УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.3

по предмету

Основы алгоритмизации и программирования

Вариант 3

Выполнил:

Бетеня К.С.

Проверила:

Данилова Г.В.

Группа 351005

Минск 2023

Задание:

Разработать алгоритм методом пошаговой детализации и программу, реализующую этот алгоритм.

Сортировка простыми вставками.

Код программы на **Delphi**:

Program lab3;

{$APPTYPE CONSOLE}

{$R \*.res}

Uses

System.SysUtils;

Const

FILE\_KEY: Integer = 1;

MIN\_ARR\_SIZE: Integer = 1;

CONSOLE\_KEY: Integer = 2;

MIN\_FILE\_WAY\_SIZE: Integer = 5;

Procedure ConditionOutput();

Begin

Write('The program is designed to sort an array', #13#10#9, 'using the simple

insertion method.', #13#10#10);

End;

Procedure PathConditionOutput();

Begin

Write('Where will we work through: ', #13#10#9, 'File: ', FILE\_KEY,

' Console: ', CONSOLE\_KEY, #13#10#10);

End;

Procedure FileRestriction();

Begin

Write(#13#10, '\*The first number is the number of elements ', #13#10, 'of the

array, and subsequent numbers of this array\*', #13#10);

End;

Function ChoosingAPath(): Integer;

Var

Path: Integer;

IsCorrect, IsCorrectInput: Boolean;

Begin

Path := 0;

IsCorrect := False;

IsCorrectInput := False;

PathConditionOutput();

Write('Please write were we should work: ');

Repeat

Try

Readln(Path);

IsCorrectInput := True;

Except

Write('Error. You should write a one natural number. Try again: ');

End;

If (Path = CONSOLE\_KEY) Or (Path = FILE\_KEY) Then

IsCorrect := True

Else

If IsCorrectInput Then

Write('Error method. Try again: ');

Until IsCorrect;

ChoosingAPath := Path;

End;

Function ArrSizeInputFromConsole(): Integer;

Var

ArrSize: Integer;

IsCorrect, IsCorrectInput: Boolean;

Begin

IsCorrect := False;

IsCorrectInput := False;

ArrSize := 0;

Write('Write your arr size: ');

Repeat

Try

Readln(ArrSize);

IsCorrectInput := True;

Except

Write('Invalid numeric input. Try again: ');

End;

If (ArrSize < MIN\_ARR\_SIZE) And IsCorrectInput Then

Write('Minimal arr size is: ', MIN\_ARR\_SIZE, '. Try again: ')

Else

If IsCorrectInput Then

IsCorrect := True;

Until IsCorrect;

ArrSizeInputFromConsole := ArrSize;

End;

Function InputCurrentNumbFromConsole(): Integer;

Var

CurrentNumb: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := False;

CurrentNumb := 0;

Repeat

Try

Readln(CurrentNumb);

IsCorrect := True;

Except

Write('Invalid numeric input. Try again: ');

End;

Until IsCorrect;

InputCurrentNumbFromConsole := CurrentNumb;

End;

Procedure ArrOfNumbInputFromConsole(Var ArrOfNumb: Array Of Integer);

Var

I: Integer;

Begin

For I := 0 To High(ArrOfNumb) Do

Begin

Write('Write your ', I + 1, ' numbers: ');

ArrOfNumb[I] := InputCurrentNumbFromConsole();

End;

End;

Function WayCondition(Way: String): Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := False;

If ExtractFileExt(Way) <> '.txt' Then

Write('Write .txt file. Try again: ')

Else

IsCorrect := True;

WayCondition := IsCorrect;

End;

Function InputWayToTheFile(): String;

Var

Way: String;

IsCorrect: Boolean;

Begin

IsCorrect := False;

Repeat

Readln(Way);

IsCorrect := WayCondition(Way);

Until IsCorrect;

InputWayToTheFile := Way;

End;

Function InputFileWay(): String;

Var

FileWay: String;

IsCorrect: Boolean;

Begin

IsCorrect := False;

FileWay := '';

Repeat

FileWay := InputWayToTheFile();

If (Not FileExists(FileWay)) Then

Write('Can not open a file. Try write another way: ')

Else

IsCorrect := True;

Until IsCorrect;

InputFileWay := FileWay;

End;

Function ArrSizeInputFromFile(ArrSize: Integer): Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := False;

If ArrSize < MIN\_ARR\_SIZE Then

Write('Minimal arr size is ', MIN\_ARR\_SIZE, '. Try again:')

Else

IsCorrect := True;

ArrSizeInputFromFile := IsCorrect;

End;

Function IsIncorrectArrOfNumbInputFromFile(Var ArrOfNumb: Array Of Integer; ArrSize: Integer; Var MyFile: TextFile): Boolean;

Var

IsCorrect: Boolean;

I: Integer;

Begin

IsCorrect := True;

For I := 0 To ArrSize - 1 Do

Begin

Try

Read(MyFile, ArrOfNumb[I]);

Except

IsCorrect := False;

End;

End;

IsIncorrectArrOfNumbInputFromFile := IsCorrect;

End;

Procedure SortMassive(Var ArrOfNumb: Array Of Integer);

Var

Temp, I, J: Integer;

Begin

For I := 1 To High(ArrOfNumb) Do

Begin

Temp := ArrOfNumb[I];

J := I - 1;

While (J >= 0) And (ArrOfNumb[J] > Temp) Do

Begin

ArrOfNumb[J + 1] := ArrOfNumb[J];

ArrOfNumb[J] := Temp;

Dec(J);

End;

End;

End;

Procedure OutputFromFile(Var ArrOfNumb: Array Of Integer);

Var

IsCorrect: Boolean;

FileWay: String;

MyFile: TextFile;

I: Integer;

Begin

IsCorrect := False;

Write('Write way to your file: ');

Repeat

FileWay := InputFileWay();

AssignFile(MyFile, FileWay);

Try

Try

Append(MyFile);

ReWrite(MyFile);

For I := 0 To High(ArrOfNumb) Do

Begin

Write(MyFile, ArrOfNumb[I]);

Write(MyFile, ' ');

End;

Write('Cheack your file.');

IsCorrect := True;

Finally

Close(MyFile);

End;

Except

Write('Bad output file. Try again: ');

End;

Until IsCorrect;

End;

Procedure OutputFromConsole(Var ArrOfNumb: Array Of Integer);

Var

I: Integer;

Begin

For I := 0 To High(ArrOfNumb) Do

Write(ArrOfNumb[I], ' ');

End;

Procedure ResultOutput(Var ArrOfNumb: Array Of Integer);

Var

Path: Integer;

Begin

Writeln('You need to choose where to write information from.');

Path := ChoosingAPath();

If Path = CONSOLE\_KEY Then

OutputFromConsole(ArrOfNumb)

Else

OutputFromFile(ArrOfNumb);

End;

Var

IsCorrect: Boolean;

ArrSize, Path: Integer;

ArrOfNumb: Array Of Integer;

FileWay: String;

MyFile: TextFile;

Begin

ArrSize := 0;

ConditionOutput();

Writeln(#13#10, 'You need to choose where to read information from.');

Path := ChoosingAPath();

If Path = CONSOLE\_KEY Then

Begin

ArrSize := ArrSizeInputFromConsole();

SetLength(ArrOfNumb, ArrSize);

ArrOfNumbInputFromConsole(ArrOfNumb);

End

Else

Begin

FileRestriction();

Write('Write way to your file: ');

Repeat

IsCorrect := True;

FileWay := InputFileWay();

AssignFile(MyFile, FileWay);

Try

Try

Reset(MyFile);

If Not Eof(MyFile) Then

Begin

Try

Readln(MyFile, ArrSize);

Except

Write('Error in array size reading. Try again: ');

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := ArrSizeInputFromFile(ArrSize);

If IsCorrect Then

Begin

SetLength(ArrOfNumb, ArrSize);

IsCorrect := IsIncorrectArrOfNumbInputFromFile

(ArrOfNumb, ArrSize, MyFile);

If not IsCorrect Then

Write('Invalid massive elements input.

Try again: ');

End;

End

Else

Write('Empty file. Try again: ');

Finally

Close(MyFile);

End;

Except

Write('Bad input file. Try again: ');

End;

Until IsCorrect;

End;

SortMassive(ArrOfNumb);

ResultOutput(ArrOfNumb);

ArrOfNumb := Nil;

End.

Код программы на **C++**:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

const int FILE\_KEY = 1;

const int MIN\_ARR\_SIZE = 1;

const int CONSOLE\_KEY = 2;

const int MIN\_FILE\_WAY\_SIZE = 5;

void conditionOutput()

{

cout << "The program is designed to sort an array\n\t"

"using the simple insertion method.\n\n";

}

void pathConditionOutput()

{

cout << "Where will we work through: \n\tFile: " <<

FILE\_KEY << " Console: " << CONSOLE\_KEY << endl << endl;

}

void fileRestriction()

{

cout << "\n\*The first number is the number of elements \n"

"of the array, and subsequent numbers of this array\*\n";

}

int choosingAPath()

{

int path = 0;

bool isIncorrect = true;

pathConditionOutput();

cout << "Please write were we should work: ";

do

{

cin >> path;

if (cin.fail() || cin.get() != '\n')

{

cerr << "Error. You should write a one natural number. Try

again: ";

cin.clear();

while (cin.get() != '\n');

}

else

{

if (path == CONSOLE\_KEY || path == FILE\_KEY)

isIncorrect = false;

else cerr << "Error method. Try again: ";

}

} while (isIncorrect);

return path;

}

/// input from console

int arrSizeInputFromConsole()

{

int arrSize;

bool isIncorrect = true;

cout << "Write your arr size: ";

do

{

cin >> arrSize;

if (cin.fail() || cin.get() != '\n')

{

cout << "Invalid numeric input. Try again: ";

cin.clear();

while (cin.get() != '\n');

}

else if (arrSize < MIN\_ARR\_SIZE)

cout << "Minimal arr size is: " << MIN\_ARR\_SIZE <<

". Try again: ";

else

isIncorrect = false;

} while (isIncorrect);

return arrSize;

}

int inputCurrentNumbFromConsole()

{

int currentNumb;

bool isIncorrect = true;

do

{

cin >> currentNumb;

if (cin.fail() || cin.get() != '\n')

{

cout << "Invalid numeric input. Try again: ";

cin.clear();

while (cin.get() != '\n');

}

else

isIncorrect = false;

} while (isIncorrect);

return currentNumb;

}

void arrOfNumbInputFromConsole(int arrSize, int\*& arrOfNumb)

{

for (int i = 0; i < arrSize; i++)

{

cout << "Write your " << i + 1 << " number: ";

arrOfNumb[i] = inputCurrentNumbFromConsole();

}

}

/// input from file

bool isCanOpenFile(string way, ios\_base::openmode mode)

{

fstream file(way, mode);

file.close();

return file.good();

}

bool wayCondition(string way)

{

if (way.size() < MIN\_FILE\_WAY\_SIZE)

{

cerr << "The path is too short. Try again: ";

return false;

}

string bufstr = way.substr(way.size() - 4);

if (bufstr != ".txt")

{

cerr << "Write .txt file. Try again: ";

return false;

}

return true;

}

string inputWayToTheFile()

{

string way;

bool isIncorrect;

do

{

cin >> way;

isIncorrect = !wayCondition(way);

} while (isIncorrect);

return way;

}

string inputFile()

{

string fileWay;

bool isIncorrect = true;

do

{

bool isCorrect = true;

fileWay = inputWayToTheFile();

if (!isCanOpenFile(fileWay, ios::in))

cout << "Can't open a file. Try write another way: ";

else

isIncorrect = false;

} while (isIncorrect);

return fileWay;

}

bool arrSizeInputFromFile(ifstream& file, int arrSize)

{

bool isIncorrect = true;

if (file.fail() || file.get() != '\n')

cout << "Error in input size of massive. Try again: ";

else if (arrSize < MIN\_ARR\_SIZE)

cout << "Minimal arr size is " << MIN\_ARR\_SIZE << ". Try again: ";

else

isIncorrect = false;

return isIncorrect;

}

bool isIncorrectInputCurrentNumbFromFile(ifstream& file)

{

bool isIncorrect = false;

if (file.fail())

isIncorrect = true;

return isIncorrect;

}

bool isIncorrectArrOfNumbInputFromFile(ifstream& file, int\*& arrOfNumb,

int arrSize)

{

bool isIncorrect = false;

for (int i = 0; i < arrSize; i++)

{

file >> arrOfNumb[i];

isIncorrect = isIncorrectInputCurrentNumbFromFile(file);

}

return isIncorrect;

}

/// result

void sortMassive(int\*& arrOfNumb, int arrSize)

{

int temp;

for (int i = 1; i < arrSize; i++)

{

temp = arrOfNumb[i];

int j = i - 1;

while (j >= 0 && arrOfNumb[j] > temp)

{

arrOfNumb[j + 1] = arrOfNumb[j];

arrOfNumb[j] = temp;

j--;

}

}

}

/// output from file

void outputFromFile(int\* arrOfNumb, int arrSize)

{

bool isIncorrect = true;

cout << "Write way to your file: ";

do

{

string fileWay = inputWayToTheFile();

if (isCanOpenFile(fileWay, ios::out))

{

ofstream file(fileWay, ios::out);

for (int i = 0; i < arrSize; i++)

file << arrOfNumb[i] << " ";

cout << "Check your file.";

file.close();

isIncorrect = false;

}

else

cerr << "Can't open a file. Try write another way: ";

} while (isIncorrect);

}

/// output from console

void outputFromConsole(int\* arrOfNumb, int arrSize)

{

for (int i = 0; i < arrSize; i++)

cout << arrOfNumb[i] << " ";

}

/// output

void resultOutput(int\* arrOfNumb, int arrSize)

{

cout << "You need to choose where to write information from.\n";

int path = choosingAPath();

path == CONSOLE\_KEY ? outputFromConsole(arrOfNumb, arrSize)

: outputFromFile(arrOfNumb, arrSize);

}

int main() {

int arrSize = 0;

int\* arrOfNumb = new int[0];

conditionOutput();

cout << "\nYou need to choose where to read information from.\n";

int path = choosingAPath();

if (path == CONSOLE\_KEY) {

arrSize = arrSizeInputFromConsole();

arrOfNumb = new int[arrSize];

arrOfNumbInputFromConsole(arrSize, arrOfNumb);

}

else {

bool isIncorrect = true;

fileRestriction();

cout << "Write way to your file: ";

do

{

string fileWay = inputFile();

ifstream file(fileWay, ios::in);

file >> arrSize;

isIncorrect = arrSizeInputFromFile(file, arrSize);

if (!isIncorrect)

{

arrOfNumb = new int[arrSize];

isIncorrect = isIncorrectArrOfNumbInputFromFile(file,

arrOfNumb, arrSize);

if (isIncorrect)

cout << "Invalid massive elements input. Try

again: ";

}

file.close();

} while (isIncorrect);

}

sortMassive(arrOfNumb, arrSize);

resultOutput(arrOfNumb, arrSize);

delete[] arrOfNumb;

arrOfNumb = nullptr;

return 0;

}

Код программы на **Java**:

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.util.Scanner;  
  
public class Lab3 {  
 static final int *FILE\_KEY* = 1;  
 static final int *CONSOLE\_KEY* = 2;  
 static final int *MIN\_ARR\_SIZE* = 1;  
 static final int *MIN\_FILE\_WAY\_SIZE* = 5;  
  
 static void conditionOutput()  
 {  
 System.*out*.println("""  
 The program is designed to sort an array  
 using the simple insertion method.  
 """);  
 }  
  
 static void pathConditionOutput()  
 {  
 System.*out*.printf("""  
 Where will we work through:\s  
 File: %d Console: %d  
   
 """, *FILE\_KEY*, *CONSOLE\_KEY*);  
 }  
  
 static void fileRestriction()  
 {  
 System.*out*.println("""  
   
 \*The first number is the number of elements  
 of the array, and subsequent numbers of this array\*  
 """);  
 }  
  
 static int choosingAPath(Scanner in)  
 {  
 int path = 0;  
 boolean isIncorrect = true;  
  
 *pathConditionOutput*();  
  
 System.*out*.print("Please write were we should work: ");  
 do  
 {  
 boolean isCorrect = false;  
  
 try  
 {  
 path = Integer.*parseInt*(in.nextLine());  
 isCorrect = true;  
 } catch(Exception error)  
 {  
 System.*err*.print("Error. You should write a one natural number.

Try again: ");  
 }  
  
 if (path == *CONSOLE\_KEY* || path == *FILE\_KEY*)  
 {  
 isIncorrect = false;  
 }  
 else if (isCorrect)  
 {  
 System.*err*.print("Error method. Try again: ");  
 }  
 } while (isIncorrect);  
  
 return path;  
 }  
  
 static int arrSizeInputFromConsole(Scanner in)  
 {  
 int arrSize = 0;  
 boolean isIncorrect = true;  
  
 System.*out*.print("Write your arr size: ");  
 do  
 {  
 boolean isCorrect = false;  
  
 try  
 {  
 arrSize = Integer.*parseInt*(in.nextLine());  
 isCorrect = true;  
 } catch(Exception error)  
 {  
 System.*err*.print("Invalid numeric input. Try again: ");  
 }  
  
 if (arrSize < *MIN\_ARR\_SIZE* && isCorrect)  
 {  
 System.*err*.printf("Minimal arr size is: %d. Try again: ",

*MIN\_ARR\_SIZE*);  
 }  
 else if (isCorrect)  
 {  
 isIncorrect = false;  
 }  
 } while (isIncorrect);  
  
 return arrSize;  
 }  
  
 static int inputCurrentNumbFromConsole(Scanner in)  
 {  
 int currentNumb = 0;  
 boolean isIncorrect = true;  
  
 do  
 {  
 try  
 {  
 currentNumb = Integer.*parseInt*(in.nextLine());  
 isIncorrect = false;  
 } catch(Exception error)  
 {  
 System.*err*.print("Invalid numeric input. Try again: ");  
 }  
  
 } while (isIncorrect);  
  
 return currentNumb;  
 }  
  
 static void arrOfNumbInputFromConsole(int arrSize, int[] arrOfNumb,

Scanner in)  
 {  
 for (int i = 0; i < arrSize; i++)  
 {  
 System.*out*.printf("Write your %d number: ", i + 1);  
 arrOfNumb[i] = *inputCurrentNumbFromConsole*(in);  
 }  
 }  
  
 static boolean isCanOpenFile(String way)  
 {  
 File file = new File(way);  
  
 return file.canRead();  
 }  
  
 static boolean wayCondition(String way)  
 {  
 if (way.length() < *MIN\_FILE\_WAY\_SIZE*)  
 {  
 System.*err*.print("The path is too short. Try again: ");  
 return false;  
 }  
  
 String bufstr = way.substring(way.length() - 4);

if (!bufstr.equals(".txt"))  
 {  
 System.*err*.print("Write .txt file. Try again: ");  
 return false;  
 }  
  
 return true;  
 }  
  
 static String inputWayToTheFile(Scanner in)  
 {  
 String way;  
 boolean isIncorrect;  
  
 do  
 {  
 way = in.nextLine();  
 isIncorrect = !*wayCondition*(way);  
 } while (isIncorrect);  
  
 return way;  
 }  
  
  
 static String inputFile(Scanner in)  
 {  
 String fileWay;  
 boolean isIncorrect = true;  
  
 do  
 {  
 fileWay = *inputWayToTheFile*(in);  
  
 if (!*isCanOpenFile*(fileWay))  
 {  
 System.*err*.print("Can't open a file. Try write another way: ");  
 }  
 else  
 {  
 isIncorrect = false;  
 }  
 } while (isIncorrect);  
  
 return fileWay;  
 }  
  
 static boolean arrSizeInputFromFile(int arrSize)  
 {  
 boolean isIncorrect = true;  
  
 if (arrSize < *MIN\_ARR\_SIZE*)  
 {  
 System.*out*.printf("Minimal arr size is %d. Try again: ",

*MIN\_ARR\_SIZE*);  
 }  
 else  
 {  
 isIncorrect = false;  
 }  
  
 return isIncorrect;  
 }  
  
 static boolean isIncorrectArrOfNumbInputFromFile(Scanner fileScanner,

int[] arrOfNumb, int arrSize)  
 {  
 boolean isIncorrect = false;  
  
 for (int i = 0; i < arrSize; i++)  
 {  
 try  
 {  
 arrOfNumb[i] = fileScanner.nextInt();  
 } catch(Exception error)  
 {  
 isIncorrect = true;  
 }  
 }  
  
 return isIncorrect;  
 }  
  
 static void sortMassive(int[] arrOfNumb, int arrSize)  
 {  
 int temp;  
  
 for (int i = 1; i < arrSize; i++)  
 {  
 temp = arrOfNumb[i];  
  
 int j = i - 1;  
 while (j >= 0 && arrOfNumb[j] > temp)  
 {  
 arrOfNumb[j + 1] = arrOfNumb[j];  
 arrOfNumb[j] = temp;  
 j--;  
 }  
 }  
 }  
  
 static void outputFromFile(int[] arrOfNumb, int arrSize, Scanner in)  
 {  
 boolean isIncorrect = true;  
  
 System.*out*.print("Write way to your file: ");  
 do  
 {  
 String fileWay = *inputWayToTheFile*(in);  
 File file = new File(fileWay);  
 StringBuilder builder;  
 if (file.canWrite())  
 {  
 try  
 {  
 FileWriter writer = new FileWriter(fileWay);  
 builder = new StringBuilder();  
 for (int i = 0 ; i < arrSize; i++)  
 builder.append(arrOfNumb[i]).append(" ");  
 writer.write(builder.toString());  
 writer.close();  
 System.*out*.print("Check your file.");  
 isIncorrect = false;  
 } catch (Exception error)  
 {  
 System.*err*.print("Can't write in this file. Try again: ");  
 }  
 }  
 else  
 {  
 System.*err*.print("Can't open a file. Try write another way: ");  
 }  
 } while (isIncorrect);  
 }  
  
 static void outputFromConsole(int[] arrOfNumb, int arrSize)  
 {  
 for (int i = 0; i < arrSize; i++)  
 {  
 System.*out*.printf("%d ", arrOfNumb[i]);  
 }  
 }  
  
 static void resultOutput(int[] arrOfNumb, int arrSize, Scanner in)  
 {  
 System.*out*.println("You need to choose where to write information from.");  
 int path = *choosingAPath*(in);  
 if (path == *CONSOLE\_KEY*)  
 {  
 *outputFromConsole*(arrOfNumb, arrSize);  
 }  
 else  
 {  
 *outputFromFile*(arrOfNumb, arrSize, in);  
 }  
 }  
  
 public static void main(String[] args)  
 {  
 Scanner in = new Scanner(System.*in*);  
  
 int arrSize = 0;  
 int[] arrOfNumb = new int[0];  
  
 *conditionOutput*();  
  
 System.*out*.println("\nYou need to choose where to read information

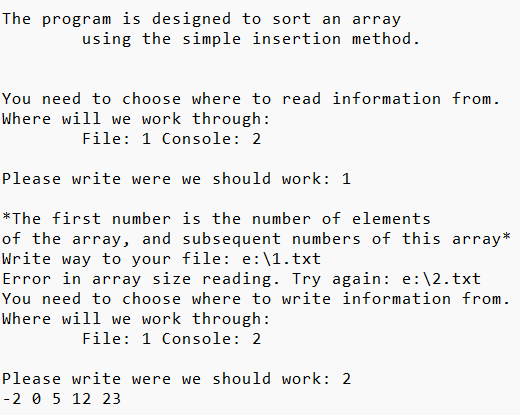
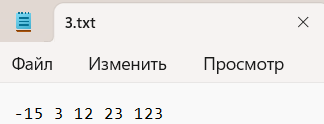
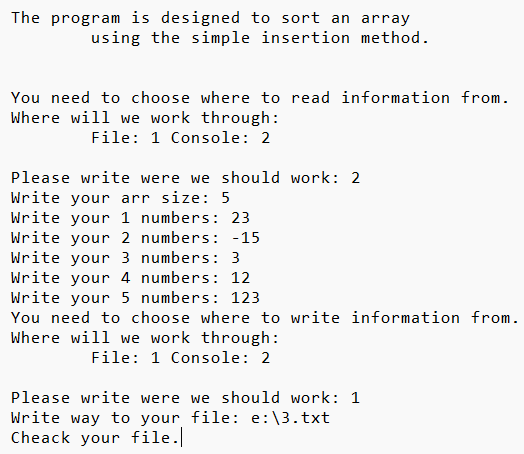
from.");  
  
 int path = *choosingAPath*(in);  
 if (path == *CONSOLE\_KEY*)  
 {  
 arrSize = *arrSizeInputFromConsole*(in);  
  
 arrOfNumb = new int[arrSize];  
 *arrOfNumbInputFromConsole*(arrSize, arrOfNumb, in);  
 }  
 else {  
 boolean isIncorrect = true;  
 *fileRestriction*();  
 System.*out*.print("Write way to your file: ");  
 do  
 {  
 String fileWay = *inputFile*(in);  
 try  
 {  
 Scanner fileScanner = new Scanner(new File(fileWay));  
 if (fileScanner.hasNextLine())  
 {  
 try {  
 arrSize = fileScanner.nextInt();  
 isIncorrect = false;  
 } catch(Exception error){  
 System.*err*.print("Error in input size of massive.

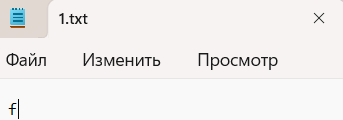
Try again: ");  
 }  
 if (!isIncorrect)  
 isIncorrect = *arrSizeInputFromFile*(arrSize);  
 if (!isIncorrect)  
 {  
 arrOfNumb = new int[arrSize];  
 isIncorrect = *isIncorrectArrOfNumbInputFromFile*

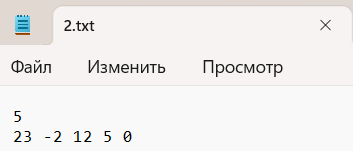
(fileScanner, arrOfNumb, arrSize);  
 if (isIncorrect)  
 System.*err*.print("Invalid massive elements input.

Try again: ");  
 }  
 }  
 else  
 {  
 System.*err*.print("Empty file. Try again: ");  
 }  
 fileScanner.close();  
 } catch (FileNotFoundException error)  
 {  
 System.*err*.print("File not found. Try again: ");  
 }  
 } while (isIncorrect);  
 }  
  
 *sortMassive*(arrOfNumb, arrSize);  
  
 *resultOutput*(arrOfNumb, arrSize, in);  
  
 in.close();  
 }  
}

Результат на **Delphi**:

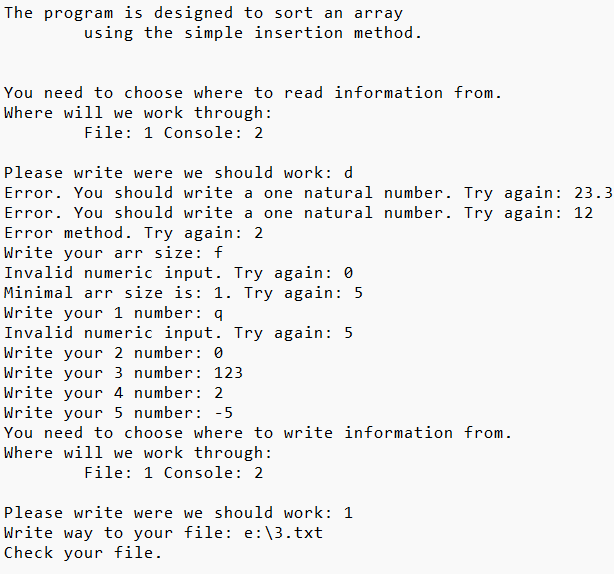


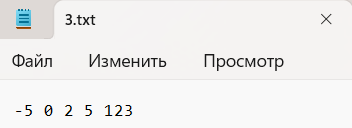


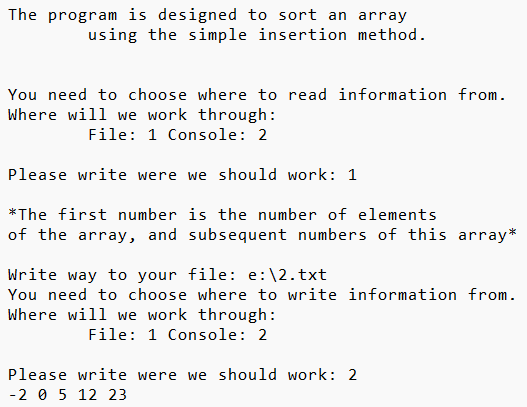
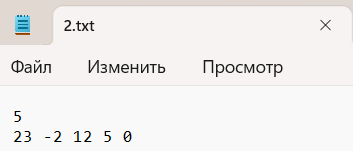
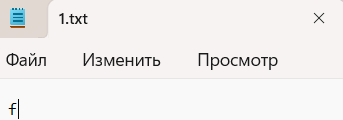


\

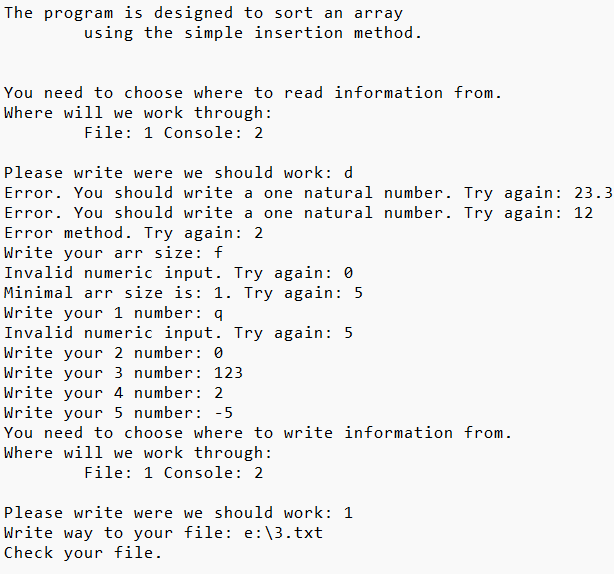
Результат на **Java**:

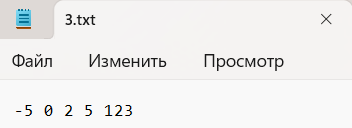


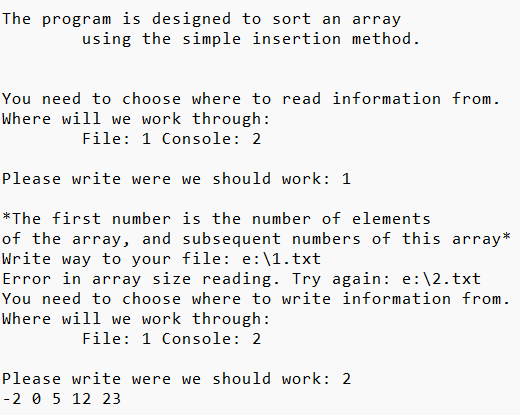


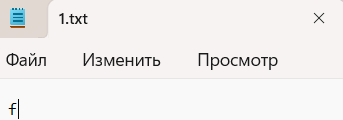
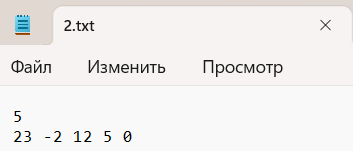


Результат на **C++**:









\

Блок-схема: