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 №9
по курсу «Программирование»

Исследование алгоритмов сортировки данных на внешней памяти

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

Киров 2016

Цель работы: познакомиться с организацией стандартных диалоговых окон, изучить принципы работы с данными на внешней памяти, получить навыки работы с типизированными файлами.

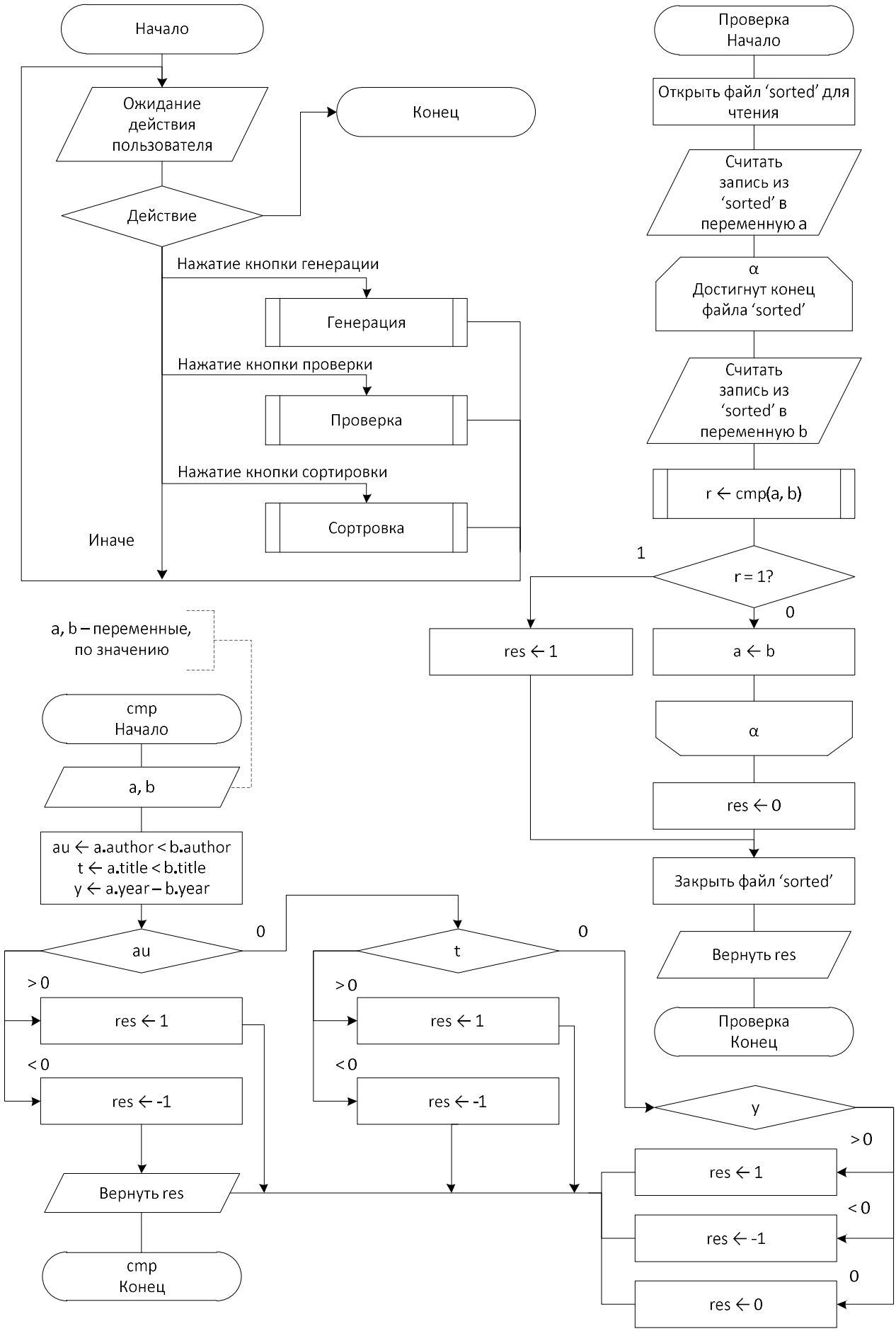
Задание:

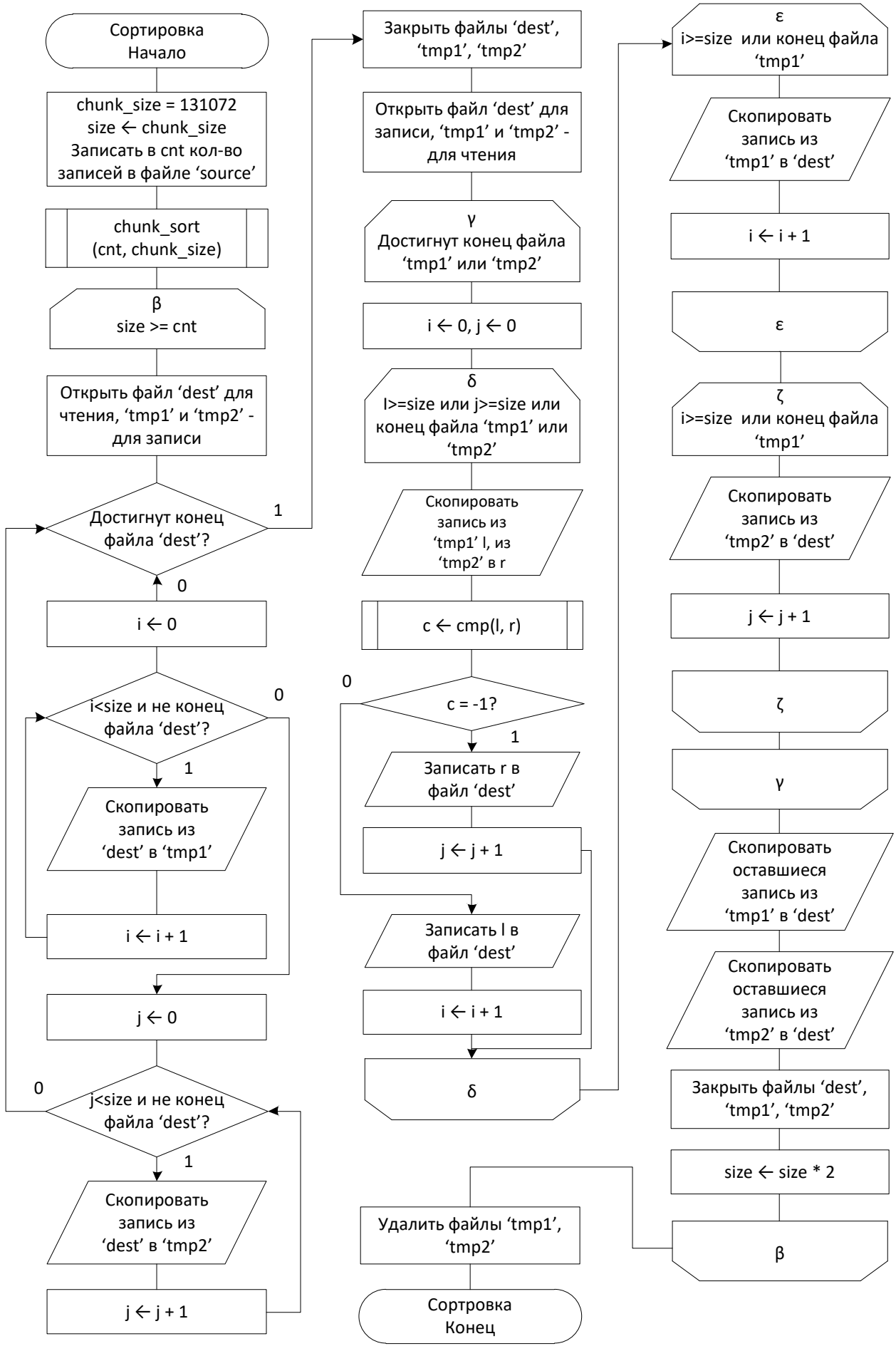
1. Разработать программу для сортировки данных, находящихся на жестком диске в типизированном файле.
2. Объем потребляемой оперативной памяти не должен превышать 10% от размера исходного файла.
3. Исходный и целевой файлы должны выбираться с помощью стандартных диалоговых окон.
4. Структуру сортируемого файла согласовать с преподавателем.

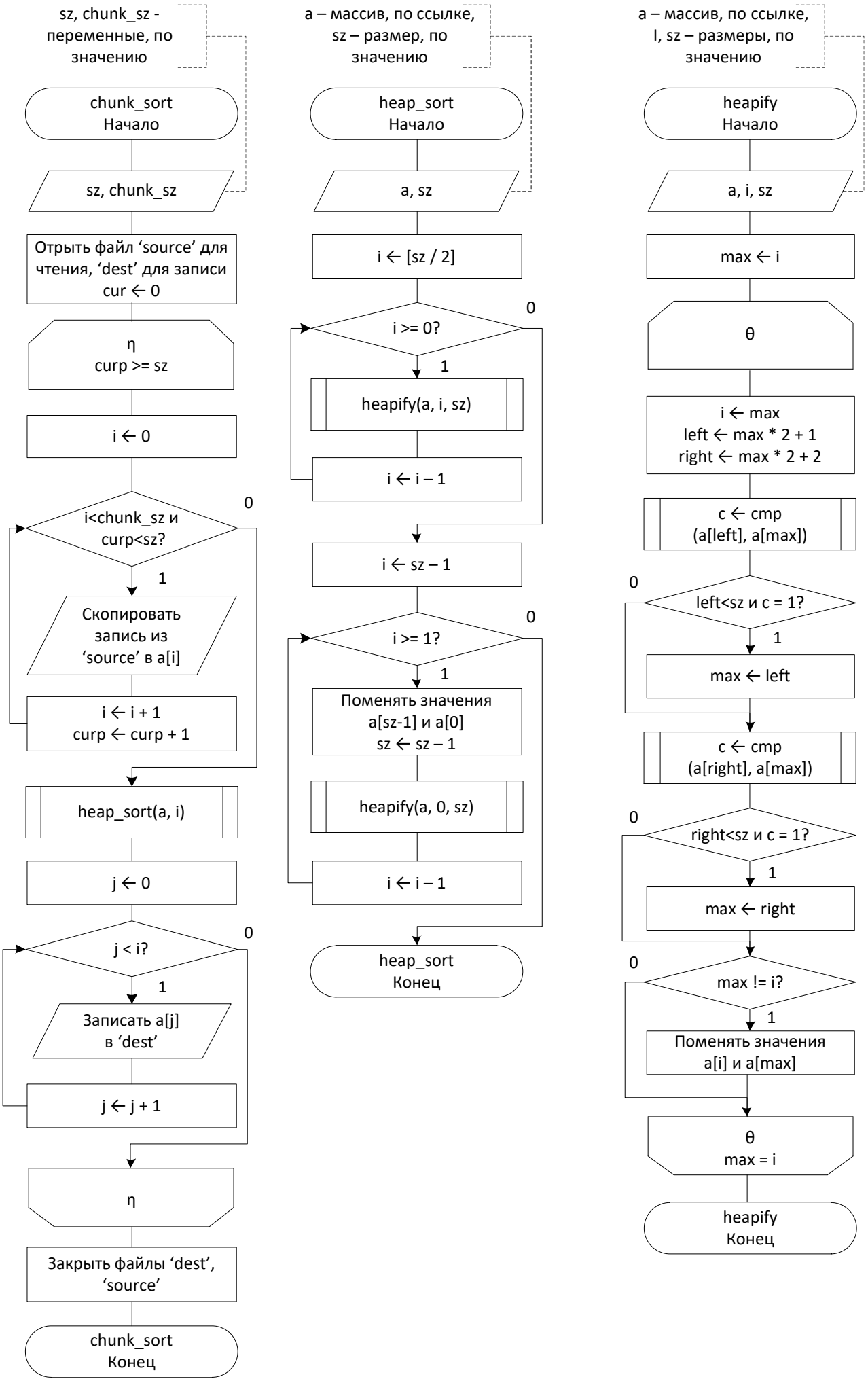
Структура:

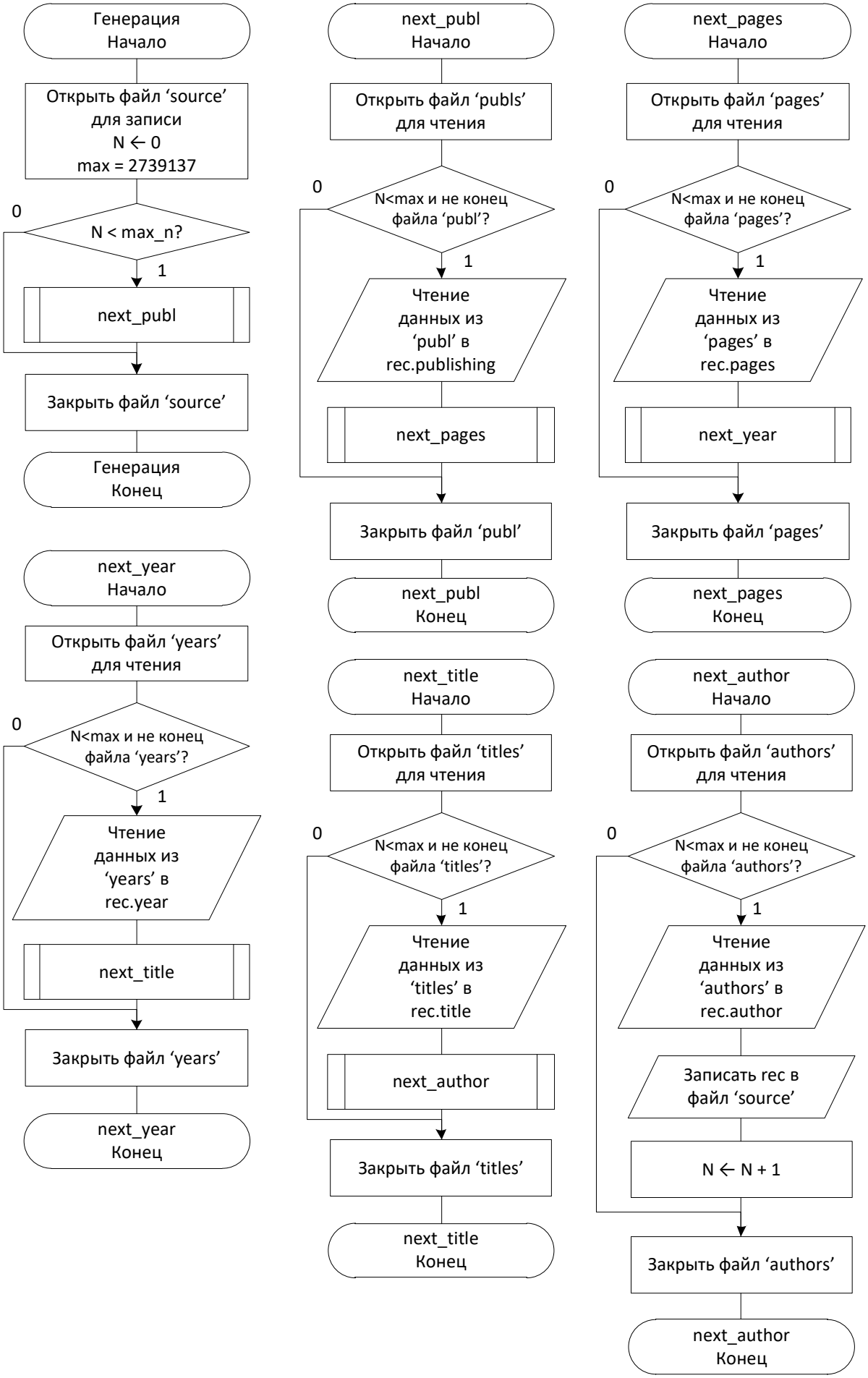
```
book = record
     title: string[200];
     author: string[150];
     publishing: string[30];
     year: integer;
     pages: longint;
end;
```

Схема работы программы:




Сортировка
Начало
chunk_size = 131072
size ← chunk_size
Записать в cnt кол-во записей в файле ‘source’
chunk_sort
(cnt, chunk_size)
β
size >= cnt
Открыть файл ‘dest’ для чтения, ‘tmp1’ и ‘tmp2’ - для записи
Достигнут конец файла ‘dest’?
1
0
i ← 0
i<size и не конец файла ‘dest’?
0
1
Скопировать запись из ‘dest’ в ‘tmp1’
i ← i + 1
j ← 0
j<size и не конец файла ‘dest’?
0
1
Скопировать запись из ‘dest’ в ‘tmp2’
j ← j + 1
Закрыть файлы ‘dest’, ‘tmp1’, ‘tmp2’
Открыть файл ‘dest’ для записи, ‘tmp1’ и ‘tmp2’ - для чтения
γ
Достигнут конец файла ‘tmp1’ или ‘tmp2’
i ← 0, j ← 0
δ
I>=size или j>=size или конец файла ‘tmp1’ или ‘tmp2’
Скопировать запись из ‘tmp1’ l, из ‘tmp2’ в r
c ← cmp(l, r)
0
c = -1?
1
Записать r в файл ‘dest’
j ← j + 1
Записать l в файл ‘dest’
i ← i + 1
δ
ε
i>=size или конец файла ‘tmp1’
Скопировать запись из ‘tmp1’ в ‘dest’
i ← i + 1
ε
ζ
i>=size или конец файла ‘tmp1’
Скопировать запись из ‘tmp2’ в ‘dest’
j ← j + 1
ζ
γ
Скопировать оставшиеся запись из ‘tmp1’ в ‘dest’
Скопировать оставшиеся запись из ‘tmp2’ в ‘dest’
Закрыть файлы ‘dest’, ‘tmp1’, ‘tmp2’
size ← size * 2
β
Удалить файлы ‘tmp1’, ‘tmp2’
Сортровка
Конец


sz, chunk_sz - переменные, по значению
chunk_sort Начало
sz, chunk_sz
Отрыть файл ‘source’ для чтения, ‘dest’ для записи
cur ← 0
η
curp >= sz
i ← 0
i<chunk_sz и curp<sz?
0
1
Скопировать запись из ‘source’ в a[i]
i ← i + 1
curp ← curp + 1
heap_sort(a, i)
j ← 0
j < i?
0
1
Записать a[j] в ‘dest’
j ← j + 1
η
Закрыть файлы ‘dest’, ‘source’
chunk_sort Конец
a – массив, по ссылке, sz – размер, по значению
heap_sort Начало
a, sz
i ← [sz / 2]
i >= 0?
0
1
heapify(a, i, sz)
i ← i – 1
i ← sz – 1
i >= 1?
0
1
Поменять значения a[sz-1] и a[0]
sz ← sz – 1
heapify(a, 0, sz)
i ← i – 1
heap_sort Конец
a – массив, по ссылке, l, sz – размеры, по значению
heapify Начало
a, i, sz
max ← i
θ
i ← max
left ← max * 2 + 1
right ← max * 2 + 2
c ← cmp (a[left], a[max])
left<sz и c = 1?
0
1
max ← left
c ← cmp (a[right], a[max])
right<sz и c = 1?
0
1
max ← right
max != i?
0
1
Поменять значения a[i] и a[max]
θ
max = i
heapify Конец


Генерация
Начало
Открыть файл ‘source’
для записи
N ← 0
max = 2739137
0
N < max_n?
1
next_publ
Закрыть файл ‘source’
Генерация
Конец
next_year
Начало
Открыть файл ‘years’
для чтения
0
N<max и не конец
файла ‘years’?
1
Чтение
данных из
‘years’ в
rec.year
next_title
Закрыть файл ‘years’
next_year
Конец
next_publ
Начало
Открыть файл ‘publs’
для чтения
0
N<max и не конец
файла ‘publ’?
1
Чтение
данных из
‘publ’ в
rec.publishing
next_pages
Закрыть файл ‘publ’
next_publ
Конец
next_title
Начало
Открыть файл ‘titles’
для чтения
0
N<max и не конец
файла ‘titles’?
1
Чтение
данных из
‘titles’ в
rec.title
next_author
Закрыть файл ‘titles’
next_title
Конец
next_pages
Начало
Открыть файл ‘pages’
для чтения
0
N<max и не конец
файла ‘pages’?
1
Чтение
данных из
‘pages’ в
rec.pages
next_year
Закрыть файл ‘pages’
next_pages
Конец
next_author
Начало
Открыть файл ‘authors’
для чтения
0
N<max и не конец
файла ‘authors’?
1
Чтение
данных из
‘authors’ в
rec.author
Записать rec в
файл ‘source’
N ← N + 1
Закрыть файл ‘authors’
next_author
Конец

Листинг кода:

Генератор

```
unit GenRecords;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, fileutil;

type
  TValue = record
    title: string[200];
    author: string[150];
    publishing: string[30];
    year: integer;
    pages: longint;
  end;

procedure next_title;
procedure next_author;
procedure next_publ;
procedure next_year;
procedure next_pages;
procedure generate(t, a, p, y, pg, d : String);

implementation

const max_n = 2739137;
var
  files : array [1..5] of Text;
  pathes : array [1..5] of String;
  dest : file of TValue;
  name : String;
  rec : TValue;
  N : Int64;

procedure generate(t, a, p, y, pg, d : String);
begin
  pathes[1] := t;
  pathes[2] := a;
  pathes[3] := p;
  pathes[4] := y;
  pathes[5] := pg;
  name := d;
  Assign(dest, name);
  Rewrite(dest);
  N := 0;
  next_publ;
  Close(dest);
end;

procedure next_title;
```

```
begin
  Assign(files[1], pathes[1]);
  Reset(files[1]);
  while (N < max_n) and (not eof(files[1])) do
  begin
    readln(files[1], rec.title);
    next_author;
  end;
  Close(files[1]);
end;

procedure next_author;
begin
  Assign(files[2], pathes[2]);
  Reset(files[2]);
  while (N < max_n) and (not eof(files[2])) do
  begin
    readln(files[2], rec.author);
    write(dest, rec);
    inc(N);
  end;
  Close(files[2]);
end;

procedure next_publ;
begin
  Assign(files[3], pathes[3]);
  Reset(files[3]);
  while (N < max_n) and (not eof(files[3])) do
  begin
    readln(files[3], rec.publishing);
    next_pages;
  end;
  Close(files[3]);
end;

procedure next_year;
begin
  Assign(files[4], pathes[4]);
  Reset(files[4]);
  while (N < max_n) and (not eof(files[4])) do
  begin
    readln(files[4], rec.year);
    next_title;
  end;
  Close(files[4]);
end;

procedure next_pages;
begin
  Assign(files[5], pathes[5]);
  Reset(files[5]);
  while (N < max_n) and (not eof(files[5])) do
  begin
```

```
    readln(files[5], rec.pages);
    next_year;
  end;
  Close(files[5]);
end;

end.
```

Основная программа

```
unit Sorting;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil;

type
    TValue = record
      title: string[200];
      author: string[150];
      publishing: string[30];
      year: integer;
      pages: longint;
    end;

    TArray = array of TValue;

    function cmp(a, b : TValue): Integer;
    procedure merge_sort(source, dest, tmp1, tmp2 : String);
    procedure swap(var a, b : TValue);
    procedure heapify(var a : TArray; i, sz : Int64);
    procedure make_heap(var a : TArray; sz : Int64);
    procedure heap_sort(var a : TArray; sz : Int64);
    function check_sorted(name : String): Boolean;
    procedure chunk_sort(name, tmp : String; sz, chunk_sz : Int64);

implementation

function cmp(a, b : TValue): Integer;
var
    t, au, y : Integer;
begin
    au := AnsiCompareStr(a.author, b.author);
    t := AnsiCompareStr(a.title, b.title);
    y := a.year - b.year;
    if au = 0 then
        if t = 0 then
            if y < 0 then
                cmp := -1
            else if y > 0 then
                cmp := 1
            else
```

```
                cmp := 0
            else if t < 0 then
                cmp := -1
            else
                cmp := 1
        else if au > 0 then
            cmp := 1
        else
            cmp := -1;
end;

procedure merge_sort(source, dest, tmp1, tmp2 : String);
const
    chunk_size = 131072;
var
    i, j, cnt, size : Int64;
    dest_f, tmp1_f, tmp2_f : file of TValue;
    tmp, l, r : TValue;
begin
    assign(tmp1_f, source);
    reset(tmp1_f);
    cnt := 0;
    // copy to dest
    while not eof(tmp1_f) do
    begin
        read(tmp1_f, tmp);
        inc(cnt);
    end;
    close(tmp1_f);

    chunk_sort(source, dest, cnt, chunk_size);
    size := chunk_size;
    while size < cnt do
    begin
        assign(dest_f, dest);
        reset(dest_f);
        assign(tmp1_f, tmp1);
        rewrite(tmp1_f);
        assign(tmp2_f, tmp2);
        rewrite(tmp2_f);
        // splitting
        while not eof(dest_f) do
        begin
            i := 0;
            while (i < size) and (not eof(dest_f)) do
            begin
                read(dest_f, tmp);
                write(tmp1_f, tmp);
                inc(i);
            end;
            j := 0;
            while (j < size) and (not eof(dest_f)) do
            begin
                read(dest_f, tmp);
```

```
        write(tmp2_f, tmp);
        inc(j);
    end;
end;
close(dest_f);
close(tmp1_f);
close(tmp2_f);

// merging
assign(dest_f, dest);
rewrite(dest_f);
assign(tmp1_f, tmp1);
reset(tmp1_f);
assign(tmp2_f, tmp2);
reset(tmp2_f);

while (not eof(tmp1_f)) and (not eof(tmp2_f)) do
begin
    i := 0; j := 0;
    while     (i < size)
          and (j < size)
          and (not eof(tmp1_f))
          and (not eof(tmp2_f)) do
    begin
        read(tmp1_f, l);
        read(tmp2_f, r);
        if cmp(r, l) = -1 then // r < l
        begin
            write(dest_f, r);
            seek(tmp1_f, FilePos(tmp1_f) - 1);
            inc(j);
        end
        else // l <= r
        begin
            write(dest_f, l);
            seek(tmp2_f, FilePos(tmp2_f) - 1);
            inc(i);
        end;
    end;
    while (i < size) and (not eof(tmp1_f)) do
    begin
        read(tmp1_f, l);
        write(dest_f, l);
        inc(i);
    end;
    while (j < size) and (not eof(tmp2_f)) do
    begin
        read(tmp2_f, r);
        write(dest_f, r);
        inc(j);
    end;
end;
while not eof(tmp1_f) do
begin
```

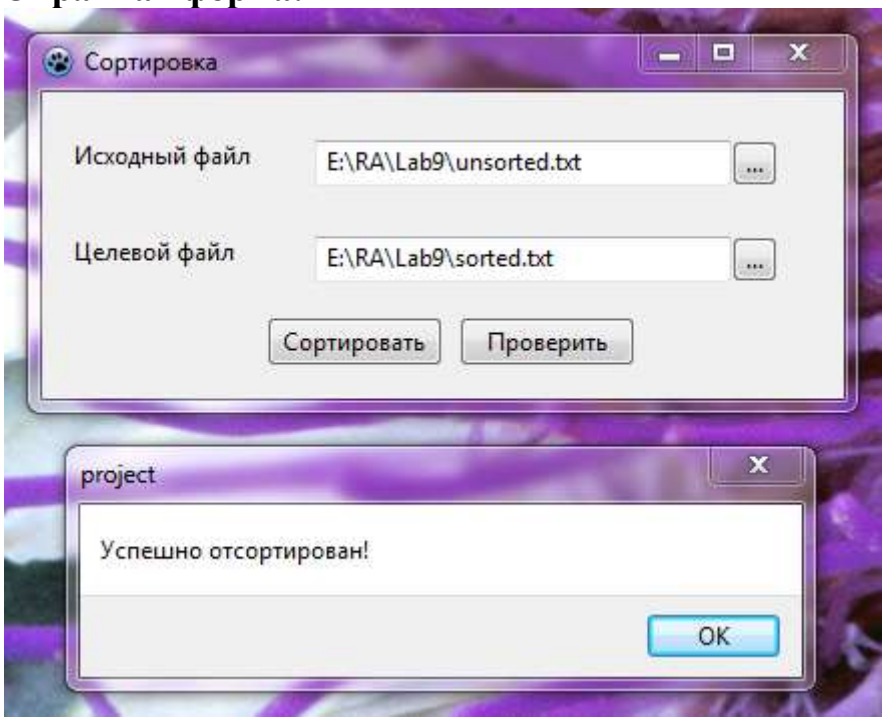
```
            read(tmp1_f, l);
            write(dest_f, l);
        end;
        while not eof(tmp2_f) do
        begin
            read(tmp2_f, r);
            write(dest_f, r);
        end;
        close(tmp1_f);
        close(tmp2_f);
        close(dest_f);
        //writeln(size);
        size := size * 2;
    end;
    erase(tmp1_f);
    erase(tmp2_f);
end;

procedure swap(var a, b : TValue);
var
    tmp : TValue;
begin
    tmp := a;
    a := b;
    b := tmp;
end;

procedure heapify(var a : TArray; i, sz : Int64);
var
    left, right, max : Int64;
begin
    max := i;
    repeat
        i := max;
        left := max * 2 + 1;
        right := max * 2 + 2;
        //max := i;
        if (left < sz) and (cmp(a[left], a[max]) = 1) then
            max := left;
        if (right < sz) and (cmp(a[right], a[max]) = 1) then
            max := right;
        if (max <> i) then
        begin
            swap(a[i], a[max]);
            //heapify(a, max, sz);
        end;
    until max = i;
end;

procedure make_heap(var a : TArray; sz : Int64);
var
    i : Int64;
begin
    i := sz div 2;
```

```
    while i >= 0 do
    begin
        heapify(a, i, sz);
        dec(i);
    end;
end;

procedure heap_sort(var a : TArray; sz : Int64);
var
    i : Int64;
begin
    make_heap(a, sz);
    i := sz - 1;
    while i >= 1 do
    begin
        swap(a[sz - 1], a[0]);
        dec(sz);
        heapify(a, 0, sz);
        dec(i);
    end;
end;

procedure chunk_sort(name, tmp : String; sz, chunk_sz : Int64);
var
    curp, i, j : Int64;
    dest_f, tmp_f : file of TValue;
    a : TArray;
begin
    assign(dest_f, name);
    reset(dest_f);
    assign(tmp_f, tmp);
    rewrite(tmp_f);
    SetLength(a, chunk_sz);
    curp := 0;
    while curp < sz do
    begin
        i := 0;
        while (i < chunk_sz) and (curp < sz) do
        begin
            read(dest_f, a[i]);
            inc(i);
            inc(curp);
        end;
        heap_sort(a, i);
        j := 0;
        while j < i do
        begin
            write(tmp_f, a[j]);
            inc(j);
        end;
    end;
    close(dest_f);
    close(tmp_f);
    //CopyFile(tmp, name);
```

```
end;

function check_sorted(name: String): Boolean;
var
    a, b : TValue;
    dest_f : file of TValue;
begin
    assign(dest_f, name);
    reset(dest_f);
    read(dest_f, a);
    check_sorted := true;
    while not eof(dest_f) do
    begin
        read(dest_f, b);
        if cmp(a, b) = 1 then
        begin
            check_sorted := false;
            break;
        end;
        a := b;
    end;
    close(dest_f);
end;

end.


unit MainWindow;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
EditBtn,
  StdCtrls, Sorting;

type

  { TMainForm }

  TMainForm = class(TForm)
    ButtonCheckFile: TButton;
    ButtonSortFile: TButton;
    EditSourceFileName: TEditButton;
    EditDestFileName: TEditButton;
    LabelSourceFile: TLabel;
    LabelDestFile: TLabel;
    OpenDialog: TOpenDialog;
    SaveDialog: TSaveDialog;
    procedure ButtonCheckFileClick(Sender: TObject);
    procedure ButtonSortFileClick(Sender: TObject);
    procedure EditDestFileNameButtonClick(Sender: TObject);
```

```
    procedure EditSourceFileNameButtonClick(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  MainForm: TMainForm;

implementation

{$R *.lfm}

{ TMainForm }

procedure TMainForm.ButtonSortFileClick(Sender: TObject);
const
    tmp1_f = 'tmp1.tmp';
    tmp2_f = 'tmp2.tmp';
begin
  merge_sort(EditSourceFileName.Text,
             EditDestFileName.Text,
             tmp1_f,
             tmp2_f);
  ShowMessage('Готово');
end;

procedure TMainForm.EditDestFileNameButtonClick(Sender: TObject);
begin
  if SaveDialog.Execute then
    EditDestFileName.Text := SaveDialog.FileName;
end;

procedure TMainForm.ButtonCheckFileClick(Sender: TObject);
begin
  if check_sorted(EditDestFileName.Text) then
    ShowMessage('Успешно отсортирован!')
  else
    ShowMessage('Ошибка при сортировке');
end;

procedure TMainForm.EditSourceFileNameButtonClick(Sender: TObject);
begin
  if OpenDialog.Execute then
    EditSourceFileName.Text := OpenDialog.FileName;
end;

end.
```

Экранная форма:



Вывод: в данной лабораторной работе были изучены принципы работы с данными на внешней памяти, получены навыки работы с типизированными файлами; изучена организация стандартных диалоговых окон.