Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное
учреждениевысшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №8
по курсу «Программирование»

Знакомство со средой программирования Delphi

Выполнил студент группы ИВТ-11 _____/Рзаев А. Э./
Проверил преподаватель _____/Чистяков Г. А./

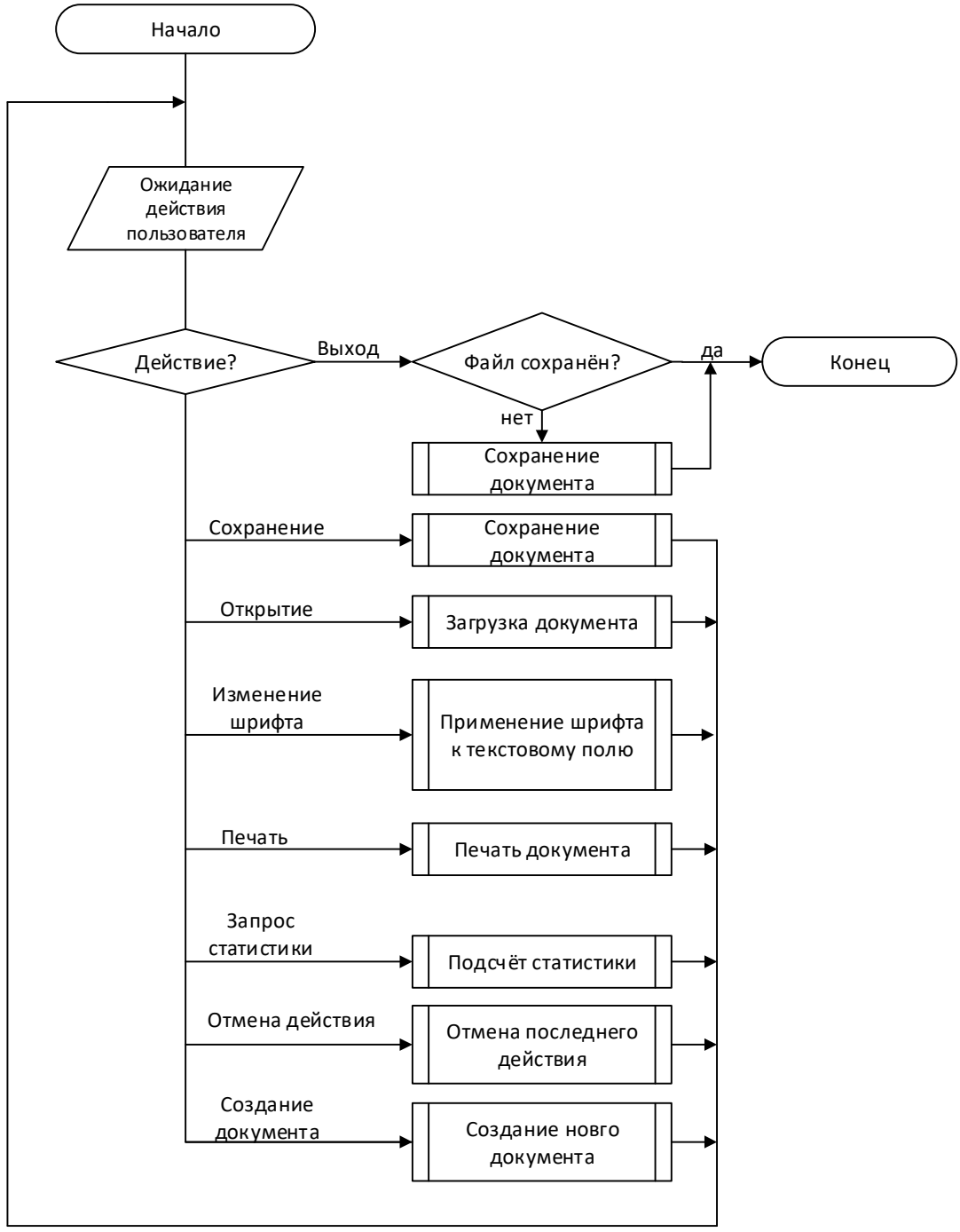
Киров 2016

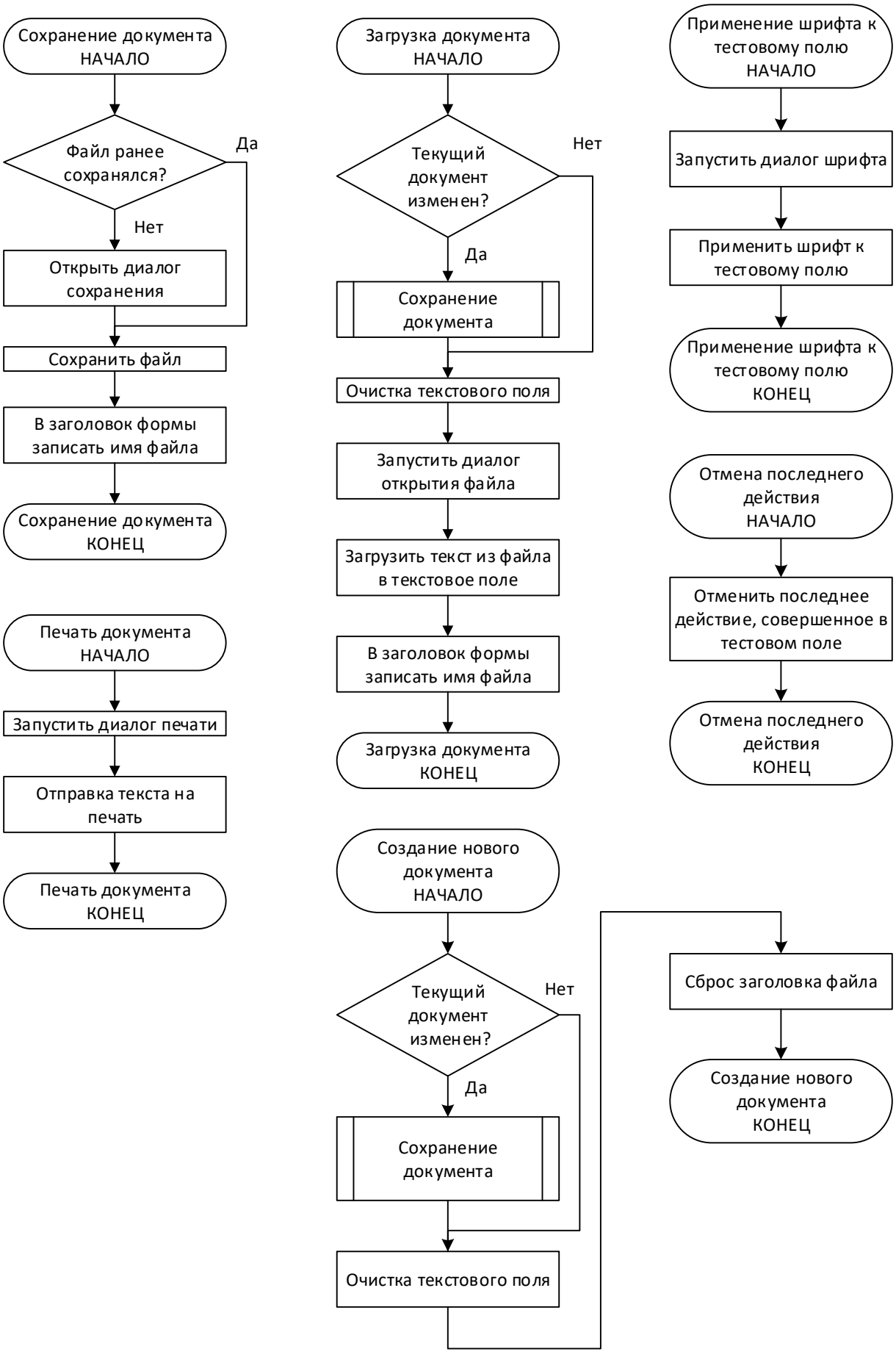
Цель работы: познакомиться с парадигмой событийно-ориентированного программирования, получить базовые навыки создания программ с графическим интерфейсом пользователя.

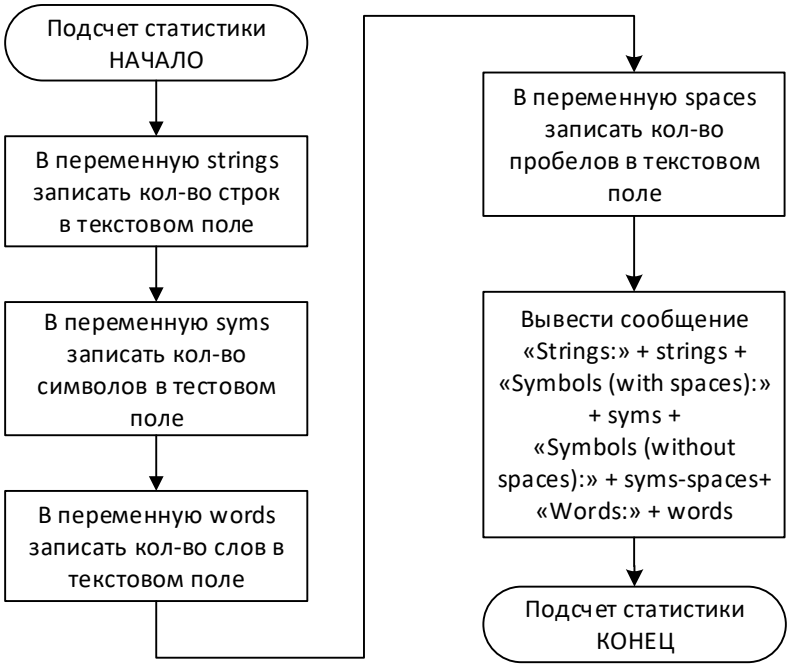
Задание:

1. Разработать программу текстовый редактор;
2. Реализовать возможности работы с начертанием шрифта, вывода текста на печать, подсчета статистики;
3. Реализация дополнительных возможностей будет оцениваться в индивидуальном порядке.

Схема работы программы:




Сохранение документа НАЧАЛО
Файл ранее сохранялся?
Да
Нет
Открыть диалог сохранения
Сохранить файл
В заголовок формы записать имя файла
Сохранение документа КОНЕЦ
Печать документа НАЧАЛО
Запустить диалог печати
Отправка текста на печать
Печать документа КОНЕЦ
Загрузка документа НАЧАЛО
Текущий документ изменен?
Нет
Да
Сохранение документа
Очистка текстового поля
Запустить диалог открытия файла
Загрузить текст из файла в текстовое поле
В заголовок формы записать имя файла
Загрузка документа КОНЕЦ
Создание нового документа НАЧАЛО
Текущий документ изменен?
Нет
Да
Сохранение документа
Очистка текстового поля
Сброс заголовка файла
Создание нового документа КОНЕЦ
Применение шрифта к тестовому полю НАЧАЛО
Запустить диалог шрифта
Применить шрифт к тестовому полю
Применение шрифта к тестовому полю КОНЕЦ
Отмена последнего действия НАЧАЛО
Отменить последнее действие, совершенное в тестовом поле
Отмена последнего действия КОНЕЦ



Листинг кода:

```
unit TextEditor;

{$mode objfpc}{$H+}{$M+}

interface

uses
  Classes, SysUtils, FileUtil, SynEdit, Forms, Controls, Graphics,
Dialogs,
  SynEditKeyCmds, PrintersDlgs, Menus, ComCtrls, RichMemo, StrUtils,
Printers;

type

  { TFile }

  TFile = class(TObject)
  public
    name : String;
    exist : Boolean;
    saved : Boolean;
    closed : Boolean;
    constructor Create;
    procedure Reset;
    procedure MakeExist(path : String);
  end;

type

  { TMainForm }

  TMainForm = class(TForm)
    ColorDialog: TColorDialog;
    FontDialog: TFontDialog;
```

```
    MainMenu: TMainMenu;
    ItemFileNew: TMenuItem;
    ItemFileOpen: TMenuItem;
    ItemFileSave: TMenuItem;
    ItemFileSaveAs: TMenuItem;
    ItemFilePrint: TMenuItem;
    ItemQuit: TMenuItem;
    ItemEditCut: TMenuItem;
    ItemEditCopy: TMenuItem;
    ItemEditPaste: TMenuItem;
    ItemViewStatusBar: TMenuItem;
    ItemEditUndo: TMenuItem;
    ItemEditSelectAll: TMenuItem;
    ItemFormatFont: TMenuItem;
    ItemFormatColor: TMenuItem;
    ItemFileStatistics: TMenuItem;
    MenuItemHelp: TMenuItem;
    MenuItemView: TMenuItem;
    MenuItemFormat: TMenuItem;
    MenuItemEdit: TMenuItem;
    MenuItemFile: TMenuItem;
    OpenDialog: TOpenDialog;
    PrintDialog: TPrintDialog;
    SaveDialog: TSaveDialog;
    StatusBar: TStatusBar;
    OpenFile : TFile;
    MainText: TSynEdit;
    //Printer: TSynEditPrint;
    procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
    procedure FormCreate(Sender: TObject);
    procedure ItemEditCopyClick(Sender: TObject);
    procedure ItemEditCutClick(Sender: TObject);
    procedure ItemEditPasteClick(Sender: TObject);
    procedure ItemEditSelectAllClick(Sender: TObject);
    procedure ItemEditUndoClick(Sender: TObject);
    procedure ItemFileNewClick(Sender: TObject);
    procedure ItemFileOpenClick(Sender: TObject);
    procedure ItemFilePrintClick(Sender: TObject);
    procedure ItemFileSaveAsClick(Sender: TObject);
    procedure ItemFileSaveClick(Sender: TObject);
    procedure ItemFileStatisticsClick(Sender: TObject);
    procedure ItemFormatColorClick(Sender: TObject);
    procedure ItemFormatFontClick(Sender: TObject);
    procedure ItemQuitClick(Sender: TObject);
    procedure ItemViewStatusBarClick(Sender: TObject);
    procedure MainTextChange(Sender: TObject);
    procedure MainTextClick(Sender: TObject);
    procedure MainTextKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState
      );
    procedure MainTextKeyUp(Sender: TObject; var Key: Word; Shift:
TShiftState);
  private
    { private declarations }
```

```
    procedure SaveNewFile;
    procedure SaveExistFile;
    procedure SaveFileAs;
    function SaveChanges : Boolean;
    procedure CloseExistFile;
    procedure OpenExistFile(path : String);
    procedure UpdateState;
  public
    { public declarations }
  end;

var
  MainForm: TMainForm;

implementation

{$R *.lfm}

{ TMainForm }

constructor TFile.Create;
begin
  name := '';
  saved := true;
  exist := false;
  inherited Create;
end;

procedure TFile.Reset;
begin
  name := '';
  exist := false;
  saved := true;
end;

procedure TFile.MakeExist(path : String);
begin
  name := path;
  exist := true;
  saved := true;
end;

function TMainForm.SaveChanges : Boolean;
var
  res : Longint;
begin
  if not OpenFile.saved then
  begin
    res := MessageDlg('Text editor', 'Do you want to save changes?',
                      mtConfirmation, [mbYes, mbNo, mbIgnore], 0);
    if res = mrYes then
    begin
      if OpenFile.exist then
        SaveExistFile
```

```
      else
        SaveNewFile;
      SaveChanges := OpenFile.saved;
    end
    else if res = mrNo then
    begin
      //CloseExistFile;
      SaveChanges := true;
    end
    else
      SaveChanges := false;
  end
  else
    SaveChanges := true;
  UpdateState;
end;

procedure TMainForm.SaveNewFile;
var
  path : String;
begin
  if SaveDialog.Execute then
  begin
    path := SaveDialog.FileName;
    OpenFile.MakeExist(path);
    MainText.Lines.SaveToFile(path);
    Caption := path;
  end;
end;

procedure TMainForm.SaveExistFile;
begin
  try
    MainText.Lines.SaveToFile(OpenFile.name);
    OpenFile.MakeExist(OpenFile.name);
  except
    SaveNewFile;
  end;
end;

procedure TMainForm.SaveFileAs;
var
  path : String;
begin
  if SaveDialog.Execute then
  begin
    path := SaveDialog.FileName;
    OpenFile.MakeExist(path);
    MainText.Lines.SaveToFile(path);
    Caption := path;
  end
end;

procedure TMainForm.OpenExistFile(path : String);
```

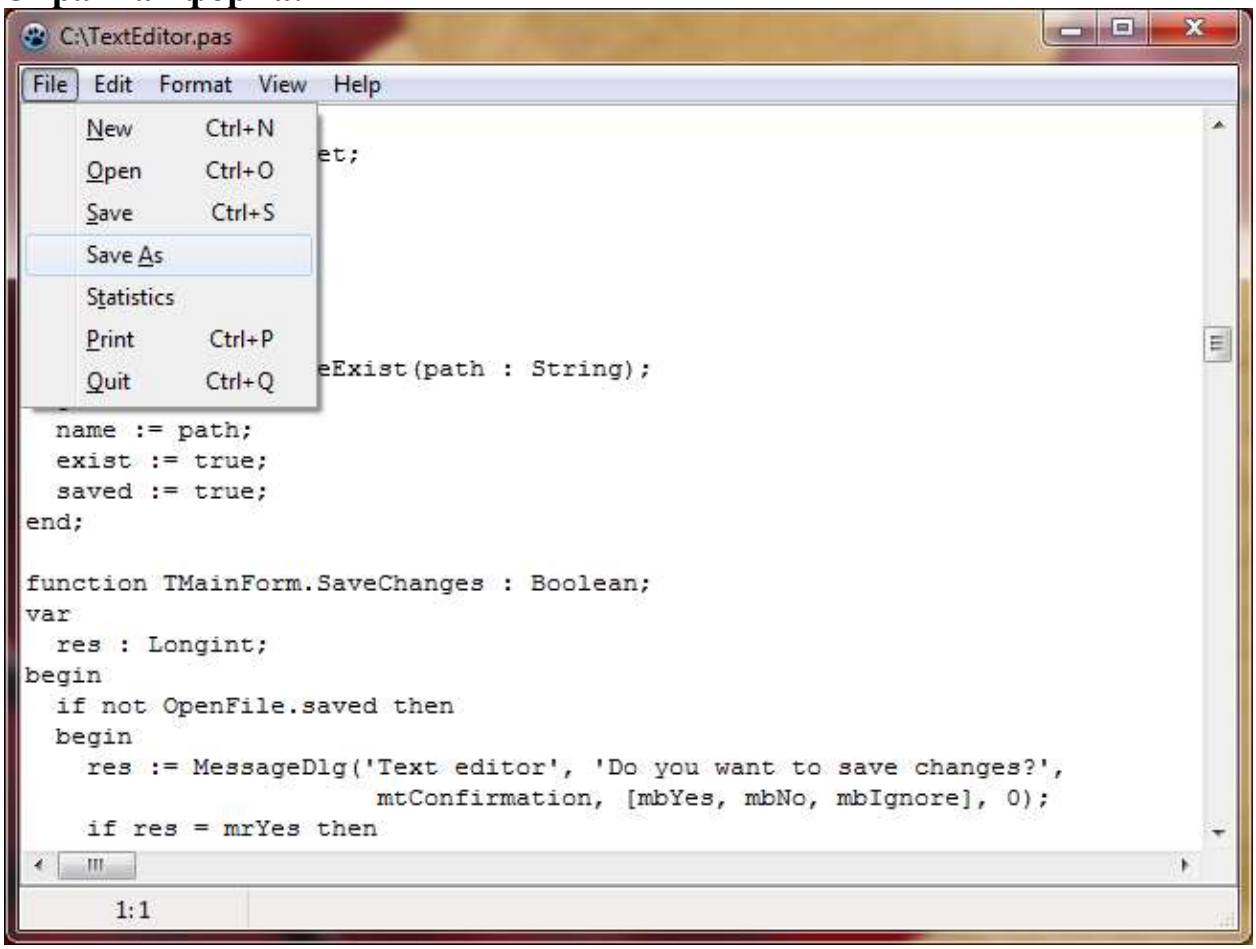
```
begin
  MainText.Lines.LoadFromFile(path);
  OpenFile.MakeExist(path);
  Caption := path;
end;

procedure TMainForm.CloseExistFile;
begin
  OpenFile.closed := true;
end;

procedure TMainForm.FormCreate(Sender: TObject);
begin
  Caption := 'New document';
  OpenFile := TFile.Create;
  UpdateState;
end;

procedure TMainForm.ItemEditCopyClick(Sender: TObject);
begin
  MainText.CopyToClipboard;
end;

procedure TMainForm.ItemEditCutClick(Sender: TObject);
begin
  MainText.CutToClipboard;
end;

procedure TMainForm.ItemEditPasteClick(Sender: TObject);
begin
  MainText.PasteFromClipboard;
end;

procedure TMainForm.ItemEditSelectAllClick(Sender: TObject);
begin
  MainText.SelectAll;
end;

procedure TMainForm.ItemEditUndoClick(Sender: TObject);
begin
  MainText.Undo;
end;

procedure TMainForm.FormCloseQuery(Sender: TObject; var CanClose:
boolean);
begin
  { Save changes }
  { Can we close app? }
  if SaveChanges then
    CanClose := true
  else
    CanClose := false;
end;
```

```
procedure TMainForm.ItemFileNewClick(Sender: TObject);
begin
  { Save changes }
  if SaveChanges then
  begin
    MainText.Lines.Clear;
    Caption := 'New document';
    OpenFile.Reset;
    UpdateState;
  end;
end;

procedure TMainForm.ItemFileOpenClick(Sender: TObject);
var
  path : String;
begin
  { Open file }
  if OpenFile.saved then
  begin
    if OpenDialog.Execute then
    begin
      path := OpenDialog.FileName;
      OpenExistFile(path);
    end;
  end
  else
  begin
    if not SaveChanges then
      exit;
    if OpenDialog.Execute then
    begin
      path := OpenDialog.FileName;
      OpenExistFile(path);
    end;
  end;
  UpdateState;
end;

procedure TMainForm.ItemFilePrintClick(Sender: TObject);
var
  pr : TPrinter;
  i : Longint;
begin
  if PrintDialog.Execute then
  begin
    pr := Printer;
    pr.BeginDoc;
    pr.Canvas.Font := MainText.Font;
    for i := 0 to MainText.Lines.Count do
    pr.Canvas.TextOut(300, 100 + i * 100, MainText.Lines.Strings[i]);
    pr.EndDoc;
  end;
end;
```

```
procedure TMainForm.ItemFileSaveAsClick(Sender: TObject);
begin
  SaveNewFile;
  UpdateState;
end;

procedure TMainForm.ItemFileSaveClick(Sender: TObject);
begin
  if OpenFile.exist then
    SaveExistFile
  else
    SaveNewFile;
  UpdateState;
end;

procedure TMainForm.ItemFileStatisticsClick(Sender: TObject);
var
  words, syms, spaces, strings, i, cnt : Longint;
  message : String;
begin
  words := 0;
  syms := 0;
  spaces := 0;
  strings := MainText.Lines.Count;
  for i := 0 to strings - 1 do
  begin
    inc(syms, Length(MainText.Lines.Strings[i]));
    cnt := WordCount(MainText.Lines.Strings[i], [' ']);
    inc(spaces, cnt - 1);
    inc(words, cnt);
  end;
  message := 'Words: ' + IntToStr(words) + chr(10) +
             'Symbols (with spaces): ' + IntToStr(syms) + chr(10) +
             'Symbols (without spaces): ' + IntToStr(syms - spaces) +
chr(10) +
             'Lines: ' + IntToStr(strings);
  MessageDlg('Statistics', message, mtInformation, [mbClose], 0);
end;

procedure TMainForm.ItemFormatColorClick(Sender: TObject);
begin
  if ColorDialog.Execute then
    MainText.Font.Color := ColorDialog.Color;
end;

procedure TMainForm.ItemFormatFontClick(Sender: TObject);
begin
  if FontDialog.Execute then
    MainText.Font := FontDialog.Font;
end;

procedure TMainForm.ItemQuitClick(Sender: TObject);
var
  CanClose : Boolean;
```

```
begin
  FormCloseQuery(Sender, CanClose);
  if CanClose then
    MainForm.Close;
end;

procedure TMainForm.ItemViewStatusBarClick(Sender: TObject);
begin
  ItemViewStatusBar.Checked := not ItemViewStatusBar.Checked;
  StatusBar.Visible := ItemViewStatusBar.Checked;
end;

procedure TMainForm.MainTextChange(Sender: TObject);
begin
  { File is changed }
  OpenFile.saved := false;
  UpdateState;
end;

procedure TMainForm.MainTextClick(Sender: TObject);
begin
  UpdateState;
end;

procedure TMainForm.MainTextKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  UpdateState;
end;

procedure TMainForm.MainTextKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  UpdateState;
end;

procedure TMainForm.UpdateState;
var
  p : TPoint;
begin
  p := MainText.LogicalToPhysicalPos(MainText.LogicalCaretXY);
  StatusBar.Panels.Items[0].Text := IntToStr(p.y) + ': ' +
IntToStr(p.x);
  if OpenFile.saved then
    StatusBar.Panels.Items[1].Text := ''
  else
    StatusBar.Panels.Items[1].Text := 'Changed';
end;

end.
```

Экранная форма:



Вывод: в данной лабораторной работе была изучена среда разработки Delphi, получены знания о создании программ с графическим интерфейсом и о взаимодействии с пользователем посредством этого интерфейса, а также получены знания о парадигме ООП.