МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Филиал ФГАОУ ВО «ЮУрГУ (НИУ)» в г. Нижневартовске

Кафедра «Гуманитарные, естественнонаучные и технические дисциплины»

Разработка программы по объединению списков

(тема работы)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

По дисциплине Объектно-ориентированное программирование

ЮУрГУ-09.03.04.2020.006 ПЗ КП

Нормоконтролер
_____
_____2020 г.

Руководитель
Доцент, кандидат технических наук
В.А. Парасич_____2020г.

Автор работы
студент группы НвФл-222
Ишниязов Т.Р._____
_____2020г.

Работа защищена
с оценкой
_____
_____2020г.

Нижневартовск 2020

ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Филиал ФГАОУ ВО «ЮУрГУ (НИУ)» в г. Нижневартовске

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Рябова И.Г.
_____ 2020г.

ЗАДАНИЕ
на курсовую работу студента
Ишниязова Тимура Руслановича

Группа НвФл-222

1 Дисциплина Объектно-ориентированное программирование
2 Тема работы Обработчик списков с реализацией объединения двух списков в одном
3 Срок сдачи студентом законченной работы 27.03.2020 г.
4 Перечень вопросов, подлежащих разработке:

- изучение компонентов Delphi используемых в программе
- установление требований к программе
- разработка кода программы

КАЛЕНДАРНЫЙ ПЛАН

Наименование разделов курсовой работы	Срок выполнения разделов работы	Отметка о выполнении руководителя
Составление списка компонентов	15.02.2020 – 25.02.2020	
Разработка требований к программе	25.02.2020 – 09.03.2020	
Написание кода программы	09.03.2020 – 20.03.2020	
Оформление работы	20.03.2020 – 26.03.2020	

Руководитель работы _____/_____Парасич В.А.__ /
(подпись)

Студент _____/___Ишниязов Т.Р.__/
(подпись)

АННОТАЦИЯ

Ишниязов Т.Р. Объектно-ориентированное программирование Нижневартовск: филиал ЮУрГУ; 2020, 35 с. библиогр. список – 10 наим.

В ходе работы над курсовой работой был разработан обработчик списков, который имеет все базовые функции редактирования списков, и в дополнении реализована функция объединения двух списков в один.

Были описаны требования к программному обеспечению пользователя и разработчика. Также представлены код программы и интерфейс, который видит пользователь.

Целью настоящей курсовой работы является овладение принципами программирования в объектно-ориентированной среде программирования Borland Delphi 2006, навыками проектирования Windows-приложений, практического использования стандартных компонентов, работы с матрицами и с типизированными файлами.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы: в современных условиях умение работать с обработчиками списков становится всё более актуальным, т.к. обеспечивает значительное повышение удобства и производительности выполнения работы.

Обработчик списков – самостоятельная компьютерная программа или компонент программного комплекса (например, программа для создания заметок или окно ввода в браузере), предназначенная для создания и изменения текстовых файлов.

Обработчик списков позволяет:

1) создавать и редактировать списки;
2) объединять два списка в один.

Обработчик списков имеет:

1) главное меню;
2) панель инструментов;
3) строку состояния.

Цель работы: Изучение базовых компонентов Delphi, выработка навыков работы в объектно-ориентированной среде программирования.

Задачи работы: Создание базового обработчика списков с дополнительно разработанной функцией, которая позволяет объединять строки из двух listbox в memo.

1. СПИСОК ИСПОЛЬЗУЕМЫХ КОМПОНЕНТОВ BORLAND DELPHI

Компонент	Описание
`ActionManager`	Список действий. Служит для централизованной реакции программы на действия пользователя, связанные с выбором одного из группы однотипных управляющих элементов, таких как опции меню, кнопки и т.д.
`ListBox`	Это массив строк. В него Можно загружать данные из текстовых файлов, и сохранять информацию в файл. Также в нём может сортировать строки.
`StatusBar`	Панель статуса предназначена для размещения разного рода служебной информации в приложениях. Пример – нижняя часть рамки окна текстового редактора Word
`ImageList`	Набор рисунков. Представляет собой хранилище для нескольких рисунков одинакового размера, например, пиктограмм для кнопок.
`ActionMainMenu`	Основная панель меню. Отображает меню и подменю пунктов.
`ActionToolBar`	Панель инструментов. Отображает действия как кнопки на панели инструментов
`TButton`	Командная кнопка. Используется для реализации в программе команд с помощью обработчика события `OnClick` этого компонента.

PopupMenu

Контекстное меню (PopupMenu) вызываемое в приложении Windows по щелчку правой кнопкой мыши является стандартной и удобной возможностью многих программ. PopupMenu предназначен для создания таких контекстных меню.

TEdit

Компонент TEdit представляет собой однострочное текстовое поле, служащее для ввода данных пользователем. Основным свойством компонента TEdit, передающим введённую информацию, является свойство Edit1.Text типа String.

TMemo

Компонент TMemo представляет собой область просмотра, служащее для вывода на экран нескольких строк текста. Текст хранится в свойстве Lines класса Tstrings и, таким образом, представляет собой пронумерованный набор строк (нумерация начинается с нуля).

2. СПИСОК ИСПОЛЬЗУЕМЫХ ОБЪЕКТОВ-ДЕЙСТВИЙ

2.1. ОСНОВНЫЕ ОБЪЕКТЫ-ДЕЙСТВИЯ

`Cut`	Удаление выделенного фрагмента и помещение его в буфер обмена
`Copy`	Помещение выделенного фрагмента в буфер обмена
`Paste`	Вставка содержимого буфера обмена в документ
`Select All`	Выделение содержимого всего документа
`Undo`	Отмена последнего действия
`Delete`	Удаление выделенного фрагмента
`Customize`	Меню настройки панели быстрого доступа

2.2. ПОЛЬЗОВАТЕЛЬСКИЕ ОБЪЕКТЫ-ДЕЙСТВИЯ

`ActionAdd`	Пополнение списков строками из `Edit1 и Edit2`
`ActionSort`	Сортировка списков
`ActionClear`	Очистка списков
`About`	Вызов справки
`Union`	Объединение списков
`ClearMemo`	Очистка списка Memo

3. ОПИСАНИЕ ПРОГРАММЫ

Объединение двух списков listbox в memo

В данной курсовой работе дополнительной реализуемой функцией было объединение двух списков в один. Был создан пользовательский объект-действие `Union`, которое добавляет строки из ListBox1 и ListBox2 в Memo1.

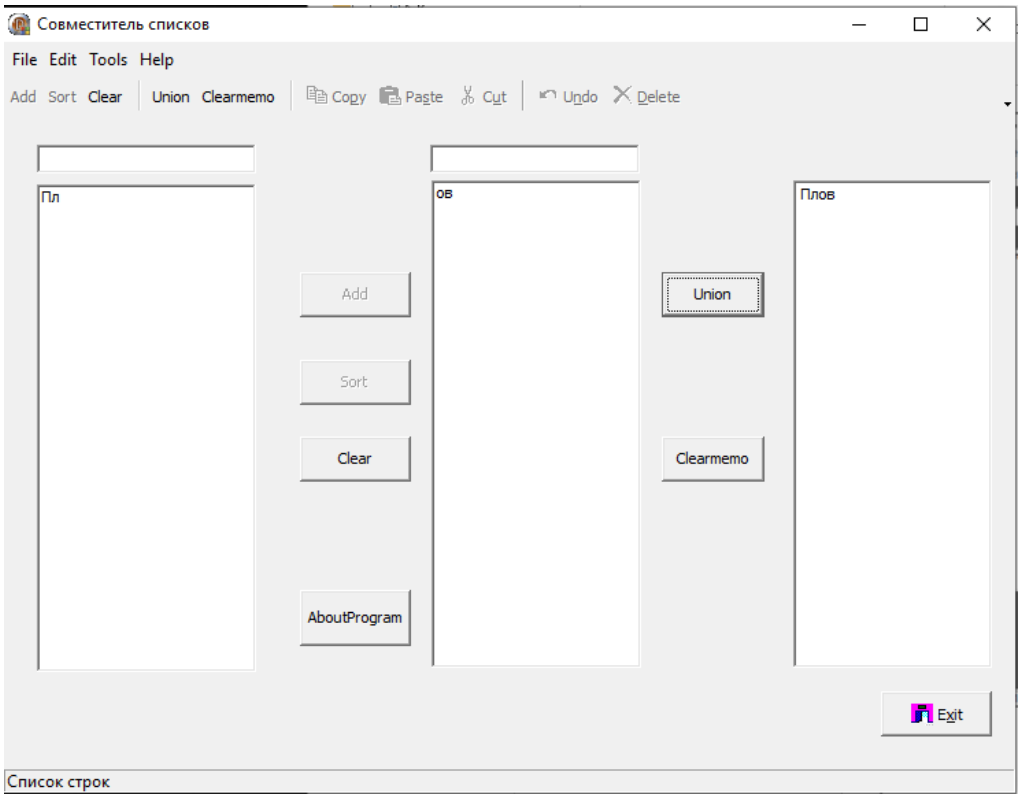

Рис. 1 – Скрин объединения двух списков Listbox в Memo

```
procedure TForm1.ActionUnionExecute(Sender: TObject);
begin
Memo1.Text:=ListBox1.Items.Commatext + Listbox2.Items.CommaText;
ActionClmemo.Enabled := True;
end;
```

Листинг 1 – Код объединения двух списков Listbox в Memo

Очистка списка Memo

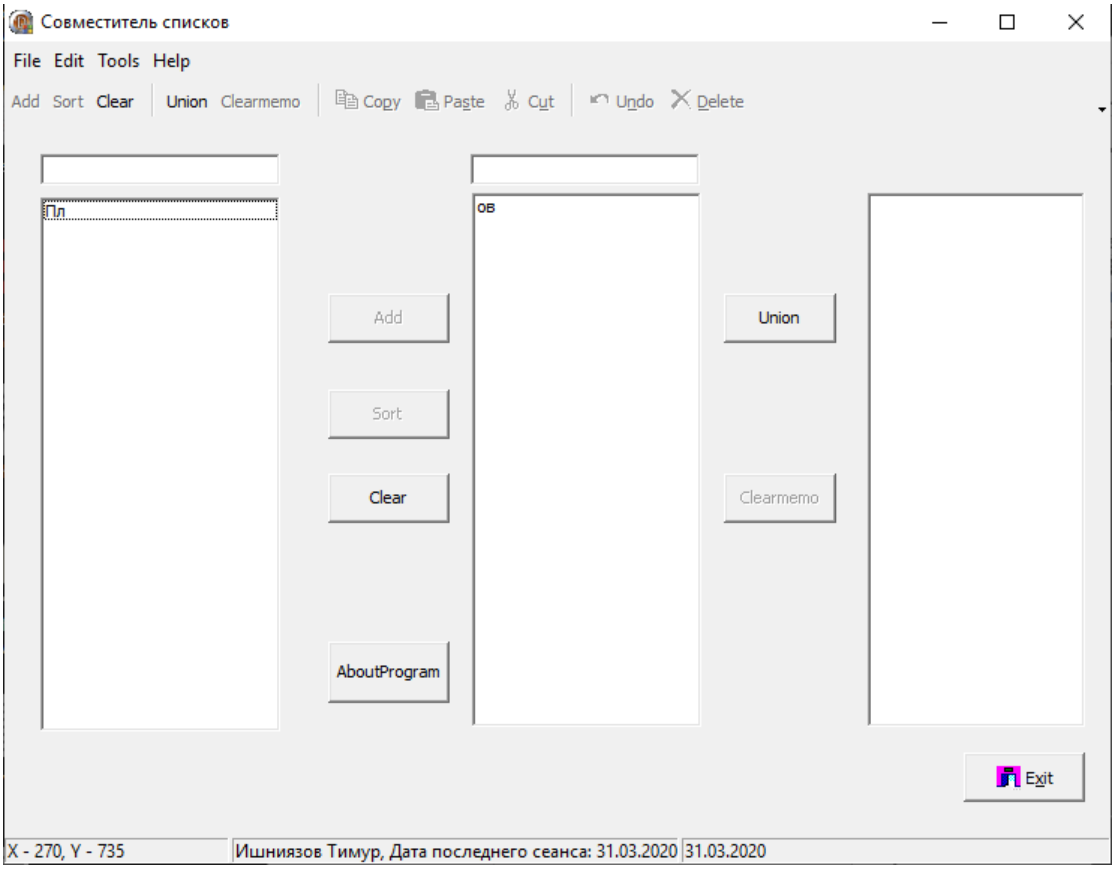

Рис. 2 – Скрин очистки списка Memo

```
procedure TForm1.ActionClmemoExecute(Sender: TObject);
begin
  if  (Memo1.Text <> '') then ActionClmemo.Enabled := False;

Memo1.Clear;
end;
```

Листинг 2 – Код очистки списка Memo

Если пользователю потребуется вставить скопированные строки в список, для этого ему необходимо выбрать необходимую строку в ComboBox, и нажать на кнопку Paste.

Сохранение состояния программы

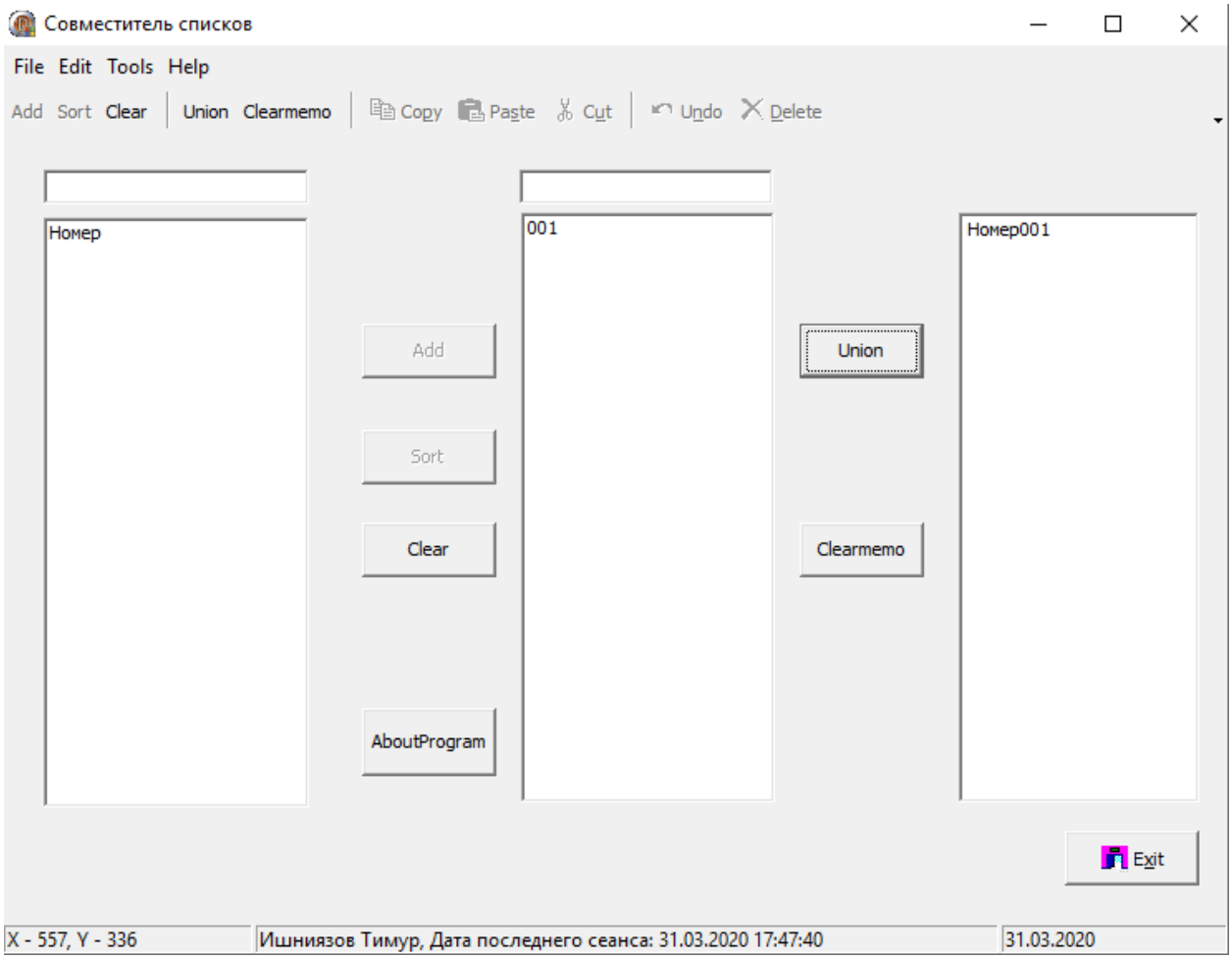

Рис. 3 – Скрин состояния программы, перед её закрытием

```
procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
var
  Status, i : Integer;
begin
    case WindowState of
      wsNormal:
        begin { если состояние нормальное, то сохраняем
положение и размер}
          IniFile.WriteInteger ('MainForm', 'Top', Top);
          IniFile.WriteInteger ('MainForm', 'Left', Left);
          IniFile.WriteInteger ('MainForm', 'Width', Width);
          IniFile.WriteInteger ('MainForm', 'Height', Height);
```

```
          Status := 1;
        end;
      wsMinimized: Status := 2;  {бесполезно: никогда не
устанавливается VCL!}
      wsMaximized: Status := 3;
    end;
    { проверяем, минимизировано ли окно, т.е. форма скрыта и не
активна }
    if not Active then
      Status := 2;

    { записываем информацию о состоянии формы }
    IniFile.WriteInteger ('MainForm', 'Status', Status );
    IniFile.WriteDateTime ('MainForm', 'Date', Now );
// записываем информацию о состоянии компонентов
    IniFile.WriteBool ('Component', 'ActionAdd_Enabled',
ActionAdd.Enabled );
    IniFile.WriteBool ('Component', 'ActionSort_Enabled',
ActionSort.Enabled );
    IniFile.WriteBool ('Component', 'ActionClear_Enabled',
ActionClear.Enabled );
    IniFile.WriteBool ('Component', 'ActionClmemo_Enabled',
ActionClmemo.Enabled );
    IniFile.WriteBool ('Component', 'ActionUnion_Enabled',
ActionUnion.Enabled );
    IniFile.WriteBool ('Component', 'Sorted1', ListBox1.Sorted
);
    IniFile.WriteBool ('Component', 'Sorted2', ListBox2.Sorted
);
    IniFile.WriteString('Component', 'Edit1_Text', Edit1.Text);
    IniFile.WriteString('Component', 'Edit2_Text', Edit2.Text);
    IniFile.WriteInteger ('Component', 'ListBox1_Count',
ListBox1.Items.Count );
    for i := 0 to ListBox1.Items.Count - 1 do
```

```
      IniFile.WriteString ('Component', IntToStr( i ),
ListBox1.Items[ i ]  );
      IniFile.WriteInteger ('Component', 'ListBox2_Count',
ListBox2.Items.Count );
    for i := 0 to ListBox2.Items.Count - 1 do
      IniFile.WriteString ('Component', IntToStr( i )+ 'k',
ListBox2.Items[ i ]  );
    IniFile.Free;   { всегда уничтожаем объект IniFile }
end;
```

Листинг 3 – Код сохранения состояния элементов

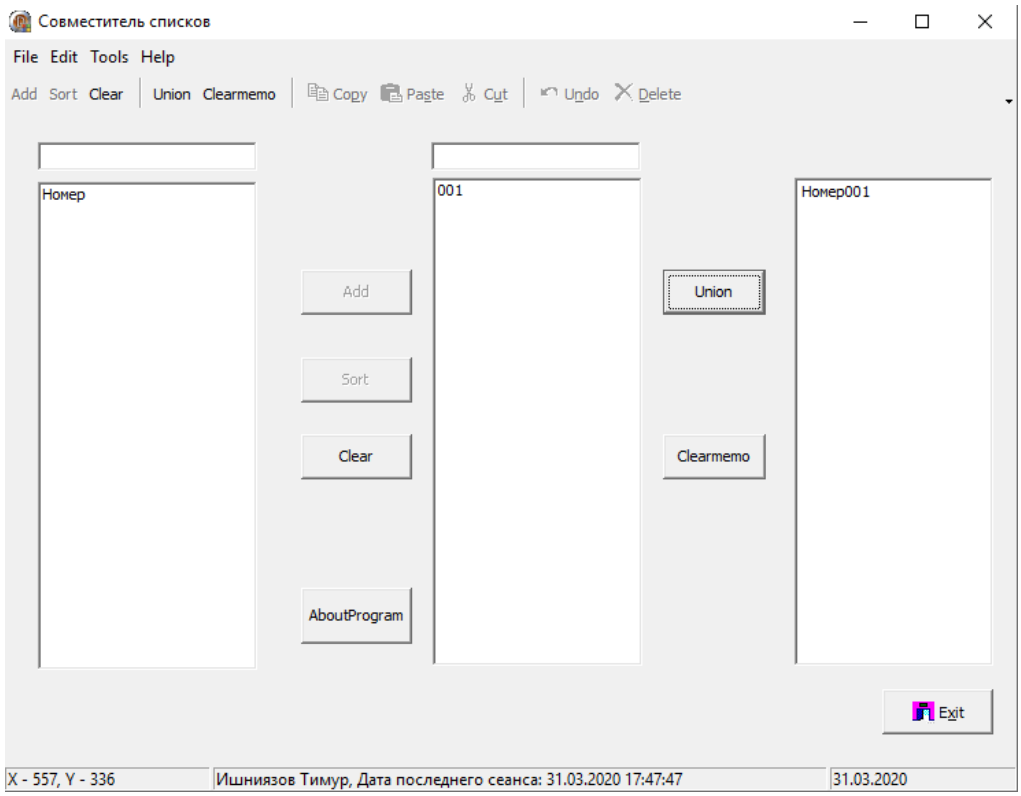

Рис 4 –Состояние программы после запуска

```
procedure TForm1.FormCreate(Sender: TObject);
var
  Status, i, N,K : Integer;
  str:string;
begin
  StatusBar1.Panels[2].Text := DateTimeToStr(Date);
  IniFile := TIniFile.Create ( ChangeFileExt (
Application.ExeName, '.ini' ) );
// - запись файла .INI в каталог программы
```

```
  { читаем значение статуса и проверяем, существует ли оно }
  Status := IniFile.ReadInteger ('MainForm', 'Status', 0);
  if Status <> 0 then
  begin { читаем положение и размеры формы, используя их текущие
значения
          в качестве значений по умолчанию }
    Top := IniFile.ReadInteger ('MainForm', 'Top', Top);
    Left := IniFile.ReadInteger ('MainForm', 'Left', Left);
    Width := IniFile.ReadInteger ('MainForm', 'Width', Width);
    Height := IniFile.ReadInteger ('MainForm', 'Height',
Height);
    { устанавливаем минимизированное или максимизированное
состояние }
    case Status of
      // 1: WindowState := wsNormal; т.к. уже установлено
      2: WindowState := wsMinimized;
      3: WindowState := wsMaximized;
    end;
  end;
// прочитали состояние кнопок
    ActionAdd.Enabled := IniFile.ReadBool ('Component',
'ActionAdd_Enabled', false );
    ActionSort.Enabled := IniFile.ReadBool ('Component',
'ActionSort_Enabled', false );
    ActionClear.Enabled := IniFile.ReadBool ('Component',
'ActionClear_Enabled', false );
    ListBox1.Sorted := IniFile.ReadBool ('Component', 'Sorted1',
false );
    ListBox2.Sorted := IniFile.ReadBool ('Component', 'Sorted2',
false );
    Edit1.Text := IniFile.ReadString('Component','Edit1_Text',
'');
     Edit2.Text := IniFile.ReadString('Component','Edit2_Text',
'');
```

```
    str := IniFile.ReadString('Component','Picture_Path', '');
    StatusBar1.Panels[1].Text := StatusBar1.Panels[1].Text + ',
Дата последнего сеанса: ' +
DateTimeToStr(IniFile.ReadDateTime('MainForm', 'Date', 0));
// прочитали число строк в ListBox
    N := IniFile.ReadInteger ('Component', 'ListBox1_Count', 0);
   for i := 0 to N - 1 do
      ListBox1.Items.Add( IniFile.ReadString ('Component',
IntToStr( i ), '') );
      K := IniFile.ReadInteger ('Component', 'ListBox2_Count',
0);
    for i := 0 to K - 1 do
      ListBox2.Items.Add( IniFile.ReadString ('Component',
IntToStr( i )+ 'k', '') );
      ActionClmemo.Enabled := IniFile.ReadBool ('Component',
'ActionClmemo_Enabled', false );
       ActionUnion.Enabled := IniFile.ReadBool ('Component',
'ActionUnion_Enabled', false );
end;
```

Листинг 4 – Присвоение сохранённых параметров компонентам

Перед закрытием формы возникает событие FormClose, в котором в файл с расширением ini, записывается состояние окна формы и её положение, а также активность или не активность кнопок, текст из строки и из списка. Также программа запоминает дату последнего сеанса, и выводит её в statusBar при следующей активации программы.

А при запуске программы, возникает событие `FormCreate`, в котором программа считывает данные из этого файла, принимает указанное в нём состояние и присваивает сохранённые параметры компонентам.

4. ЛИСТИНГ ПРОГРАММЫ

4.1. КОД ПРОГРАММЫ

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdActns, BandActn, ActnCtrls, ActnList, ToolWin,
ActnMan, ActnMenus,
  ComCtrls, ImgList, XPStyleActnCtrls, Buttons, StdCtrls,
AppEvnts, Menus ;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    ListBox2: TListBox;
    Memo1: TMemo;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    BitBtn1: TBitBtn;
    ActionManager1: TActionManager;
    ImageList1: TImageList;
    StatusBar1: TStatusBar;
    ActionMainMenuBar1: TActionMainMenuBar;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    ActionAdd: TAction;
    ActionSort: TAction;
    ActionClear: TAction;
    ActionUnion: TAction;
    ActionClmemo: TAction;
    Button4: TButton;
    Button5: TButton;
    ActionToolBar1: TActionToolBar;
    EditCut1: TEditCut;
    EditCopy1: TEditCopy;
    EditPaste1: TEditPaste;
    EditSelectAll1: TEditSelectAll;
    EditUndo1: TEditUndo;
    EditDelete1: TEditDelete;
    FileExit1: TFileExit;
    CustomizeActionBars1: TCustomizeActionBars;
    Button6: TButton;
    AboutProgram: TAction;
    ApplicationEvents1: TApplicationEvents;
    PopupMenu1: TPopupMenu;
```

```
    Add1: TMenuItem;
    Clear1: TMenuItem;
    Clear2: TMenuItem;
    N1: TMenuItem;
    Union1: TMenuItem;
    Clearmemo1: TMenuItem;
    N2: TMenuItem;
    Copy1: TMenuItem;
    Paste1: TMenuItem;
    Cut1: TMenuItem;
    N3: TMenuItem;
    Undo1: TMenuItem;
    Delete1: TMenuItem;
    procedure ActionSortExecute(Sender: TObject);
    procedure ActionAddExecute(Sender: TObject);
    procedure ActionClearExecute(Sender: TObject);
    procedure ActionUnionExecute(Sender: TObject);
    procedure ActionClmemoExecute(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action:
TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure AboutProgramExecute(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure FormMouseMove(Sender: TObject; Shift: TShiftState;
X, Y: Integer);
    procedure ApplicationEvents1Hint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
uses ABOUT,IniFiles;
Var IniFile: TIniFile;
{$R *.dfm}

procedure TForm1.AboutProgramExecute(Sender: TObject);
begin
  AboutBox.ShowModal;
end;

procedure TForm1.ActionAddExecute(Sender: TObject);
begin
   if (edit1.Text <> '') or (edit2.text <> '') then
     begin
  ListBox1.Items.Add(Edit1.Text);
  ListBox2.Items.Add(Edit2.Text);
  ActionClear.Enabled := true;
```

```
  if (ListBox1.Count > 1) and (ListBox2.Count > 1) then
  ActionSort.Enabled := true  ;
  if(listBox1.Items.Count > 0 ) and (listbox2.Items.count > 0 )
then
 ActionUnion.Enabled := true
else
 ActionUnion.Enabled := false;
  Edit1.Text := '';
  Edit2.Text := '';
  end;
  ActionAdd.enabled := false
end;

procedure TForm1.ActionClearExecute(Sender: TObject);
begin
  ListBox1.Clear;
  ListBox2.Clear;
  ActionSort.Enabled := false;
  ActionClear.Enabled := false;
  ListBox1.Sorted := False;
  ActionUnion.Enabled := False;


end;

procedure TForm1.ActionClmemoExecute(Sender: TObject);
begin
 if  (Memo1.Text <> '') then ActionClmemo.Enabled := False;

Memo1.Clear;
end;

procedure TForm1.ActionSortExecute(Sender: TObject);
begin
ListBox1.Sorted := True;
  ListBox2.Sorted := True;
  ActionSort.Enabled := false;
end;

procedure TForm1.ActionUnionExecute(Sender: TObject);
begin
 Memo1.Text := ListBox1.Items.Commatext +
Listbox2.Items.CommaText;
ActionClmemo.Enabled := True;
end;

procedure TForm1.ApplicationEvents1Hint(Sender: TObject);
begin
 if Length ( Application.Hint ) <> 0 then
    Begin  { При показе всплывающей подсказки переводим
             StatusBar в режим простой панели }
      StatusBar1.SimplePanel := True;
```

```
      StatusBar1.SimpleText := Application.Hint;
    end
  else  { Возврат к отображению StatusBar с многими панелями }
    StatusBar1.SimplePanel := False
end;

procedure TForm1.Edit1Change(Sender: TObject);
begin
 if (Edit1.Text <> '') or (edit2.Text <>'') then
ActionAdd.Enabled := true
  else ActionAdd.Enabled := false
end;

procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
var
  Status, i : Integer;
begin
    case WindowState of
      wsNormal:
        begin { если состояние нормальное, то сохраняем
положение и размер}
          IniFile.WriteInteger ('MainForm', 'Top', Top);
          IniFile.WriteInteger ('MainForm', 'Left', Left);
          IniFile.WriteInteger ('MainForm', 'Width', Width);
          IniFile.WriteInteger ('MainForm', 'Height', Height);
          Status := 1;
        end;
      wsMinimized: Status := 2;  {бесполезно: никогда не
устанавливается VCL!}
      wsMaximized: Status := 3;
    end;
    { проверяем, минимизировано ли окно, т.е. форма скрыта и не
активна }
    if not Active then
      Status := 2;

    { записываем информацию о состоянии формы }
    IniFile.WriteInteger ('MainForm', 'Status', Status );
    IniFile.WriteDateTime ('MainForm', 'Date', Now );
// записываем информацию о состоянии компонентов
    IniFile.WriteBool ('Component', 'ActionAdd_Enabled',
ActionAdd.Enabled );
    IniFile.WriteBool ('Component', 'ActionSort_Enabled',
ActionSort.Enabled );
    IniFile.WriteBool ('Component', 'ActionClear_Enabled',
ActionClear.Enabled );
    IniFile.WriteBool ('Component', 'ActionClmemo_Enabled',
ActionClmemo.Enabled );
    IniFile.WriteBool ('Component', 'ActionUnion_Enabled',
ActionUnion.Enabled );
```

```
    IniFile.WriteBool ('Component', 'Sorted1', ListBox1.Sorted
);
    IniFile.WriteBool ('Component', 'Sorted2', ListBox2.Sorted
);
    IniFile.WriteString('Component', 'Edit1_Text', Edit1.Text);
    IniFile.WriteString('Component', 'Edit2_Text', Edit2.Text);
    IniFile.WriteInteger ('Component', 'ListBox1_Count',
ListBox1.Items.Count );
    for i := 0 to ListBox1.Items.Count - 1 do
      IniFile.WriteString ('Component', IntToStr( i ),
ListBox1.Items[ i ]  );
      IniFile.WriteInteger ('Component', 'ListBox2_Count',
ListBox2.Items.Count );
    for i := 0 to ListBox2.Items.Count - 1 do
      IniFile.WriteString ('Component', IntToStr( i )+ 'k',
ListBox2.Items[ i ]  );
    IniFile.Free;   { всегда уничтожаем объект IniFile }
end;
procedure TForm1.FormCreate(Sender: TObject);
var
  Status, i, N,K : Integer;
  str:string;
begin
  StatusBar1.Panels[2].Text := DateTimeToStr(Date);

  IniFile := TIniFile.Create ( ChangeFileExt (
Application.ExeName, '.ini' ) );
// - запись файла .INI в каталог программы

  { читаем значение статуса и проверяем, существует ли оно }
  Status := IniFile.ReadInteger ('MainForm', 'Status', 0);
  if Status <> 0 then
  begin { читаем положение и размеры формы, используя их текущие
значения
          в качестве значений по умолчанию }
    Top := IniFile.ReadInteger ('MainForm', 'Top', Top);
    Left := IniFile.ReadInteger ('MainForm', 'Left', Left);
    Width := IniFile.ReadInteger ('MainForm', 'Width', Width);
    Height := IniFile.ReadInteger ('MainForm', 'Height',
Height);

    { устанавливаем минимизированное или максимизированное
состояние }
    case Status of
      // 1: WindowState := wsNormal; т.к. уже установлено
      2: WindowState := wsMinimized;
      3: WindowState := wsMaximized;
    end;
  end;
// прочитали состояние кнопок
    ActionAdd.Enabled := IniFile.ReadBool ('Component',
'ActionAdd_Enabled', false );
```

```
    ActionSort.Enabled := IniFile.ReadBool ('Component',
'ActionSort_Enabled', false );
    ActionClear.Enabled := IniFile.ReadBool ('Component',
'ActionClear_Enabled', false );
    ListBox1.Sorted := IniFile.ReadBool ('Component', 'Sorted1',
false );
    ListBox2.Sorted := IniFile.ReadBool ('Component', 'Sorted2',
false );
    Edit1.Text := IniFile.ReadString('Component','Edit1_Text',
'');
     Edit2.Text := IniFile.ReadString('Component','Edit2_Text',
'');
    str := IniFile.ReadString('Component','Picture_Path', '');
    StatusBar1.Panels[1].Text := StatusBar1.Panels[1].Text + ',
Дата последнего сеанса: ' +
DateTimeToStr(IniFile.ReadDateTime('MainForm', 'Date', 0));
// прочитали число строк в ListBox
    N := IniFile.ReadInteger ('Component', 'ListBox1_Count', 0);
   for i := 0 to N - 1 do
      ListBox1.Items.Add( IniFile.ReadString ('Component',
IntToStr( i ), '') );
      K := IniFile.ReadInteger ('Component', 'ListBox2_Count',
0);
    for i := 0 to K - 1 do
      ListBox2.Items.Add( IniFile.ReadString ('Component',
IntToStr( i )+ 'k', '') );
      ActionClmemo.Enabled := IniFile.ReadBool ('Component',
'ActionClmemo_Enabled', false );
       ActionUnion.Enabled := IniFile.ReadBool ('Component',
'ActionUnion_Enabled', false );
end;
procedure TForm1.FormMouseMove(Sender: TObject; Shift:
TShiftState; X,
  Y: Integer);
var
  P:TPoint;
begin
  GetCursorPos(P);
  StatusBar1.Panels[0].Text:= 'X - '+ IntToStr(P.X) + ', Y - ' +
IntToStr(P.Y)
end;
end.
```

Листинг 5 – Код программы

4.2. ФАЙЛ ФОРМЫ ПРОГРАММЫ

```
object Form1: TForm1
  Left = 0
  Top = 0
  BorderStyle = bsSingle
  Caption = {}
  ClientHeight = 542
  ClientWidth = 739
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  PopupMenu = PopupMenu1
  ShowHint = True
  OnClose = FormClose
  OnCreate = FormCreate
  OnMouseMove = FormMouseMove
  DesignSize = (
    739
    542)
  PixelsPerInch = 96
  TextHeight = 13
  object ListBox1: TListBox
    Left = 24
    Top = 101
    Width = 161
    Height = 356
    Hint = {}
    Anchors = [akLeft, akRight]
    ItemHeight = 13
    TabOrder = 0
    OnMouseMove = FormMouseMove
  end
  object ListBox2: TListBox
    Left = 312
    Top = 101
    Width = 153
    Height = 356
    Hint = {}
    ItemHeight = 13
    TabOrder = 1
    OnMouseMove = FormMouseMove
  end
  object Memo1: TMemo
    Left = 576
    Top = 101
    Width = 145
```

```
  Height = 356
  Hint = {}
  TabOrder = 2
  OnMouseMove = FormMouseMove
end
object Edit1: TEdit
  Left = 24
  Top = 74
  Width = 161
  Height = 21
  Hint = {}
  Anchors = [akLeft, akRight]
  ParentShowHint = False
  ShowHint = True
  TabOrder = 3
  Text = 'Edit1'
  OnChange = Edit1Change
  OnMouseMove = FormMouseMove
end
object Edit2: TEdit
  Left = 312
  Top = 74
  Width = 153
  Height = 21
  Hint = {}
  Anchors = []
  TabOrder = 4
  Text = 'Edit2'
  OnChange = Edit1Change
  OnMouseMove = FormMouseMove
end
object Button1: TButton
  Left = 216
  Top = 168
  Width = 81
  Height = 33
  Hint = {}
  Action = ActionAdd
  Enabled = False
  TabOrder = 5
end
object Button2: TButton
  Left = 216
  Top = 232
  Width = 81
  Height = 33
  Hint = {}
  Action = ActionSort
  TabOrder = 6
end
object Button3: TButton
  Left = 216
```

```
  Top = 288
  Width = 81
  Height = 33
  Hint = {}
  Action = ActionClear
  TabOrder = 7
end
object BitBtn1: TBitBtn
  Left = 640
  Top = 474
  Width = 81
  Height = 33
  Action = FileExit1
  Caption = 'E&xit'
  TabOrder = 8
  Glyph.Data = {}
end
object StatusBar1: TStatusBar
  Left = 0
  Top = 523
  Width = 739
  Height = 19
  Panels = <
    item
      Width = 150
    end
    item
      Text = {}
      Width = 450
    end
    item
      Width = 50
    end>
end
object ActionMainMenuBar1: TActionMainMenuBar
  Left = 0
  Top = 0
  Width = 739
  Height = 26
  UseSystemFont = False
  ActionManager = ActionManager1
  Caption = 'ActionMainMenuBar1'
  ColorMap.HighlightColor = clWhite
  ColorMap.BtnSelectedColor = clBtnFace
  ColorMap.UnusedColor = clWhite
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -12
  Font.Name = 'Segoe UI'
  Font.Style = []
  Spacing = 0
end
```

```
object Button4: TButton
  Left = 480
  Top = 168
  Width = 75
  Height = 33
  Hint = {}
  Action = ActionUnion
  TabOrder = 11
end
object Button5: TButton
  Left = 480
  Top = 288
  Width = 75
  Height = 33
  Hint = {}
  Action = ActionClmemo
  TabOrder = 12
end
object ActionToolBar1: TActionToolBar
  Left = 0
  Top = 26
  Width = 739
  Height = 26
  ActionManager = ActionManager1
  Caption = 'ActionToolBar1'
  ColorMap.HighlightColor = clWhite
  ColorMap.BtnSelectedColor = clBtnFace
  ColorMap.UnusedColor = clWhite
  Spacing = 0
end
object Button6: TButton
  Left = 216
  Top = 400
  Width = 81
  Height = 41
  Hint = {}
  Action = AboutProgram
  TabOrder = 14
end
object ActionManager1: TActionManager
  FileName = 'BigBOOm'
  ActionBars = <
    item
      Items = <
        item
          Action = ActionAdd
        end
        item
          Action = ActionSort
        end
        item
          Action = ActionClear
```

```
      end
      item
        Caption = '-'
      end
      item
        Action = ActionUnion
      end
      item
        Action = ActionClmemo
      end
      item
        Caption = '-'
      end
      item
        Action = EditCopy1
        Caption = 'Co&py'
        ImageIndex = 1
        ShortCut = 16451
      end
      item
        Action = EditPaste1
        Caption = 'Pa&ste'
        ImageIndex = 2
        ShortCut = 16470
      end
      item
        Action = EditCut1
        Caption = 'C&ut'
        ImageIndex = 0
        ShortCut = 16472
      end
      item
        Caption = '-'
      end
      item
        Action = EditUndo1
        Caption = 'U&ndo'
        ImageIndex = 3
        ShortCut = 16474
      end
      item
        Action = EditDelete1
        ImageIndex = 4
        ShortCut = 46
      end>
    ActionBar = ActionToolBar1
    AutoSize = False
  end
  item
    Items = <
      item
        Items = <
```

```
      item
        Action = FileExit1
        ImageIndex = 5
      end>
    Caption = '&File'
  end
  item
    Items = <
      item
        Action = ActionAdd
      end
      item
        Action = ActionSort
      end
      item
        Action = ActionClear
      end
      item
        Caption = '-'
      end
      item
        Action = ActionUnion
      end
      item
        Action = ActionClmemo
      end
      item
        Caption = '-'
      end
      item
        Action = EditCut1
        Caption = 'C&ut'
        ImageIndex = 0
        ShortCut = 16472
      end
      item
        Action = EditCopy1
        Caption = 'Co&py'
        ImageIndex = 1
        ShortCut = 16451
      end
      item
        Action = EditPaste1
        Caption = 'Pa&ste'
        ImageIndex = 2
        ShortCut = 16470
      end
      item
        Caption = '-'
      end
      item
        Action = EditSelectAll1
```

```
            Caption = 'S&elect All'
            ShortCut = 16449
          end
          item
            Caption = '-'
          end
          item
            Action = EditUndo1
            Caption = 'U&ndo'
            ImageIndex = 3
            ShortCut = 16474
          end
          item
            Action = EditDelete1
            ImageIndex = 4
            ShortCut = 46
          end>
        Caption = '&Edit'
      end
      item
        Items = <
          item
            Action = CustomizeActionBars1
          end>
        Caption = '&Tools'
      end
      item
        Items = <
          item
            Action = AboutProgram
            Caption = '&AboutProgram'
          end>
        Caption = '&Help'
      end>
    ActionBar = ActionMainMenuBar1
  end>
Images = ImageList1
Left = 32
Top = 112
StyleName = 'XP Style'
object ActionAdd: TAction
  Category = 'Edit'
  Caption = 'Add'
  OnExecute = ActionAddExecute
end
object ActionSort: TAction
  Category = 'Edit'
  Caption = 'Sort'
  OnExecute = ActionSortExecute
end
object ActionClear: TAction
  Category = 'Edit'
```

```
      Caption = 'Clear'
      OnExecute = ActionClearExecute
    end
    object ActionUnion: TAction
      Category = 'Edit'
      Caption = 'Union'
      OnExecute = ActionUnionExecute
    end
    object ActionClmemo: TAction
      Category = 'Edit'
      Caption = 'Clearmemo'
      OnExecute = ActionClmemoExecute
    end
    object EditCut1: TEditCut
      Category = 'Edit'
      Caption = 'Cu&t'
      Enabled = False
      Hint = 'Cut|Cuts the selection and puts it on the
Clipboard'
      ImageIndex = 0
      ShortCut = 16472
    end
    object EditCopy1: TEditCopy
      Category = 'Edit'
      Caption = '&Copy'
      Enabled = False
      Hint = 'Copy|Copies the selection and puts it on the
Clipboard'
      ImageIndex = 1
      ShortCut = 16451
    end
    object EditPaste1: TEditPaste
      Category = 'Edit'
      Caption = '&Paste'
      Hint = 'Paste|Inserts Clipboard contents'
      ImageIndex = 2
      ShortCut = 16470
    end
    object EditSelectAll1: TEditSelectAll
      Category = 'Edit'
      Caption = 'Select &All'
      Hint = 'Select All|Selects the entire document'
      ShortCut = 16449
    end
    object EditUndo1: TEditUndo
      Category = 'Edit'
      Caption = '&Undo'
      Hint = 'Undo|Reverts the last action'
      ImageIndex = 3
      ShortCut = 16474
    end
    object EditDelete1: TEditDelete
```

```
    Category = 'Edit'
    Caption = '&Delete'
    Hint = 'Delete|Erases the selection'
    ImageIndex = 4
    ShortCut = 46
  end
  object FileExit1: TFileExit
    Category = 'File'
    Caption = 'E&xit'
    Hint = 'Exit|Quits the application'
    ImageIndex = 5
  end
  object CustomizeActionBars1: TCustomizeActionBars
    Category = 'Tools'
    Caption = '&Customize...'
    CustomizeDlg.StayOnTop = False
  end
  object AboutProgram: TAction
    Category = 'Help'
    Caption = 'AboutProgram'
    OnExecute = AboutProgramExecute
  end
end
object ImageList1: TImageList
  Left = 152
  Top = 112
  Bitmap = {}
end
object OpenDialog1: TOpenDialog
  Left = 152
  Top = 240
end
object SaveDialog1: TSaveDialog
  DefaultExt = 'txt'
  Left = 160
  Top = 408
end
object ApplicationEvents1: TApplicationEvents
  OnHint = ApplicationEvents1Hint
  Left = 32
  Top = 400
end
object PopupMenu1: TPopupMenu
  Images = ImageList1
  Left = 32
  Top = 240
  object Add1: TMenuItem
    Action = ActionAdd
  end
  object Clear1: TMenuItem
    Action = ActionSort
  end
```

```
      object Clear2: TMenuItem
        Action = ActionClear
      end
      object N1: TMenuItem
        Caption = '-'
      end
      object Union1: TMenuItem
        Action = ActionUnion
      end
      object Clearmemo1: TMenuItem
        Action = ActionClmemo
      end
      object N2: TMenuItem
        Caption = '-'
      end
      object Copy1: TMenuItem
        Action = EditCopy1
      end
      object Paste1: TMenuItem
        Action = EditPaste1
      end
      object Cut1: TMenuItem
        Action = EditCut1
      end
      object N3: TMenuItem
        Caption = '-'
      end
      object Undo1: TMenuItem
        Action = EditUndo1
      end
      object Delete1: TMenuItem
        Action = EditDelete1
      end
    end
  end
```

Листинг 6 – Код файла формы

ЗАКЛЮЧЕНИЕ

В данной курсовой работе удалось разработать программу с использованием компонентов среды визуального проектирования Delphi. Выполнение курсовой работы было разделено на два этапа:

- теоретический анализ поставленной задачи, включающий в себя постановку задачи, выделение основных классов, выделение основных действия и алгоритмов их реализации;
- программная реализация задачи средствами Delphi.

Итогом курсовой работы можно считать закрепление знаний в области создания алгоритмов, навыков программирования на языке Pascal и создании приложений для операционных системы, полученных в ходе курса "объектно-ориентированное программирование".

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Архангельский, А.Я. Программирование в Delphi: Учебник по классическим версиям Delphi / А.Я. Архангельский. - М.: Бином-Пресс, 2014. - 816 с.
2. Архангельский, А.Я. Программирование в Delphi: Учебник по классическим версиям Delphi / А.Я. Архангельский. - М.: Бином, 2014. - 816 с.
3. Архангельский, А.Я. Программирование в Delphi для Windows. Версии 2006, 2007. Turbo Delphi / А.Я. Архангельский. - М.: Бином, 2014. - 1240 с.
4. Архангельский, А.Я. Программирование в Delphi для Windows. Версии 2006, 2007, Turbo Delphi / А.Я. Архангельский. - М.: Бином-Пресс, 2014. - 1248 с.
5. Белов, В.В. Программирование в Delphi: процедурное, объектно-ориентированное, визуальное: Учебное пособие для вузов / В.В. Белов, В.И. Чистякова. - М.: РиС, 2015. -240с.
6. Белов, В.В. Программирование в Delphi: процедурное, объектно-ориентированное, визуальное: Учебное пособие для вузов / В.В. Белов, В.И. Чистякова. - М.: ГЛТ , 2015.-240с.
7. Кузан, Д.Я. Программирование Win32 API в Delphi / Д.Я. Кузан. - СПб.: BHV, 2013.-368с.
8. Осипов, Д. Delphi. Профессиональное программирование / Д. Осипов. - СПб.: Символ-плюс,2016.-1056с.
9. Санников, Е. Курс практического программирования в Delphi. Объектно - ориентированное программирование / Е. Санников. - М.: Солон-пресс, 2014. - 188 с.
10. Фаронов, В. Delphi. Программирование на языке высокого уровня / В. Фаронов. - СПб.: Питер, 2014. - 640 с.