МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**Вятский государственный университет**»

**(ФГБОУ ВО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра ЭВМ

Отчёт

### Лабораторная работа № 9 по дисциплине

«Программирование»

«Исследование алгоритмов сортировки данных на внешней памяти»

Выполнил студент группы ИВТб-1301 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Жеребцов К. А.

## Проверил преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Чистяков Г.А.

Киров 2021

**Задание:**

1. Разработать программу для сортировки данных, находящихся на жестком диске в типизированном файле.
2. Размер сортируемых данных должен быть не менее 1 ГБ. Объем потребляемой оперативной памяти не должен превышать 10% от размера исходного файла.
3. Исходный и целевой файл должны выбираться с помощью стандартных диалоговых окон.
4. Структуру сортируемого файла и ключ сортировки согласовать с преподавателем.
5. Помимо основной программы, требуется реализовать две вспомогательные утилиты – для генерации массива данных и проверки упорядоченности.

**Листинг:**

*Программа сортировки:*

unit unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls;

const

number = 31;

type

{ TForm1 }

TForm1 = class(TForm)

Check: TButton;

Label1: TLabel;

Label2: TLabel;

Start: TButton;

Last: TButton;

SortButton: TButton;

OpenDialog1: TOpenDialog;

procedure CheckClick(Sender: TObject);

procedure LastClick(Sender: TObject);

procedure SortButtonClick(Sender: TObject);

procedure StartClick(Sender: TObject);

private

public

end;

TDoc = record

Name: string[100];

day, month, year: integer;

money: integer;

end;

bTdoc = record

zapis: Tdoc;

bool: boolean;

end;

massTDoc = array of bTDoc;

fileTDoc = file of Tdoc;

mas = array [1..312500] of TDoc;

var

r: massTDoc;

a: TDoc;

PathStart: string;

PathLast: string;

FileStart, FileLast: File of TDoc;

fil: array [0..number] of file of TDoc;

zap, b: mas;

Form1: TForm1;

implementation

{$R \*.lfm}

{ TForm1 }

// процедура сравнения 2-х записей

function sravn(t1, t2: TDoc): boolean;

begin

if t1.year < t2.year then

begin

exit(false);

end

else

if t1.year > t2.year then

begin

exit(true);

end

else

if t1.year = t2.year then

begin

if t1.month < t2.month then

begin

exit(false);

end

else

if t1.month > t2.month then

begin

exit(true);

end

else

if t1.month = t2.month then

begin

if t1.day < t2.day then

begin

exit(false);

end

else

if t1.day > t2.day then

begin

exit(true);

end

else

if t1.day = t2.day then

begin

if t1.money <= t2.money then

begin

exit(false);

end

else

if t1.money > t2.money then

begin

exit(true);

end;

end;

end;

end;

end;

function NumoFile(var arr: massTDoc; var rr: TDoc):integer;

var i, k: integer;

begin

for i := 0 to length(arr) - 1 do begin

if arr[i].bool then begin

rr := arr[i].zapis;

k := i;

break;

end;

end;

for i:=1 to length(arr) - 1 do begin

if sravn(rr, arr[i].zapis) and arr[i].bool and (i <> k) then begin

rr := arr[i].zapis;

k := i;

end;

end;

NumoFile := k;

end;

// процедура слияния

procedure sliv;

var

f: fileTDoc;

i, j, num: longint;

begin

AssignFile(f, pathlast);

Rewrite(f);

for i := 0 to number do begin

Reset(fil[i]);

end;

setlength(r, number + 1);

for j := 0 to number do begin

read(fil[j], r[j].zapis);

r[j].bool := true;

end;

num := NumoFile(r, a);

write(f, a);

for i := 2 to 10000000 do begin

if not eof(fil[num]) then read(fil[num], r[num].zapis)

else if r[num].bool then r[num].bool := false;

num := NumoFile(r, a);

write(f, a);

end;

for i := 0 to number do begin

CloseFile(fil[i]);

DeleteFile('C:\Users\Кирилл\Desktop\ЛЕКЦИИ\LP\Программирование\Семестр 2\LAB 9\Files\F'+inttostr(i)+'.txt');

end;

CloseFile(f);

end;

//Процедура проверки упорядоченности

procedure CheckFile;

var

f: file of TDoc;

prov: string;

i, j: integer;

t1, t2: TDoc;

begin

Form1.Label2.Visible:= false;

Form1.OpenDialog1.Execute;

prov:= Form1.OpenDialog1.FileName;

assign(f,prov);

reset(f);

i:=0;

j:=0;

repeat

seek(f,i);

read(f,t1);

i:=i+1;

seek(f,i);

read(f,t2);

if t1.year < t2.year then

begin

j:=j+1;

end

else

if t1.year > t2.year then

begin

break;

end

else

if t1.year = t2.year then

begin

if t1.month < t2.month then

begin

j:=j+1;

end

else

if t1.month > t2.month then

begin

break;

end

else

if t1.month = t2.month then

begin

if t1.day < t2.day then

begin

j:=j+1;

end

else

if t1.day > t2.day then

begin

break;

end

else

if t1.day = t2.day then

begin

if t1.money <= t2.money then

begin

j:=j+1;

end

else

if t1.money > t2.money then

begin

break;

end;

end;

end;

end

until(i=(filesize(f)-1));

if j=(filesize(f)-1) then

begin

Form1.Label2.Caption:= 'Файл отсортирован.';

Form1.Label2.Visible:= True;

end

else

begin

Form1.Label2.Caption:= 'Файл не отсортирован.';

Form1.Label2.Visible:= True;

end;

close(f);

end;

// процедура сортировки

procedure QuickSort(left, right, c: integer);

var

newLeft, newRight : integer;

temp, pivot : TDoc;

begin

if c = 1 then begin

newLeft := left;

newRight := right;

pivot := zap[(left + right) div 2];

repeat

while zap[newLeft].year < pivot.year do begin

newLeft := newLeft + 1;

end;

while zap[newRight].year > pivot.year do

newRight := newRight - 1;

if newLeft <= newRight then

begin

temp := zap[newLeft];

zap[newLeft] := zap[newRight];

zap[newRight] := temp;

newLeft := newLeft + 1;

newRight := newRight - 1;

end;

until newLeft > newRight;

if left < newRight then

QuickSort(left, newRight, 1);

if newLeft < right then

QuickSort(newLeft, right, 1);

end

else

if c = 2 then begin

newLeft := left;

newRight := right;

pivot := zap[(left + right) div 2];

repeat

while zap[newLeft].month < pivot.month do begin

newLeft := newLeft + 1;

end;

while zap[newRight].month > pivot.month do

newRight := newRight - 1;

if newLeft <= newRight then

begin

temp.month := zap[newLeft].month;

zap[newLeft].month := zap[newRight].month;

zap[newRight].month := temp.month;

temp.day := zap[newLeft].day;

zap[newLeft].day := zap[newRight].day;

zap[newRight].day := temp.day;

temp.money := zap[newLeft].money;

zap[newLeft].money := zap[newRight].money;

zap[newRight].money := temp.money;

temp.Name := zap[newLeft].Name;

zap[newLeft].Name := zap[newRight].Name;

zap[newRight].Name := temp.Name;

newLeft := newLeft + 1;

newRight := newRight - 1;

end;

until newLeft > newRight;

if left < newRight then

QuickSort(left, newRight, 2);

if newLeft < right then

QuickSort(newLeft, right, 2);

end

else

if c = 3 then begin

newLeft := left;

newRight := right;

pivot := zap[(left + right) div 2];

repeat

while zap[newLeft].day < pivot.day do begin

newLeft := newLeft + 1;

end;

while zap[newRight].day > pivot.day do

newRight := newRight - 1;

if newLeft <= newRight then

begin

temp.day := zap[newLeft].day;

zap[newLeft].day := zap[newRight].day;

zap[newRight].day := temp.day;

temp.money := zap[newLeft].money;

zap[newLeft].money := zap[newRight].money;

zap[newRight].money := temp.money;

temp.Name := zap[newLeft].Name;

zap[newLeft].Name := zap[newRight].Name;

zap[newRight].Name := temp.Name;

newLeft := newLeft + 1;

newRight := newRight - 1;

end;

until newLeft > newRight;

if left < newRight then

QuickSort(left, newRight, 3);

if newLeft < right then

QuickSort(newLeft, right, 3);

end

else

if c = 4 then begin

newLeft := left;

newRight := right;

pivot := zap[(left + right) div 2];

repeat

while zap[newLeft].money < pivot.money do begin

newLeft := newLeft + 1;

end;

while zap[newRight].money > pivot.money do

newRight := newRight - 1;

if newLeft <= newRight then

begin

temp.money := zap[newLeft].money;

zap[newLeft].money := zap[newRight].money;

zap[newRight].money := temp.money;

temp.Name := zap[newLeft].Name;

zap[newLeft].Name := zap[newRight].Name;

zap[newRight].Name := temp.Name;

newLeft := newLeft + 1;

newRight := newRight - 1;

end;

until newLeft > newRight;

if left < newRight then

QuickSort(left, newRight, 4);

if newLeft < right then

QuickSort(newLeft, right, 4);

end;

end;

// процедура, которая выбирает поле сортировки

Procedure Sort(first, last: integer);

var

i, j:integer;

begin

j:= first;

i:= first;

repeat

while zap[i].year = zap[i+1].year do

inc(i);

QuickSort(j, i, 2);

inc(i);

j:=i;

until i > last;

j:= first;

i:= first;

repeat

while zap[i].month = zap[i+1].month do

inc(i);

QuickSort(j, i, 3);

inc(i);

j:=i;

until i > last;

j:= first;

i:= first;

repeat

while zap[i].day = zap[i+1].day do

inc(i);

QuickSort(j, i, 4);

inc(i);

j:=i;

until i > last;

end;

// обработка кнопки сортировки

procedure TForm1.SortButtonClick(Sender: TObject);

var

i, j: integer;

t: TDoc;

begin

Form1.Label1.Visible:=False;

AssignFile(FileStart, PathStart);

reset(FileStart);

for i:= 0 to number do begin

AssignFile(fil[i], 'C:\Users\Кирилл\Desktop\ЛЕКЦИИ\LP\Программирование\Семестр 2\LAB 9\Files\F'+inttostr(i)+'.txt');

rewrite(fil[i]);

end;

i:=0;

while not EOF(FileStart) do begin

while i <= number do begin

if not EOF(FileStart) then

read(FileStart, t)

else

break;

write(fil[i], t);

if i = number then

i:= -1;

Inc(i);

end;

end;

CloseFile(FileStart);

for i:=0 to number do begin

CloseFile(fil[i]);

end;

for i:=0 to number do begin

Reset(fil[i]);

while not EOF(fil[i]) do begin

for j:= 1 to 312500 do begin

read(fil[i], t);

zap[j]:= t;

end;

QuickSort(1, 312500, 1);

Sort(1, 312500);

end;

CloseFile(fil[i]);

rewrite(fil[i]);

for j:= 1 to 312500 do

begin

write(fil[i], zap[j]);

end;

CloseFile(fil[i]);

end;

sliv;

Form1.Label1.Visible:=True;

end;

// выбор целевого файла

procedure TForm1.LastClick(Sender: TObject);

begin

If OpenDialog1.Execute then

PathLast:= OpenDialog1.FileName;

end;

// обработка кнопки проверки упорядоченности

procedure TForm1.CheckClick(Sender: TObject);

begin

CheckFile;

end;

// выбор исходного файла

procedure TForm1.StartClick(Sender: TObject);

begin

If OpenDialog1.Execute then

PathStart:= OpenDialog1.FileName;

end;

end.

*Программа генерации массива данных:*

unit Unit1;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls;

type

{ TForm1 }

TForm1 = class(TForm)

Choose: TButton;

Generate: TButton;

Label1: TLabel;

OpenDialog1: TOpenDialog;

procedure ChooseClick(Sender: TObject);

procedure GenerateClick(Sender: TObject);

private

public

end;

TDoc = record

Name: string[100];

day, month, year: integer;

money: longint;

end;

var

Form1: TForm1;

path: string;

implementation

{$R \*.lfm}

{ TForm1 }

//выбор файла для шенерации

procedure TForm1.ChooseClick(Sender: TObject);

begin

if OpenDialog1.Execute then

path:= OpenDialog1.FileName;

end;

// генерация массива данных

procedure TForm1.GenerateClick(Sender: TObject);

const

char: array[1..26] of char = ('a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z');

var

Doc: Tdoc;

a: integer;

f: file of TDoc;

i,j: longint;

begin

randomize;

AssignFile(f, Path);

Rewrite(f);

For i := 1 to 10000000 do begin

a:= random(99)+1;

for j:= 1 to a do begin

Doc.Name[j]:= char[random(length(char))+1];

end;

Doc.day:= random(31)+1;

Doc.month:=random(12)+1;

Doc.year:=random(2021)+1;

Doc.money:= random(13999999)+1;

write(f, Doc);

end;

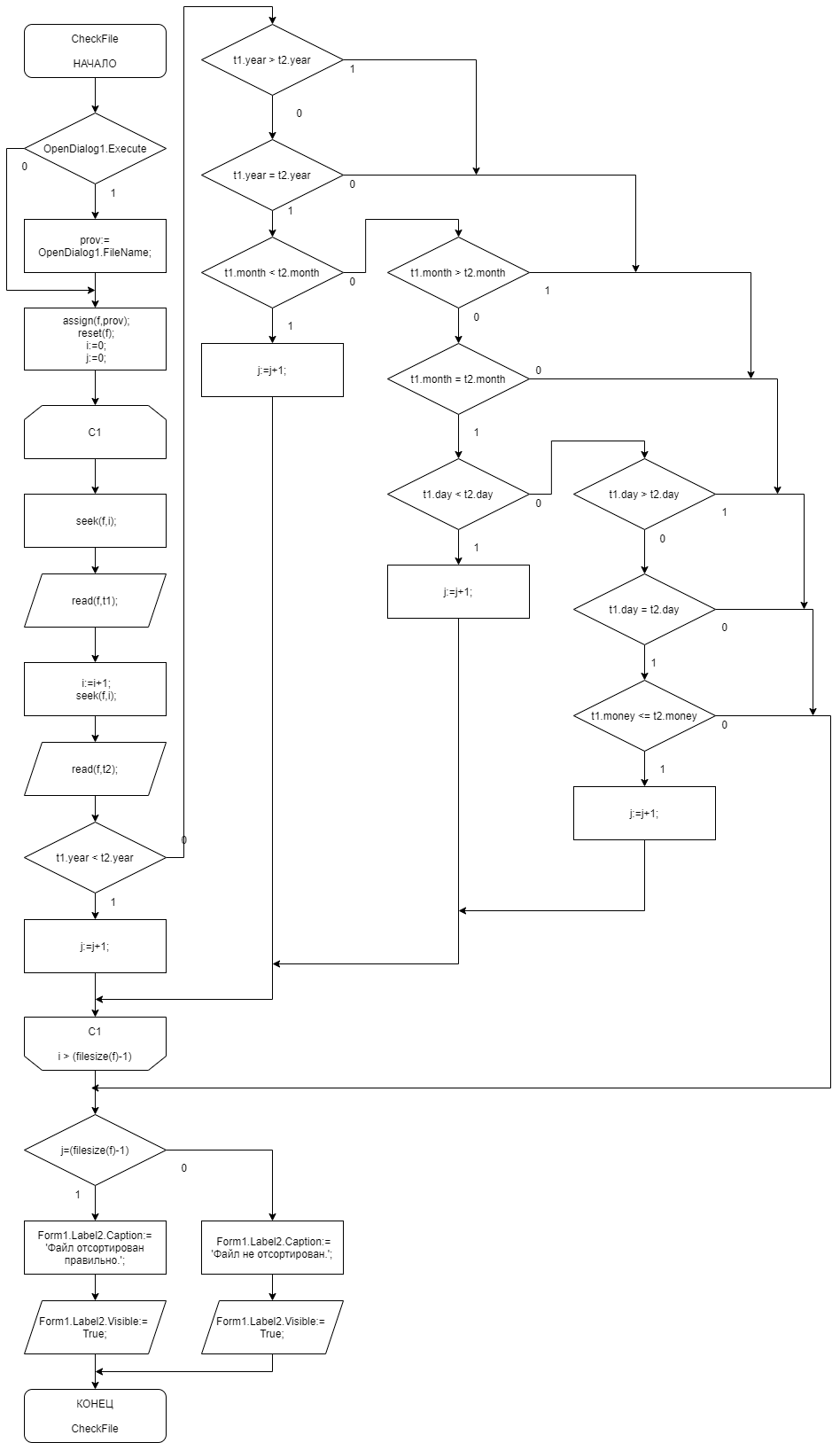
closefile(f);

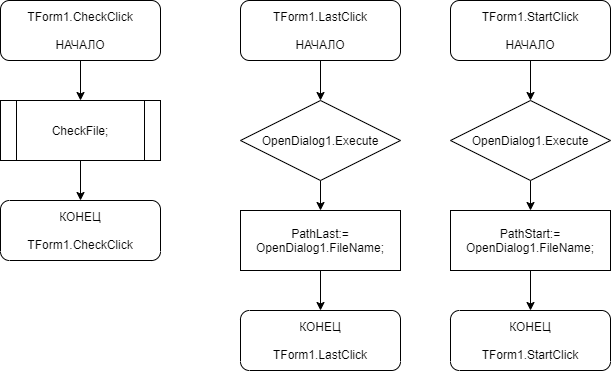
Form1.Label1.Visible:=True;

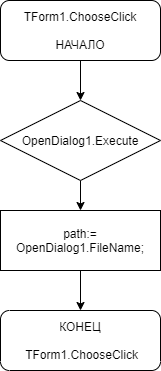
end;

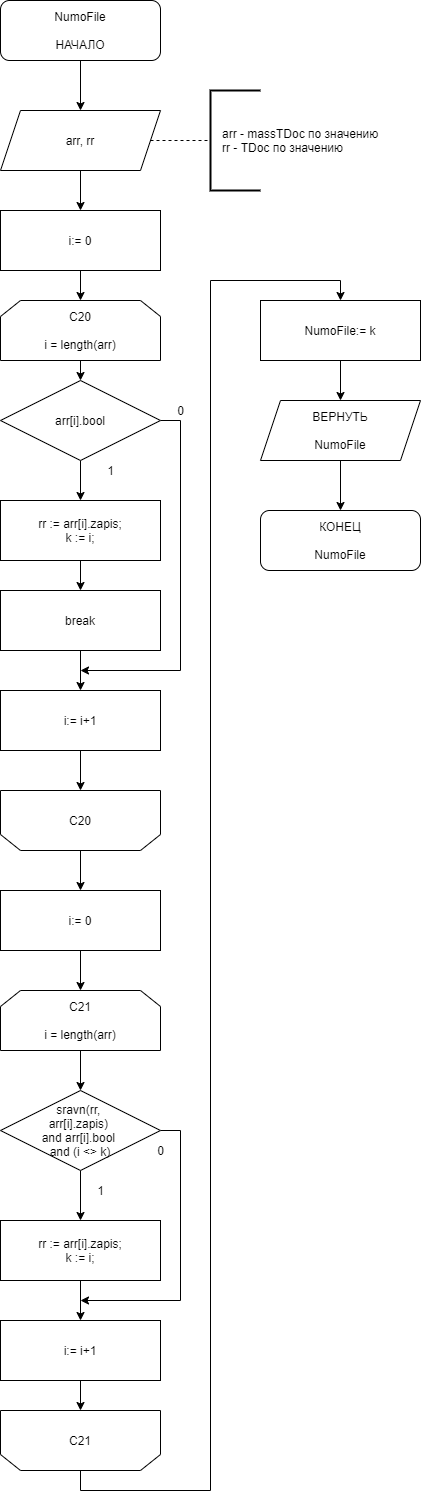
end.

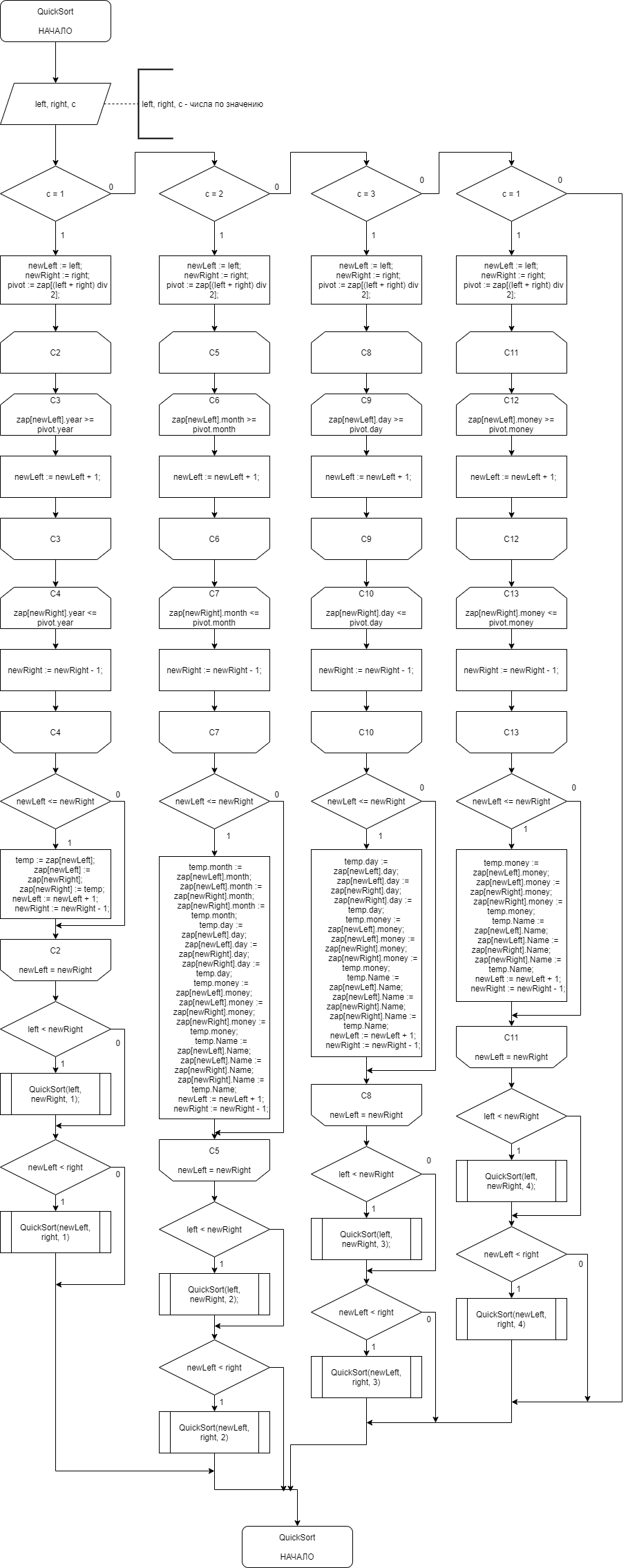
**Схемы алгоритма:**

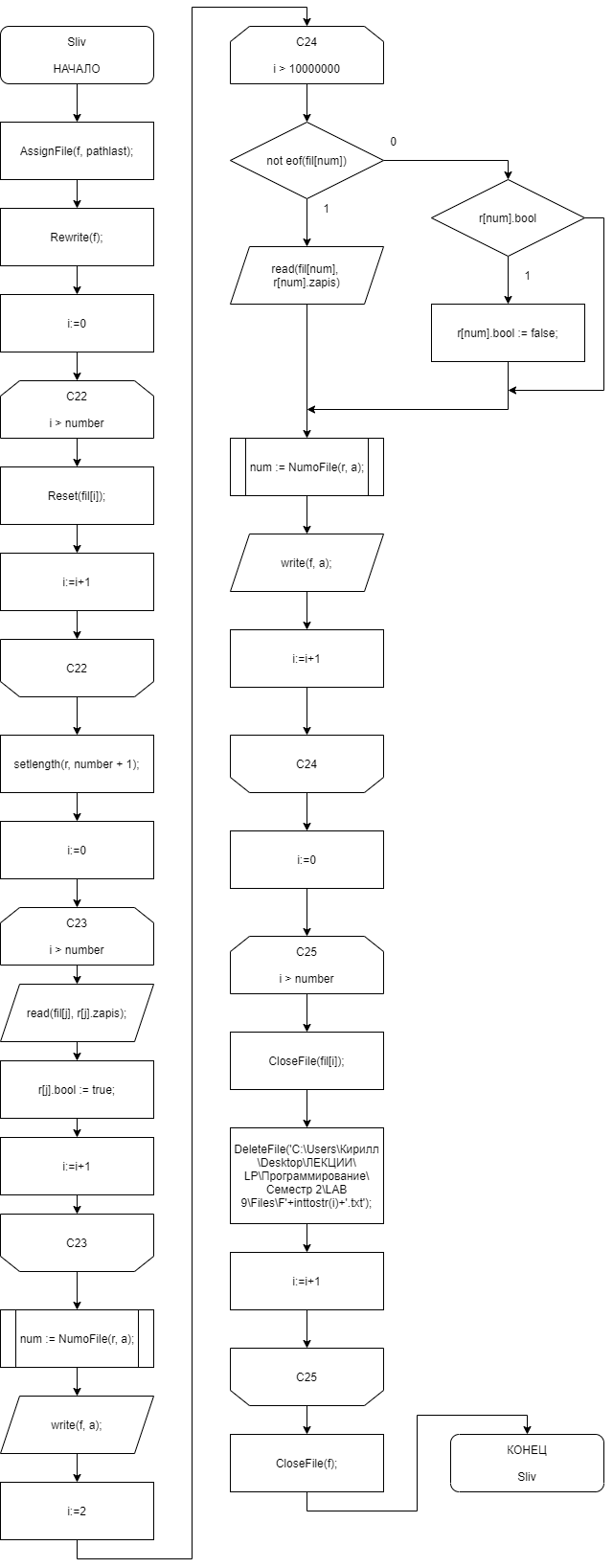


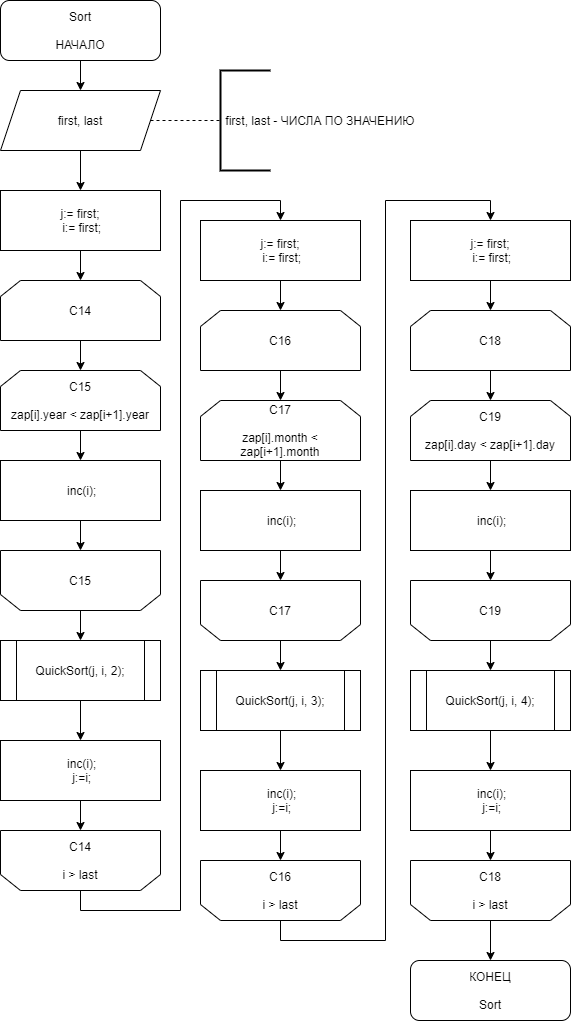


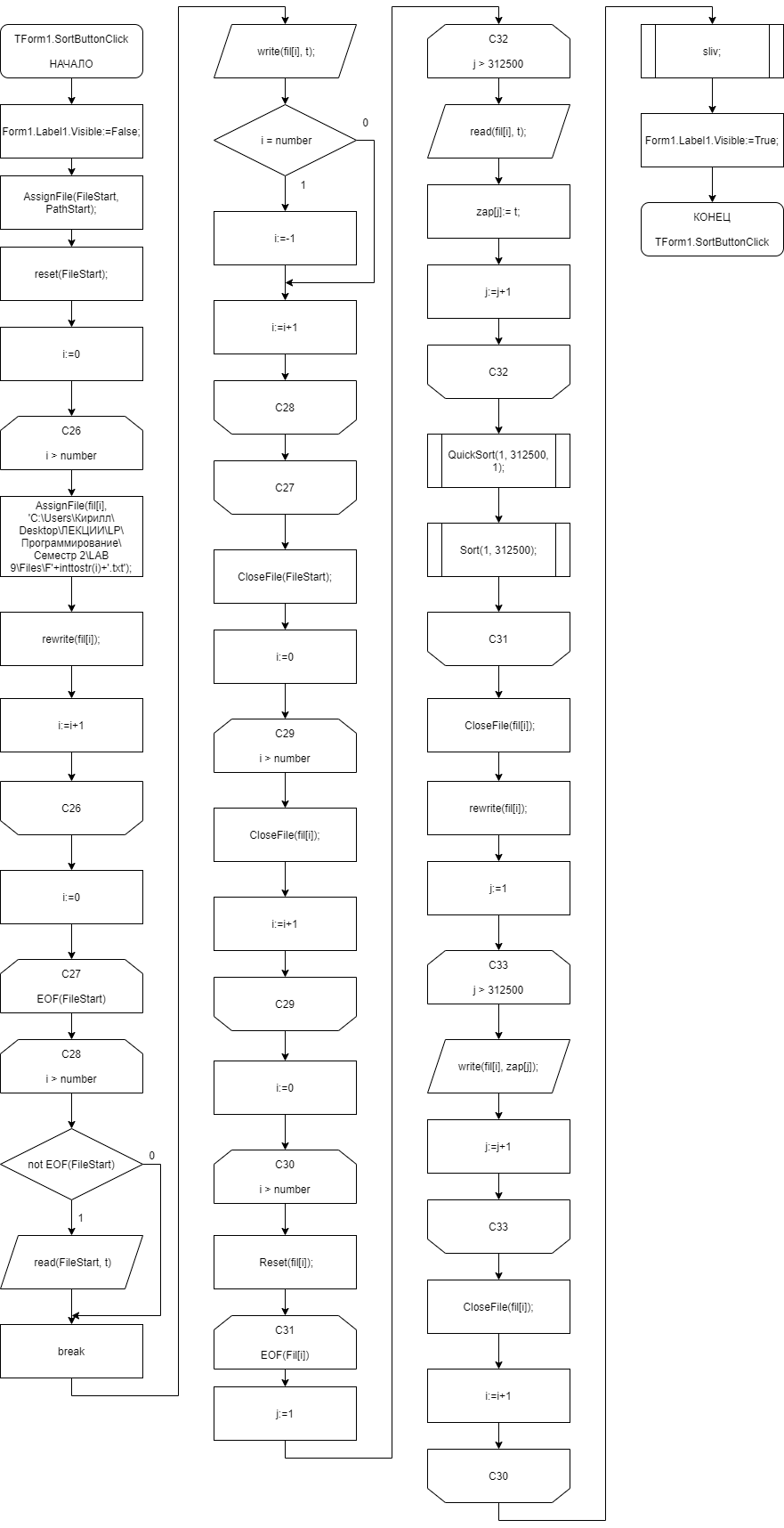


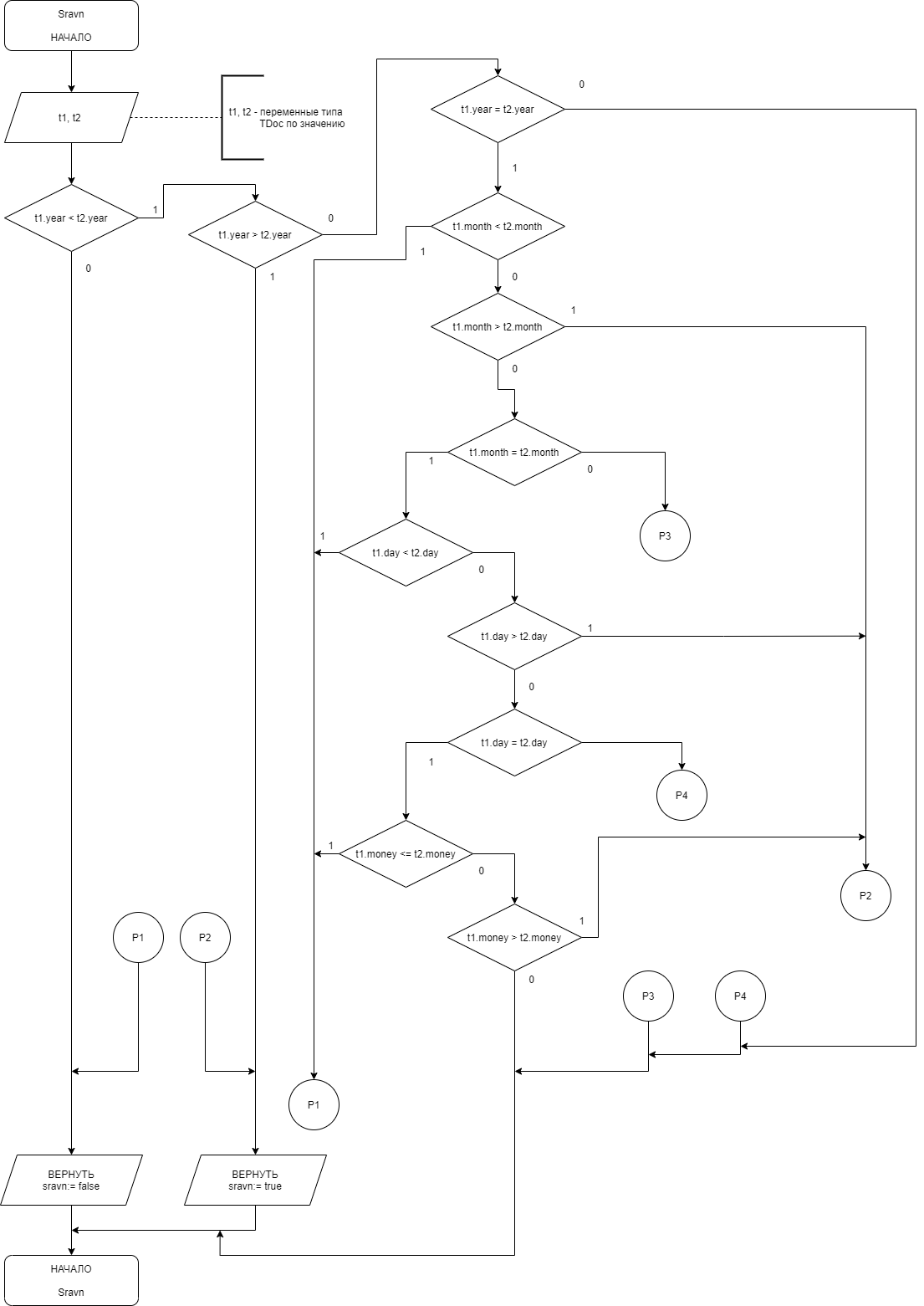


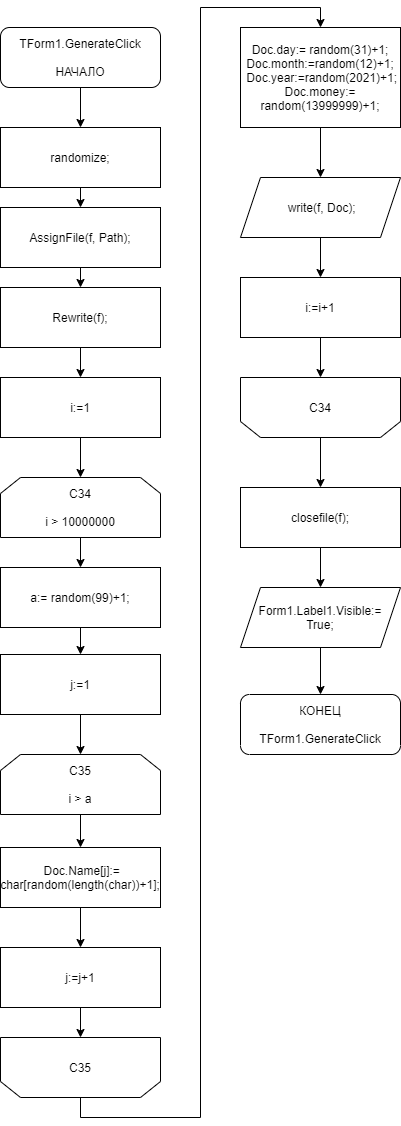




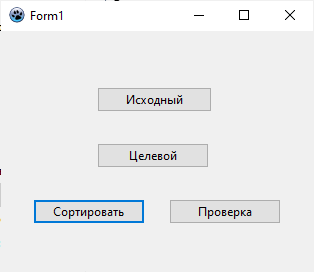


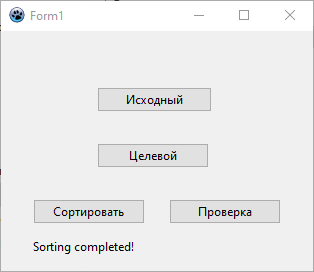


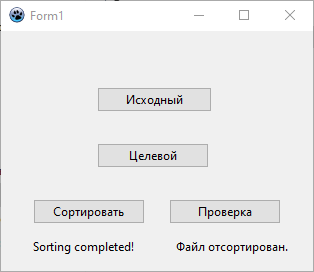




**Экранные формы:**





**Вывод:**

Была разработана программа для сортировки данных, находящихся на жестком диске в типизированном файле. Помимо основной программы были реализованы вспомогательные утилиты для генерации массива данных и проверки упорядоченности.

Были получены принципы работы с данными на внешней памяти и получены навыки работы с типизированными файлами.

Для совершения сортировки на внешней памяти исходный файл пришлось разбить на несколько вспомогательных файлов, каждый из которых был отсортирован отдельно. Далее было произведено слияние вспомогательных файлов в целевой.