переводим строку символов в число*

```
if a mod 2=0        //если остаток от деления на 2 равен нулю,     значит число четное – записываем его в файл
then
begin
writeln(F,a);       // в файл записываем значение переменной a
ShowMessage ('Число записано в файл!')
end

else
        ShowMessage(‘Числонечетное!’);
Closefile(F);                       // закрытьфайл
End;
```

3.5.5 Чтение чисел из текстового файла, подсчет среднего арифметического значения

Для осуществления чтения данных из файла на форму Delphi необходимо добавить дополнительные компоненты (рисунок 14).

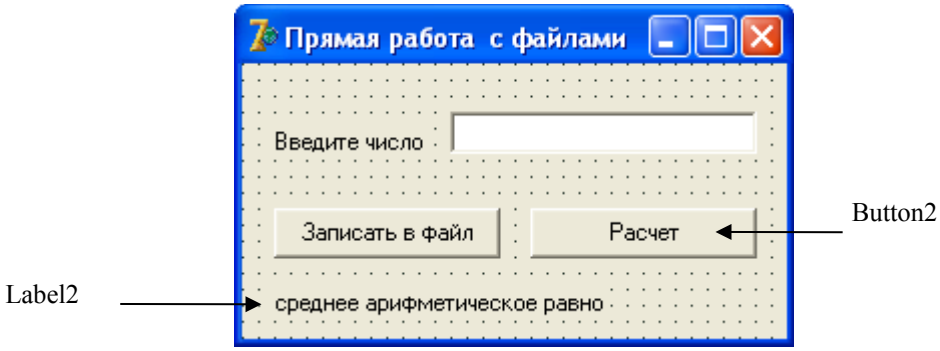

Рисунок 14 – Измененная форма проекта

Тело метода, осуществляющего чтение чисел из файла и расчет их среднего арифметического значения представлено ниже:

```
sum:=0;                    // в переменную суммы заносим пустое значение
i:=0;                      // «обнуляем» счетчик
AssignFile(F,'F:\1.txt');  // связываем файл с файловой переменной
Reset(F);                  //открыть файл для чтения
WhilenotEof(f) do          // «пока не достигнут конец файла делать…»
begin
readln(F,a);               // читаем число в переменную «a»
sum:=sum+a;                // вычисляем сумму
i:=i+1;                    // счетчик количества чисел
end;                       // закрыть цикл «пока»
SrAr:=sum/i;               // расчет среднего арифметического
Label2.Caption:='Среднееарифметическоеравно - '+FloatToStr(SrAr);
CloseFile(F);              // закрыть файл
```

3.6 Варианты индивидуальных заданий

1 Составить программу, с помощью которой можно просматривать и редактировать список группы и информацию о каждом студенте. Форма может иметь вид, представленный на рисунке 15:

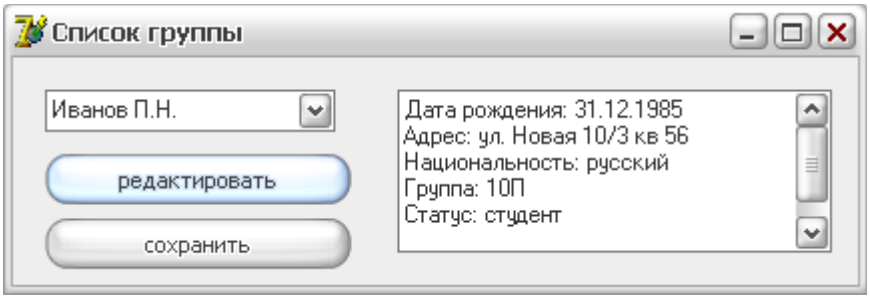

Рисунок 15 – Примерный вид формы

2 Разработать программу, которая в поле Memo выводит содержимое текстового файла. Рекомендуемый вид формы приведен на рисунке 16:

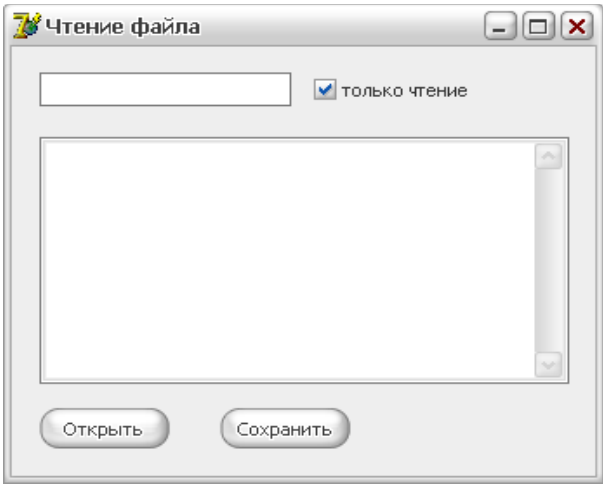

Рисунок 16 – Примерный вид формы

3 Разработать программу, которая в поле Memo выводит содержимое текстового файла. Для получения от пользователя имени файла используйте стандартное диалоговое окно «Открытие файла». Рекомендуемый вид на рисунке 17:

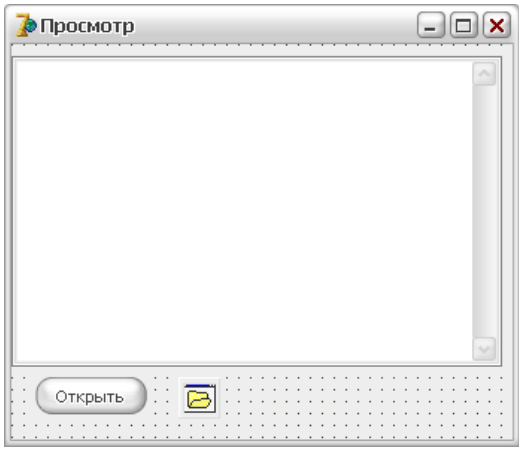

Рисунок 17 – Примерный вид формы

4 Разработать программу, которая добавляет ведет учет температуры воздуха в определенный день, реализованную в виде текстового файла, информацию о дневной температуре. Для ввода даты используйте компонент MonthCalendar. Если файл данных отсутствует, то программа должна его создать. Рекомендуемый вид на рисунке 18:

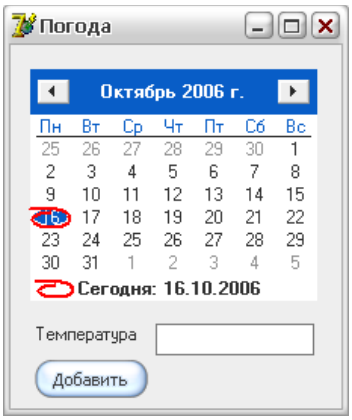

Рисунок 18 – Примерный вид формы

5 Разработать программу-ежедневник, которая добавляет в файл планы на текущий или предстоящий день. Рекомендуемый вид формы на рисунке 19:

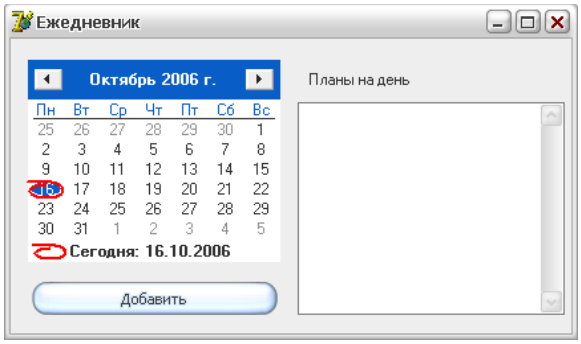

Рисунок 19 – Примерный вид формы

6 Разработать программу тестирования, в которой выбор правильного ответа осуществляется при помощи переключателей. Вопросы и варианты заданий хранятся в файле. В тесте задаются три вопроса: что такое инкапсуляция, наследование, полиморфизм. Рекомендуемый вид формы на рисунке 20:

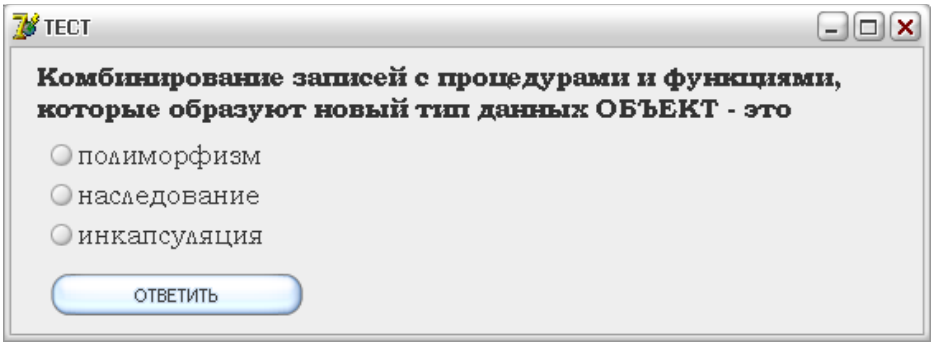

Рисунок 20 – Примерный вид формы

7 Разработать программу, которая ведет учет основных сведений о руководстве фирмы. Программа должна позволять вносить изменения и сохранять новые данные. Рекомендуемый вид формы на рисунке 21:

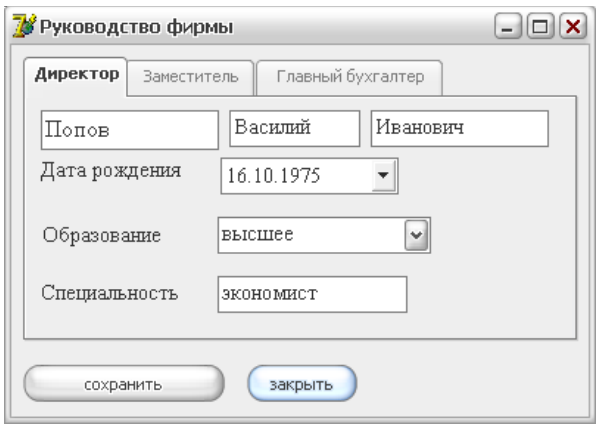

Рисунок 21 – Примерный вид формы

8 Разработать программу, которая выводит сведения о конфигурации ПК, характеристику устройства. Внешний вид формы аналогичен заданию №1;

9 Разработать программу, которая позволяет просматривать и корректировать сведения об организации (Название, полное именование, ИНН, Банк и его реквизиты, партнеры, список учредителей и т.п.). Рекомендуемый внешний вид аналогичен заданию №7;

10 Разработать программу «Ведомость успеваемости». Программа, должна содержать список группы и успеваемость конкретного студента по трем предметам. Рекомендуемый вид формы на рисунке 22:

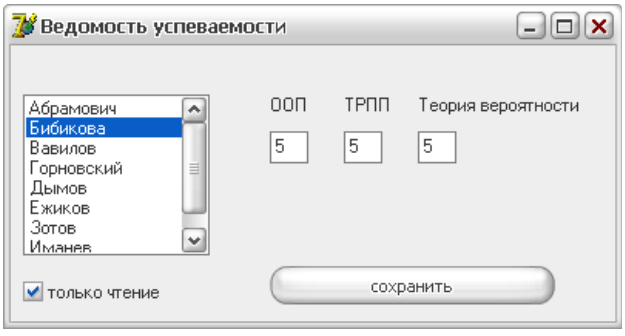

Рисунок 22 – Примерный вид формы

11 Разработать программу, которая позволяет оформлять заказ клиента фирмы и сохранять эту информацию в файле с его именем. Рекомендуемый вид формы на рисунке 23:

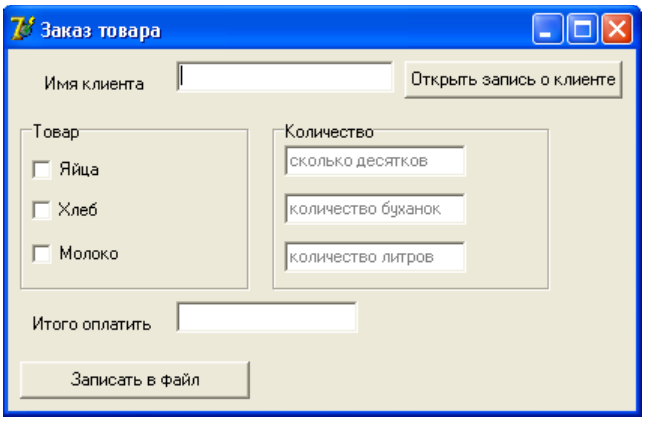

Рисунок 23 – Примерный вид формы

12 Разработать программу, которая позволяет просматривать список книг в библиотеке и редактировать его по желанию. Если книга удалена или получена новая, должен перезаписываться соответствующий файл;

13 Разработать программу, которая позволяет производить вычисления в матрице. Данные матрицы хранятся в файле. Программа производит считывание из файла и формирует их на форме. Рекомендуемый вид формы на рисунке 24:

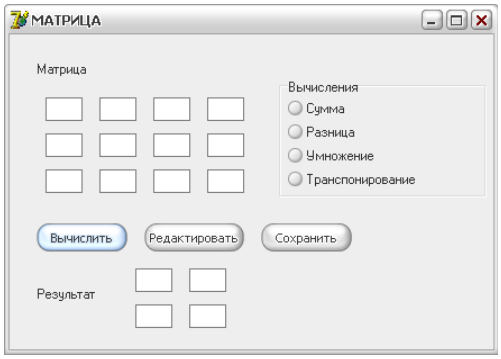

Рисунок 24 – Рекомендуемый внешний вид формы

14 Разработать программу, которая осуществляет ведение сведений о группах колледжа, их численности, кураторе и закрепленной аудиторией. Программа позволяет просматривать и редактировать данные. Внешний вид аналогичен рисунку 25:

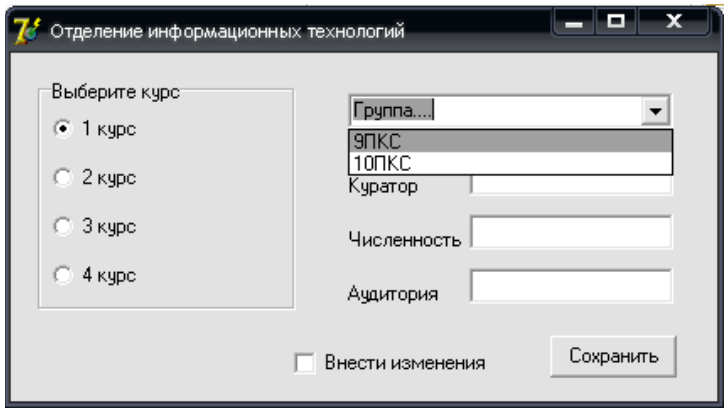

Рисунок 25 – Рекомендуемый внешний вид формы

15 Разработать программу, которая осуществляет хранение в файле данных о книгах в книжном магазине. В файл записывать сведения о книге, авторе, стоимости книги. Программа должна запрашивать у покупателя выбор книги, сумму за книгу, подсчитывать сдачу (или выводить сообщение о разнице в цене). Рекомендуемый вид формы на рисунке 26:

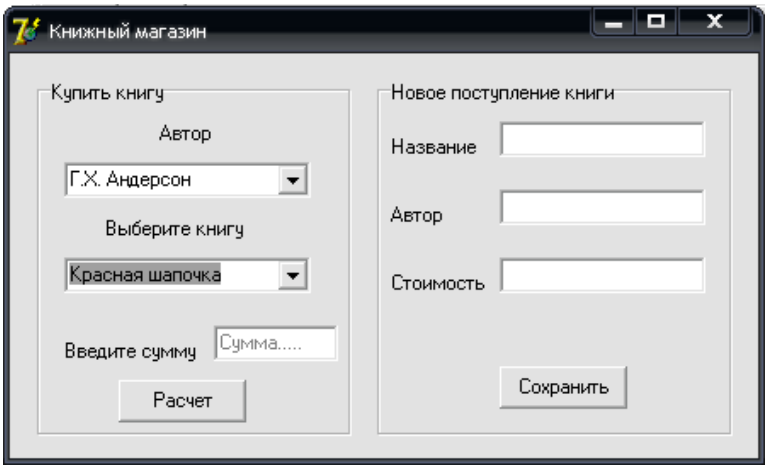

Рисунок 26 – Рекомендуемый внешний вид формы

Список использованных источников

1 Хомоненко А.Д. Delphi 7 / А.Д. Хомоненко [и др.]. – СПб.: БХВ-Петербург, 2008. – 1216 с. : ил.

2 Культин Н.Б. Delphi в задачах и примерах [Комплект] / Н.Б. Культин. – 3 изд. – СПб.: БХВ-Петербург, 2012. – 228 с. : ил. – 1 электрон. опт. диск (CD-ROM).

3 Катаев М.Ю. Объектно-ориентированное программирование : лабораторный практикум / М.Ю. Катаев. – Томск: Томский межвузовский центр дистанционного образования, 2007. – 68 с.

4 Марченко А.И. Программирование в среде TurboPascal 7.0 : учебное пособие / А.И. Марченко, Л.А. Марченко. – 9 изд. – СПб.: Корона-Век, 2007. – 458 с.

5 Бескоровайный И.В. Азбука Delphi : программирование с нуля / И.В. Бескоровайный. – Новосибирск: Сиб. унив. изд-во, 2008. – 112 с.

6 Хомоненко А. Delphi 7. [Электронный ресурс] : Электронно-библиотечная система ibooks.ru. / А. Хомоненко, В. Гофман, Е. Мещеряков. – СПб.: Айбукс. – 2010. – Режим доступа : http://ibooks.ru/product.php?productid=18411

7 Профессиональные программы для разработчиков [Электронный ресурс] : Delphi World / под ред. Н. Акулова. – Алматы: WDS, 2002. – Режим доступа : http://delphiworld.narod.ru/.

8 Королевство Delphi. Виртуальный клуб программистов [Электронный ресурс] / под ред. Е. Филипповой. – М.: DOTNETPARK, 1998. – Режим доступа : http://delphikingdom.ru/index.asp.