УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.3

по предмету «Основы алгоритмизации и программирования»

Вариант 12

Выполнил:

Галуха П. А.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Сортировка двухпутевыми вставками.

**Код программы Delphi:**

Program Lab33;

Uses

System.SysUtils;

Type

TArr = Array Of Integer;

TPosition = (PointerLeft, PointerRight);

ERRORS\_CODE = (CORRECT,

INCORRECT\_RANGE,

INCORRECT\_NUM,

INCORRECT\_CHOICE,

IS\_NOT\_TXT,

IS\_NOT\_EXIST,

IS\_NOT\_READABLE,

IS\_NOT\_WRITEABLE,

INCORRECT\_ARR\_LEN,

EXTRA\_DATA);

Const

MIN\_A = -1000000;

MAX\_A = 1000000;

MIN\_L = 1;

MAX\_L = 100;

ERRORS: Array [ERRORS\_CODE] Of String = ( '',

'Значение не попадает в диапазон!',

'Введено не число!',

'Некорректный выбор!',

'Расширение файла не .txt!',

'Проверьте корректность ввода пути к файлу!',

'Файл закрыт для чтения!',

'Файл закрыт для записи!',

'Неправильная длина массива!',

'Лишние данные в файле!');

Procedure PrintTask();

Begin

WriteLn('Данная программа реализует сортировку двухпутевыми вставками.');

End;

Procedure PrintError(Error: ERRORS\_CODE);

Begin

WriteLn(ERRORS[Error], #13#10'Повторите попытку: ');

End;

Function IsCorrectRange(Num: Integer; Const MIN, MAX: Integer) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

If (Num < MIN) Or (Num > MAX) Then

Error := INCORRECT\_RANGE;

IsCorrectRange := Error;

End;

Function ChooseOption(Quantity: Integer) : Integer;

Var

Error: ERRORS\_CODE;

SOption: String;

IOption: Integer;

Begin

IOption := 1;

Repeat

Error := CORRECT;

ReadLn(SOption);

Try

IOption := StrToInt(SOption)

Except

Error := INCORRECT\_CHOICE;

End;

If (Error = CORRECT) And ((IOption < 1) Or (IOption > Quantity)) Then

Error := INCORRECT\_CHOICE;

If Error <> CORRECT Then

PrintError(Error);

Until Error = CORRECT;

ChooseOption := IOption;

End;

Function GetPartStr(Str: String; PosStart, PosEnd: Integer) : String;

Var

PartStr: String;

I: Integer;

Begin

partStr := '';

For I := PosStart To PosEnd Do

PartStr := PartStr + Str[I];

GetPartStr := PartStr;

End;

Function IsFileTXT(PathToFile: String) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

If (Length(PathToFile) < 5) Or (GetPartStr(PathToFile, Length(PathToFile) - 3, Length(PathToFile)) <> '.txt') Then

Error := IS\_NOT\_TXT;

IsFileTXT := Error;

End;

Function IsExist(PathToFile: String) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

If Not FileExists(PathToFile) Then

Error := IS\_NOT\_EXIST;

IsExist := Error;

End;

Function IsReadable(Var F: TextFile) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Try

Reset(F);

Finally

CloseFile(F);

End;

Except

Error := IS\_NOT\_READABLE;

End;

IsReadable := Error;

End;

Function IsWriteable(Var F: TextFile) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Try

Append(F);

Finally

CloseFile(F);

End;

Except

Error := Is\_NOT\_WRITEABLE;

End;

IsWriteable := Error;

End;

Procedure GetFileNormalReading(Var F: TextFile);

Var

Error: ERRORS\_CODE;

PathToFile: String;

Begin

Repeat

ReadLn(PathToFile);

Error := IsFileTXT(PathToFile);

If Error = CORRECT Then

Error := IsExist(PathToFile);

If Error = CORRECT Then

AssignFile(F, PathToFile);

If Error = CORRECT Then

Error := IsReadable(F);

If Error <> CORRECT Then

PrintError(Error);

Until Error = CORRECT;

End;

Procedure GetFileNormalWriting(Var F: TextFile);

Var

Error: ERRORS\_CODE;

PathToFile: String;

Begin

Repeat

ReadLn(PathToFile);

Error := IsFileTXT(PathToFile);

If Error = CORRECT Then

Error := IsExist(PathToFile);

If Error = CORRECT Then

AssignFile(F, PathToFile);

If Error = CORRECT Then

Error := IsWriteable(F);

If Error <> CORRECT Then

PrintError(Error);

Until Error = CORRECT;

End;

Function ReadFileNum(Var F: TextFile; Var Num: Integer; Const MIN, MAX: Integer) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Read(F, Num);

Except

Error := INCORRECT\_NUM;

End;

If Error = CORRECT Then

Error := IsCorrectRange(Num, MIN, MAX);

ReadFileNum := Error;

End;

Function ReadFileArrLen(Var F: TextFile; Var ArrLen: Integer) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := ReadFileNum(F, ArrLen, MIN\_L, MAX\_L);

If (Error = CORRECT) And (Not EOLN(F)) Then

Error := EXTRA\_DATA;

ReadFileArrLen := Error;

End;

Function ReadFileArr(Var F: TextFile; Var Arr: TArr; ArrLen: Integer) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

I: Integer;

Begin

SetLength(Arr, ArrLen);

Error := CORRECT;

I := 0;

ReadLn(F);

While (Error = CORRECT) And (I < ArrLen) Do

Begin

Error := ReadFileNum(F, Arr[I], MIN\_A, MAX\_A);

Inc(I);

End;

If (Error = CORRECT) And (I <> ArrLen) Then

Error := INCORRECT\_ARR\_LEN;

If (Error = CORRECT) And (Not EOF(F)) Then

Error := EXTRA\_DATA;

ReadFileArr := Error;

End;

Function ReadFileData(Var F: TextFile; Var Arr: TArr) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

ArrLen: Integer;

Begin

Reset(F);

Error := ReadFileArrLen(F, ArrLen);

If Error = CORRECT Then

Error := ReadFileArr(F, Arr, ArrLen);

CloseFile(F);

ReadFileData := Error

End;

Procedure ReadFile(Var Arr: TArr);

Var

Error: ERRORS\_CODE;

F: TextFile;

Begin

WriteLn('Введите путь к файлу с расширением .txt.');

WriteLn('Содержимое: Массив с длиной[', MIN\_L, '; ', MAX\_L, '] и элементами в

диапазоне[', MIN\_A, '; ', MAX\_A, ']: ');

Repeat

GetFileNormalReading(F);

Error := ReadFileData(F, Arr);

If Error <> CORRECT Then

PrintError(Error);

Until Error = CORRECT;

End;

Procedure ReadConsoleNum(Var Num: Integer; Const MIN, MAX: Integer);

Var

Error: ERRORS\_CODE;

SNum: String;

Begin

Repeat

Error := CORRECT;

ReadLn(SNum);

Try

Num := StrToInt(SNum);

Except

Error := INCORRECT\_NUM;

End;

If Error = CORRECT Then

Error := IsCorrectRange(Num, MIN, MAX);

If Error <> CORRECT Then

PrintError(Error);

Until Error = CORRECT;

End;

Procedure ReadConsoleArrLen(Var ArrLen: Integer);

Begin

WriteLn('Введите длину массива в диапазоне[', MIN\_L, ': ', MAX\_L, ']: ');

ReadConsoleNum(ArrLen, MIN\_L, MAX\_L);

End;

Procedure ReadConsoleArr(Var Arr: TArr; ArrLen: Integer);

Var

I: Integer;

Begin

SetLength(Arr, ArrLen);

Write('Введите массив: '#13#10);

For I := Low(Arr) To High(Arr) Do

Begin

Write('Введите ', I + 1, ' элемент массива в диапазоне[', MIN\_A, ': ', MAX\_A, ']: ');

ReadConsoleNum(Arr[I], MIN\_A, MAX\_A);

End;

End;

Procedure ReadConsoleData(Var Arr: TArr);

Var

ArrLen: Integer;

Begin

ReadConsoleArrLen(ArrLen);

ReadConsoleArr(Arr, ArrLen);

End;

Procedure ReadConsole(Var Arr: TArr);

Begin

ReadConsoleData(Arr);

End;

Procedure ReadArr(Var Arr: TArr);

Var

Option: Integer;

Begin

WriteLn('Вы хотите: ');

WriteLn('Вводить массив через файл - 1');

WriteLn('Вводить массив через консоль - 2');

Option := ChooseOption(2);

If Option = 1 Then

ReadFile(Arr)

Else

ReadConsole(Arr);

End;

Procedure ShowProcess(HelpArr: TArr);

Var

I: Integer;

Begin

For I := Low(HelpArr) To High(HelpArr) Do

Write(HelpArr[I], ' ');

Writeln;

End;

Procedure SortArr(Arr: TArr);

Var

HelpArr: TArr;

LeftIndex, RightIndex, I, J, CurrentElem: Integer;

Begin

SetLength(HelpArr, Length(Arr) \* 2 - 1);

HelpArr[High(Arr)] := Arr[0];

LeftIndex := High(Arr);

RightIndex := High(Arr);

For I := 1 To High(Arr) Do

Begin

CurrentElem := Arr[I];

If CurrentElem > Arr[I - 1] Then

Begin

Inc(RightIndex);

J := RightIndex;

While CurrentElem < HelpArr[J - 1] Do

Begin

HelpArr[J] := HelpArr[J - 1];

Dec(J);

End;

End

Else

Begin

Dec(LeftIndex);

J := LeftIndex;

While CurrentElem > HelpArr[J + 1] Do

Begin

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

Inc(J);

End;

End;

HelpArr[J] := CurrentElem;

ShowProcess(HelpArr);

End;

For J := Low(Arr) To High(Arr) Do

Arr[J] := HelpArr[J + LeftIndex];

End;

Procedure PrintConsoleResult(Arr: TArr);

Var

I: Integer;

Begin

WriteLn(#13#10'Отсортированный массив: ');

For I := Low(Arr) To High(Arr) Do

Write(Arr[I], ' ');

End;

Procedure PrintFileResult(Arr: TArr);

Var

F: TextFile;

I: Integer;

Begin

WriteLn('Введите путь к файлу с расширением .txt для получения результата: ');

GetFileNormalWriting(F);

Append(F);

WriteLn(F, #13#10'Отсортированный массив: ');

For I := Low(Arr) To High(Arr) Do

Write(F, Arr[I], ' ');

CloseFile(F);

End;

Procedure PrintResult(Arr: TArr);

Var

Option: Integer;

Begin

WriteLn('Вы хотите: ');

WriteLn('Выводить массив через файл - 1');

WriteLn('Выводить массив через консоль - 2');

Option := ChooseOption(2);

If Option = 1 Then

PrintFileResult(Arr)

Else

PrintConsoleResult(Arr);

End;

Var

Arr: TArr;

Begin

PrintTask();

ReadArr(Arr);

SortArr(Arr);

PrintResult(Arr);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <set>

#include <string>

#include <fstream>

enum ErrorsCode

{

CORRECT,

INCORRECT\_RANGE,

INCORRECT\_NUM,

INCORRECT\_CHOICE,

IS\_NOT\_TXT,

IS\_NOT\_EXIST,

IS\_NOT\_READABLE,

IS\_NOT\_WRITEABLE,

INCORRECT\_ARR\_LEN,

EXTRA\_DATA,

};

const int

MIN\_A = -1000000,

MAX\_A = 1000000,

MIN\_L = 1,

MAX\_L = 100;

const std::string

ERRORS[] = { "",

"Значение не попадает в диапазон!",

"Введено не число!!",

"Некорректный выбор!",

"Расширение файла не .txt!",

"Проверьте корректность ввода пути к файлу!",

"Файл закрыт для чтения!",

"Файл закрыт для записи!",

"Неправильная длина массива!",

"Лишние данные в файле!"};

void printTask()

{

std::cout << "Данная программа реализует сортировку двухпутевыми вставками.\n";

}

void printError(ErrorsCode error)

{

std::cout << ERRORS[error] << "\nПовторите попытку: \n";

}

ErrorsCode isCorrectRange(int num, const int MIN, const int MAX)

{

ErrorsCode error;

error = CORRECT;

if (num < MIN || num > MAX)

error = INCORRECT\_RANGE;

return error;

}

int chooseOption(int quantity)

{

ErrorsCode error;

int option;

option = 1;

do {

error = CORRECT;

std::cin >> option;

if (std::cin.fail())

{

error = INCORRECT\_CHOICE;

std::cin.clear();

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

}

if (error == CORRECT && std::cin.get() != '\n')

{

error = INCORRECT\_CHOICE;

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

}

if (error == CORRECT && (option < 1 || option > quantity))

error = INCORRECT\_CHOICE;

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

return option;

}

std::string getPartStr(std::string str, int posStart, int posEnd)

{

std::string partStr;

int i;

partStr = "";

for (i = posStart; i <= posEnd; i++)

partStr = partStr + str[i];

return partStr;

}

ErrorsCode isFileTXT(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

if (pathToFile.length() < 5 || getPartStr(pathToFile, (int)pathToFile.length() - 4,

(int)pathToFile.length() - 1) != ".txt")

error = IS\_NOT\_TXT;

return error;

}

ErrorsCode isExist(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

std::ifstream file(pathToFile);

if (!file.good())

error = IS\_NOT\_EXIST;

file.close();

return error;

}

ErrorsCode isReadable(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

std::ifstream file(pathToFile);

if (!file.is\_open())

error = IS\_NOT\_READABLE;

file.close();

return error;

}

ErrorsCode isWriteable(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

std::ofstream file(pathToFile, std::ios::app);

if (!file.is\_open())

error = IS\_NOT\_WRITEABLE;

file.close();

return error;

}

void getFileNormalReading(std::string& pathToFile)

{

ErrorsCode error;

do

{

std::getline(std::cin, pathToFile);

error = isFileTXT(pathToFile);

if (error == CORRECT)

error = isExist(pathToFile);

if (error == CORRECT)

error = isReadable(pathToFile);

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

}

void getFileNormalWriting(std::string& pathToFile)

{

ErrorsCode error;

do

{

std::getline(std::cin, pathToFile);

error = isFileTXT(pathToFile);

if (error == CORRECT)

error = isExist(pathToFile);

if (error == CORRECT)

error = isWriteable(pathToFile);

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

}

ErrorsCode readFileNum(std::ifstream& file, int& num, const int MIN, const int MAX)

{

ErrorsCode error;

error = CORRECT;

file >> num;

if (std::cin.fail())

{

error = INCORRECT\_NUM;

std::cin.clear();

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

}

if (error == CORRECT && std::cin.get() != '\n')

{

error = INCORRECT\_NUM;

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

}

if (error == CORRECT)

error = isCorrectRange(num, MIN, MAX);

return error;

}

ErrorsCode readFileArrLen(std::ifstream& file, int& arrLen)

{

ErrorsCode error;

error = readFileNum(file, arrLen, MIN\_L, MAX\_L);

if (error == CORRECT && file.peek() != '\n')

error = EXTRA\_DATA;

return error;

}

ErrorsCode readFileArr(std::ifstream& file, int\*& arr, int arrLen)

{

ErrorsCode error;

int i;

error = CORRECT;

i = 0;

arr = new int[arrLen];

file.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

while (error == CORRECT && i < arrLen)

{

error = readFileNum(file, arr[i], MIN\_A, MAX\_A);

i++;

}

if (error == CORRECT && i != arrLen)

error = INCORRECT\_ARR\_LEN;

if (error == CORRECT && !file.eof())

error = EXTRA\_DATA;

return error;

}

ErrorsCode readFileData(std::string pathToFile, int\*& arr, int& arrLen)

{

ErrorsCode error;

std::ifstream file(pathToFile);

error = readFileArrLen(file, arrLen);

if (file.good()) {

error = readFileArrLen(file, arrLen);

if (error == CORRECT)

error = readFileArr(file, arr, arrLen);

}

else

error = IS\_NOT\_EXIST;

file.close();

return error;

}

ErrorsCode readFile(int\*& arr, int& arrLen)

{

ErrorsCode error;

std::string pathToFile;

pathToFile = "";

std::cout << "Введите путь к файлу с расширением .txt.\n";

std::cout << "Содержание: Массив с длиной[" << MIN\_L << "; " << MAX\_L << "] и

элементами в диапазоне[" << MIN\_A << "; " << MAX\_A << "]: ";

do

{

getFileNormalReading(pathToFile);

error = readFileData(pathToFile, arr, arrLen);

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

return error;

}

void readConsoleNum(int& num, const int MIN, const int MAX)

{

ErrorsCode error;

do

{

error = CORRECT;

std::cin >> num;

if (std::cin.fail())

{

error = INCORRECT\_NUM;

std::cin.clear();

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

}

if (error == CORRECT && std::cin.get() != '\n')

{

error = INCORRECT\_NUM;

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

}

if (error == CORRECT)

error = isCorrectRange(num, MIN, MAX);

if (error != CORRECT)

printError(error);

} while (error != CORRECT);

}

void readConsoleArrLen(int& arrLen)

{

std::cout << "Введите длину массива в диапазоне[" << MIN\_L << ": " << MAX\_L << "]: ";

readConsoleNum(arrLen, MIN\_L, MAX\_L);

}

void readConsoleArr(int\*& arr, int arrLen)

{

int i;

arr = new int[arrLen];

std::cout << "Введите массив: \n";

for (i = 0; i < arrLen; i++)

{

std::cout << "Введите " << (i + 1) << " элемент массива в

диапазоне[" << MIN\_A << ": " << MAX\_A << "]: ";

readConsoleNum(arr[i], MIN\_A, MAX\_A);

}

}

void readConsoleData(int\*& arr, int& arrLen)

{

readConsoleArrLen(arrLen);

readConsoleArr(arr, arrLen);

}

void readConsole(int\*& arr, int& arrLen)

{

readConsoleData(arr, arrLen);

}

void readArr(int\*& arr, int& arrLen)

{

int option;

std::cout << "Вы хотите: \n";

std::cout << "Вводить массив через файл - 1\n";

std::cout << "Вводить массив через консоль - 2\n";

option = chooseOption(2);

if (option == 1)

readFile(arr, arrLen);

else

readConsole(arr, arrLen);

}

void showProcess(int\* helpArr, int helpArrLen)

{

int i;

for (i = 0; i < helpArrLen; i++)

std::cout << helpArr[i] << " ";

std::cout << "\n";

}

void sortArr(int\*& arr, int arrLen)

{

int\* helpArr;

int helpArrLen;

int leftIndex, rightIndex, i, j, currentElem;

helpArrLen = arrLen \* 2 - 1;

helpArr = new int[helpArrLen]();

helpArr[arrLen - 1] = arr[0];

leftIndex = rightIndex = arrLen - 1;

for (i = 1; i < arrLen; i++)

{

currentElem = arr[i];

if (currentElem > arr[i - 1])

{

rightIndex++;

j = rightIndex;

while (currentElem < helpArr[j - 1])

{

helpArr[j] = helpArr[j - 1];

j--;

}

}

else

{

leftIndex--;

j = leftIndex;

while (currentElem > helpArr[j + 1])

{

helpArr[j] = helpArr[j + 1];

j++;

}

}

helpArr[j] = currentElem;

showProcess(helpArr, helpArrLen);

}

for (j = 0; j < arrLen; j++)

arr[j] = helpArr[j + leftIndex];

delete[] helpArr;

}

void printConsoleResult(int\* arr, int arrLen)

{

int i;

std::cout << "\nОтсортированный массив: \n";

for (i = 0; i < arrLen; i++)

std::cout << arr[i] << " ";

}

void printFileResult(int\* arr, int arrLen)

{

std::string pathToFile;

int i;

pathToFile = "";

std::cout << "Введите путь к файлу с расширением .txt для получения результата: \n";

getFileNormalWriting(pathToFile);

std::ofstream file(pathToFile, std::ios::app);

file << "\nОтсортированный массив: \n";

for (i = 0; i < arrLen; i++)

file << arr[i] << " ";

file.close();

}

void printResult(int\* arr, int arrLen)

{

int option;

std::cout << "Вы хотите: \n";

std::cout << "Выводить массив через файл - 1\n";

std::cout << "Выводить массив через консоль - 2\n";

option = chooseOption(2);

if (option == 1)

printFileResult(arr, arrLen);

else

printConsoleResult(arr, arrLen);

}

int main()

{

setlocale(LC\_ALL, "RU");

int\* arr;

int arrLen;

printTask();

readArr(arr, arrLen);

sortArr(arr, arrLen);

printResult(arr, arrLen);

delete[] arr;

return 0;

}

**Код программы Java:**

import java.util.Scanner;

import java.io.File;

import java.io.FileWriter;

public class Main {

public enum ErrorsCode {

CORRECT,

INCORRECT\_RANGE,

INCORRECT\_NUM,

INCORRECT\_CHOICE,

IS\_NOT\_TXT,

IS\_NOT\_EXIST,

IS\_NOT\_READABLE,

IS\_NOT\_WRITEABLE,

INCORRECT\_ARR\_LEN,

EXTRA\_DATA,

}

public static final int

MIN\_A = -1000000,

MAX\_A = 1000000,

MIN\_L = 1,

MAX\_L = 100;

public static final String[]

ERRORS = { "",

"Значение не попадает в диапазон!",

"Введено не число!!",

"Некорректный выбор!",

"Расширение файла не .txt!",

"Проверьте корректность ввода пути к файлу!",

"Файл закрыт для чтения!",

"Файл закрыт для записи!",

"Неправильная длина массива!",

"Лишние данные в файле!" };

static Scanner scanConsole = new Scanner(System.in);

public static void printTask() {

System.out.print("Данная программа реализует сортировку двухпутевыми вставками.\n");

}

public static void printError(ErrorsCode error) {

System.err.print(ERRORS[error.ordinal()] + "\nПовторите попытку: \n");

}

public static ErrorsCode isCorrectRange(int num, final int MIN, final int MAX) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

if (num < MIN || num > MAX)

error = ErrorsCode.INCORRECT\_RANGE;

return error;

}

public static int chooseOption(int quantity) {

ErrorsCode error;

String sOption;

int iOption;

iOption = 1;

do {

error = ErrorsCode.CORRECT;

sOption = scanConsole.nextLine();

try {

iOption = Integer.parseInt(sOption);

} catch (NumberFormatException e) {

error = ErrorsCode.INCORRECT\_CHOICE;

}

if (error == ErrorsCode.CORRECT && (iOption < 1 || iOption > quantity))

error = ErrorsCode.INCORRECT\_CHOICE;

if (error != ErrorsCode.CORRECT)

printError(error);

} while (error != ErrorsCode.CORRECT);

return iOption;

}

public static String getPartStr(String str, int posStart, int posEnd) {

String partStr;

int i;

partStr = "";

for (i = posStart; i <= posEnd; i++)

partStr = partStr + str.charAt(i);

return partStr;

}

public static ErrorsCode isFileTXT(String pathToFile) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

if (pathToFile.length() < 5 || !getPartStr(pathToFile, pathToFile.length() - 4,

pathToFile.length() - 1).equals(".txt"))

error = ErrorsCode.IS\_NOT\_TXT;

return error;

}

public static ErrorsCode isExist(File file) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

if (!file.exists())

error = ErrorsCode.IS\_NOT\_EXIST;

return error;

}

public static ErrorsCode isReadable(File file) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

try(Scanner scanFile = new Scanner(file)) {

} catch (Exception e) {

error = ErrorsCode.IS\_NOT\_READABLE;

}

return error;

}

public static ErrorsCode isWriteable(File file) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

if (!file.canWrite())

error = ErrorsCode.IS\_NOT\_WRITEABLE;

return error;

}

public static File getFileNormalReading() {

File file;

ErrorsCode error;

String pathToFile;

do {

pathToFile = scanConsole.nextLine();

error = isFileTXT(pathToFile);

file = new File(pathToFile);

if (error == ErrorsCode.CORRECT)

error = isExist(file);

if (error == ErrorsCode.CORRECT)

error = isReadable(file);

if (error != ErrorsCode.CORRECT)

printError(error);

} while (error != ErrorsCode.CORRECT);

return file;

}

public static File getFileNormalWriting() {

File file;

ErrorsCode error;

String pathToFile;

do {

pathToFile = scanConsole.nextLine();

error = isFileTXT(pathToFile);

file = new File(pathToFile);

if (error == ErrorsCode.CORRECT)

error = isExist(file);

if (error == ErrorsCode.CORRECT)

error = isWriteable(file);

if (error != ErrorsCode.CORRECT)

printError(error);

} while (error != ErrorsCode.CORRECT);

return file;

}

public static ErrorsCode readFileNum(Scanner scanFile, int[] num, final int MIN,

final int MAX) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

try {

num[0] = scanFile.nextInt();

} catch (NumberFormatException e) {

error = ErrorsCode.INCORRECT\_NUM;

}

if (error == ErrorsCode.CORRECT)

error = isCorrectRange(num[0], MIN, MAX);

return error;

}

public static ErrorsCode readFileArrLen(Scanner scanFile, int[] arrLen) {

ErrorsCode error;

error = readFileNum(scanFile, arrLen, MIN\_L, MAX\_L);

return error;

}

public static ErrorsCode readFileArr(Scanner scanFile, int[] arr, int arrLen) {

ErrorsCode error;

int[] num = new int[1];

int i;

error = ErrorsCode.CORRECT;

i = 0;

while (error == ErrorsCode.CORRECT && i < arrLen) {

error = readFileNum(scanFile, num, MIN\_A, MAX\_A);

arr[i] = num[0];

i++;

}

if (error == ErrorsCode.CORRECT && i != arrLen)

error = ErrorsCode.INCORRECT\_ARR\_LEN;

if (error == ErrorsCode.CORRECT && !scanFile.hasNext())

error = ErrorsCode.EXTRA\_DATA;

return error;

}

public static ErrorsCode readFileData(File file, int[] arr) {

ErrorsCode error;

int i;

int[] n = new int[1];

int arrLen;

error = ErrorsCode.CORRECT;

try(Scanner scanFile = new Scanner(file)) {

error = readFileArrLen(scanFile, n);

arrLen = n[0];

if (error == ErrorsCode.CORRECT)

error = readFileArr(scanFile, arr, arrLen);

} catch (Exception e) {

error = ErrorsCode.IS\_NOT\_READABLE;

}

return error;

}

public static int[] readFile() {

ErrorsCode error;

int[] n = new int[1];

int[] arr = n;

File file;

System.out.print("Введите путь к файлу с расширением .txt.\n");

System.out.print("Содержание: Массив с длиной[" + MIN\_L + "; " +

MAX\_L + "] и элементами в диапазоне[" + MIN\_A + "; " + MAX\_A + "]: ");

do {

file = getFileNormalReading();

error = readFileData(file, arr);

if (error != ErrorsCode.CORRECT)

printError(error);

} while (error != ErrorsCode.CORRECT);

return arr;

}

public static int readConsoleNum(final int MIN, final int MAX) {

ErrorsCode error;

String sNum;

int num;

num = 0;

do {

error = ErrorsCode.CORRECT;

sNum = scanConsole.nextLine();

try {

num = Integer.parseInt(sNum);

} catch (NumberFormatException e) {

error = ErrorsCode.INCORRECT\_NUM;

}

if (error == ErrorsCode.CORRECT)

error = isCorrectRange(num, MIN, MAX);

if (error != ErrorsCode.CORRECT)

printError(error);

} while (error != ErrorsCode.CORRECT);

return num;

}

public static int readConsoleArrLen() {

int arrLen;

System.out.print("Введите длину массива в диапазоне[" + MIN\_L + ": " + MAX\_L + "]: ");

arrLen = readConsoleNum(MIN\_L, MAX\_L);

return arrLen;

}

public static int[] readConsoleArr(int arrLen) {

int[] arr;

int i;

arr = new int[arrLen];

System.out.print("Введите массив: \n");

for (i = 0; i < arrLen; i++) {

System.out.print("Введите " + (i + 1) + " элемент массива в

диапазоне[" + MIN\_A + ": " + MAX\_A + "]: ");

arr[i] = readConsoleNum(MIN\_A, MAX\_A);

}

return arr;

}

public static int[] readConsoleData() {

int[] arr;

int arrLen;

arrLen = readConsoleArrLen();

arr = readConsoleArr(arrLen);

return arr;

}

public static int[] readConsole() {

int[] arr;

arr = readConsoleData();

return arr;

}

public static int[] readArr() {

int[] arr;

int option;

System.out.print("Вы хотите: \n");

System.out.print("Вводить массив через файл - 1\n");

System.out.print("Вводить массив через консоль - 2\n");

option = chooseOption(2);

if (option == 1)

arr = readFile();

else

arr = readConsole();

return arr;

}

public static void showProcess(int[] helpArr) {

int i;

for (i = 0; i < helpArr.length; i++)

System.out.print(helpArr[i] + " ");

System.out.print("\n");

}

public static void sortArr(int[] arr) {

int[] helpArr;

int leftIndex, rightIndex, i, j, currentElem;

helpArr = new int[arr.length \* 2 - 1];

helpArr[arr.length - 1] = arr[0];

leftIndex = rightIndex = arr.length - 1;

for (i = 1; i < arr.length; i++) {

currentElem = arr[i];

if (currentElem > arr[i - 1]) {

rightIndex++;

j = rightIndex;

while (currentElem < helpArr[j - 1]) {

helpArr[j] = helpArr[j - 1];

j--;

}

}

else {

leftIndex--;

j = leftIndex;

while (currentElem > helpArr[j + 1]) {

helpArr[j] = helpArr[j + 1];

j++;

}

}

helpArr[j] = currentElem;

showProcess(helpArr);

}

for (j = 0; j < arr.length; j++)

arr[j] = helpArr[j + leftIndex];

}

public static void printConsoleResult(int[] arr) {

int i;

System.out.print("\nОтсортированный массив: \n");

for (i = 0; i < arr.length; i++)

System.out.print(arr[i] + " ");

}

public static void printFileResult(int[] arr) {

ErrorsCode error;

File file;

int i;

System.out.print("Введите путь к файлу с расширением .txt для получения

результата: \n");

file = getFileNormalWriting();

try(FileWriter writer = new FileWriter(file, true)) {

writer.write("\nОтсортированный массив: \n");

for (i = 0; i < arr.length; i++)

writer.write(arr[i] + " ");

} catch (Exception e) {

error = ErrorsCode.IS\_NOT\_WRITEABLE;

printError(error);

}

}

public static void printResult(int[] arr) {

int option;

System.out.print("Вы хотите: \n");

System.out.print("Выводить массив через файл - 1\n");

System.out.print("Выводить массив через консоль - 2\n");

option = chooseOption(2);

if (option == 1)

printFileResult(arr);

else

printConsoleResult(arr);

}

public static void main(String[] args) {

int[] arr;

printTask();

arr = readArr();

sortArr(arr);

printResult(arr);

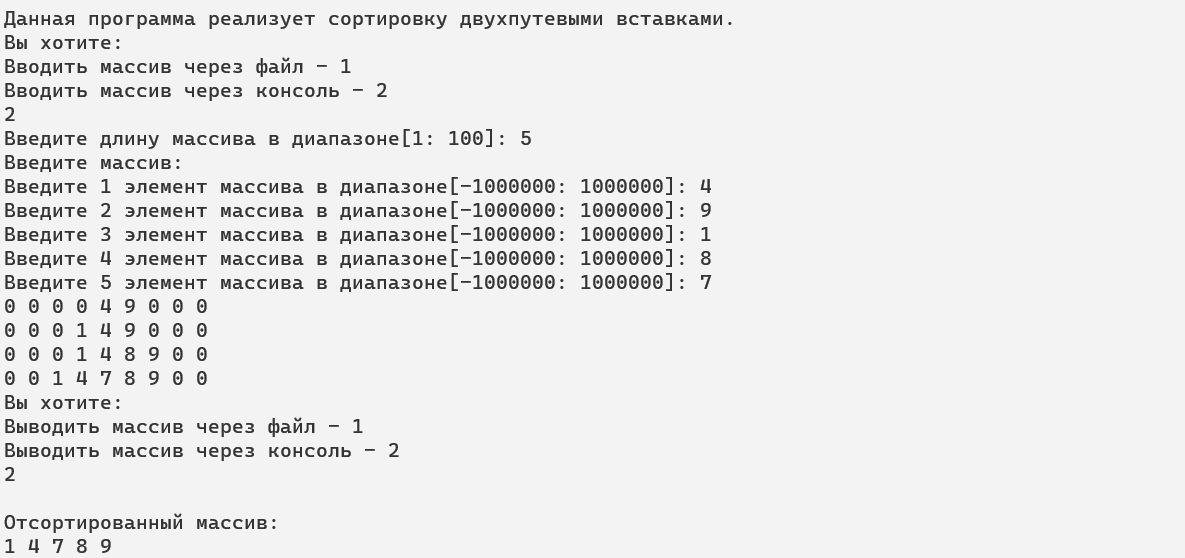
scanConsole.close();

}

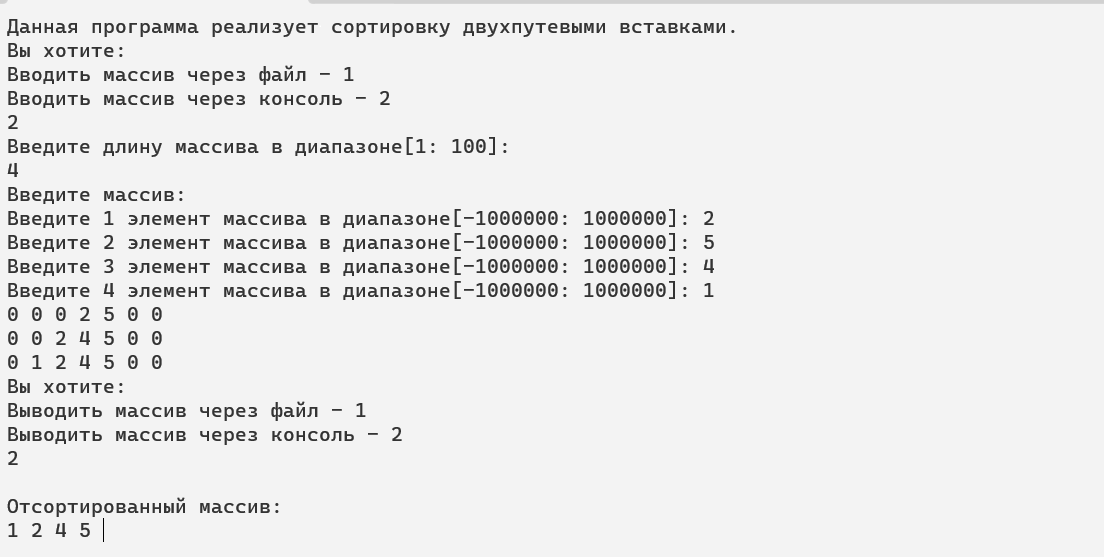
}

**Скриншоты:**

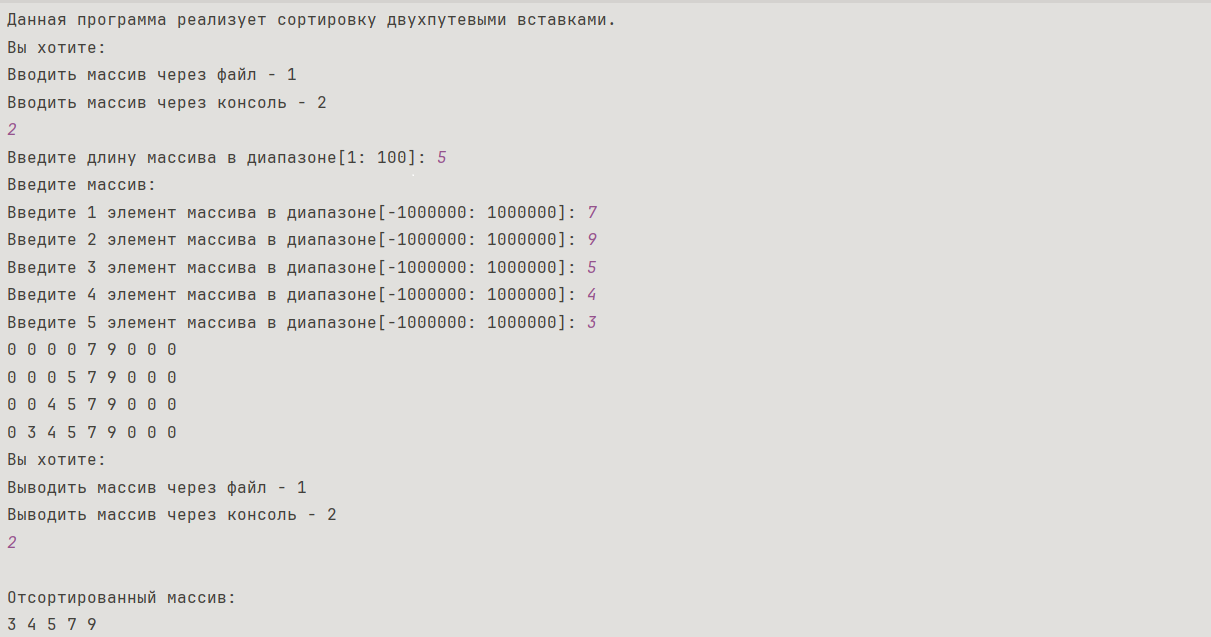
**Delphi:**



**C++:**



**Java:**



**Блок-схем**

