УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.3

по предмету «Основы алгоритмизации и программирования»

Вариант 20

Выполнил:

Захвей И.В.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Дана двумерная матрица А порядка n. Расставить элементы строк с четными номерами матрицы в порядке убывания.

**Код программы Delphi:**

Program Project2;

Uses

System.SysUtils;

Type

OneSizeArr = Array Of Integer;

TwoSizeArr = Array Of OneSizeArr;

Const

MAX\_SIZE = 100;

MIN\_SIZE = 2;

MIN\_ELEM = -MaxInt - 1;

MAX\_ELEM = MaxInt;

Procedure PrintInf();

Begin

Writeln('Program sort even rows of square matrix from larger to smaller');

End;

Function SortArr(Arr: OneSizeArr): OneSizeArr;

Var

I, J, Bufer: Integer;

IsNotSort: Boolean;

Begin

IsNotSort := True;

While IsNotSort Do

Begin

IsNotSort := False;

For I := 1 To High(Arr) Do

For J := I To High(Arr) Do

If Arr[J - 1] < Arr[J] Then

Begin

IsNotSort := True;

Bufer := Arr[J];

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

Arr[J - 1] := Bufer;

End;

End;

SortArr := Arr;

End;

Function InpValidNum(Const MIN, MAX: Integer): Integer;

Var

IsCorrect: Boolean;

Num: Integer;

Begin

Repeat

Try

Readln(Num);

IsCorrect := True;

Except

Writeln('Data is not correct, or number is too large',

' (it should be from ', MIN, ' to ', MAX, ' )');

IsCorrect := False;

Writeln('Please, enter again');

End;

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

Begin

Writeln('It should be from ', MIN, ' to ', MAX);

IsCorrect := False;

Writeln('Please, enter again');

End;

Until IsCorrect;

InpValidNum := Num;

End;

Function EnterArr(Row, Column: Integer): TwoSizeArr;

Var

Arr: TwoSizeArr;

I, J: Integer;

Begin

SetLength(Arr, Row, Column);

For I := 0 To High(Arr) Do

For J := 0 To High(Arr[I]) Do

Begin

Writeln('Enter a', I + 1, J + 1, ' element');

Arr[I][J] := InpValidNum(MIN\_ELEM, MAX\_ELEM);

End;

EnterArr := Arr;

End;

Procedure PrintArr(Arr: TwoSizeArr; Row: Integer; Column: Integer);

Var

I, J, RealRow, RealCol: Integer;

Begin

RealRow := Row - 1;

RealCol := Column - 1;

For I := 0 To RealRow Do

Begin

For J := 0 To RealCol Do

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

Writeln;

End;

End;

Function MakeCopy(DefaultArr: TwoSizeArr): TwoSizeArr;

Var

CopyArr: TwoSizeArr;

I, J: Integer;

Begin

SetLength(CopyArr, Length(DefaultArr), Length(DefaultArr));

For I := 0 To High(CopyArr) Do

For J := 0 To High(CopyArr[I]) Do

CopyArr[I][J] := DefaultArr[I][J];

MakeCopy := CopyArr;

End;

Function SortEvenRow(DefaultArr: TwoSizeArr; Row: Integer): TwoSizeArr;

Var

I: Integer;

Arr: TwoSizeArr;

Begin

Arr := MakeCopy(DefaultArr);

If High(Arr) > 0 Then

Begin

I := 1;

While (I < Row) Do

Begin

Arr[I] := SortArr(Arr[I]);

Inc(I, 2);

End;

End;

SortEvenRow := Arr;

End;

Function ReadSizeFromFile(Var InfFile: TextFile): Integer;

Var

IsCorrect: Boolean;

Size: Integer;

Begin

Size := 0;

Try

Read(InfFile, Size);

Except

Writeln('Size is not correct');

Size := 0;

IsCorrect := False;

End;

If IsCorrect And ((Size < MIN\_SIZE) Or (Size > MAX\_SIZE)) Then

Begin

Writeln('Size of array should be from ', MIN\_SIZE, ' to ', MAX\_SIZE);

IsCorrect := False;

Size := 0;

End;

ReadSizeFromFile := Size;

End;

Function IsFileCorrect(Var InfFile: TextFile): Boolean;

Var

IsCanReset: Boolean;

Begin

IsCanReset := True;

Try

Try

Reset(InfFile);

Finally

Close(InfFile);

End;

Except

IsCanReset := False;

Writeln('File is can not be opened');

End;

IsFileCorrect := IsCanReset;

End;

Function IsFileTxt(Name: String): Boolean;

Var

LastFourChar: String;

Begin

LastFourChar := Copy(Name, Length(Name) - 3, Length(Name));

If LastFourChar = '.txt' Then

IsFileTxt := True

Else

Begin

Writeln('File is not .txt');

IsFileTxt := False;

End;

End;

Function IsFileOk(FileName: String): Boolean;

Var

MyFile: TextFile;

Begin

AssignFile(MyFile, FileName);

If Not FileExists(FileName) Then

Writeln('File is not exist');

IsFileOk := IsFileTxt(FileName) And FileExists(FileName) And IsFileCorrect(MyFile);

End;

Procedure ChekFileAfterReading(Var IsCorrect: Boolean; Var MyFile: TextFile;

IsElemIncorrect: Boolean);

Begin

If IsElemIncorrect Then

Begin

Writeln('One of the element is incorrect');

IsCorrect := False;

End

Else If Eof(MyFile) And IsCorrect Then

Begin

Writeln('Reading is successfull')

End

Else If IsCorrect Then

Begin

Writeln('Count of element is too a lot');

IsCorrect := False;

End

Else

Begin

Writeln('Count of element is not enough');

IsCorrect := False;

End;

End;

Function ReadValidFileInf(Name: String; Var Size: Integer): TwoSizeArr;

Var

InfFile: TextFile;

IsCorrect, IsElemIncorrect: Boolean;

I, J, Buffer: Integer;

Arr: TwoSizeArr;

Begin

IsCorrect := True;

AssignFile(InfFile, Name);

If IsFileOk(Name) Then

Begin

Reset(InfFile);

Size := ReadSizeFromFile(InfFile);

IsCorrect := Size > 1;

IsElemIncorrect := False;

SetLength(Arr, Size, Size);

For I := 0 To High(Arr) Do

For J := 0 To High(Arr[I]) Do

Begin

If Eof(InfFile) Then

IsCorrect := False

Else

Begin

Try

Read(InfFile, Arr[I][J]);

Except

IsElemIncorrect := True;

End;

End;

End;

ChekFileAfterReading(IsCorrect, InfFile, IsElemIncorrect);

CloseFile(InfFile);

End

Else

IsCorrect := False;

If Not IsCorrect Then

Arr := [[]];

ReadValidFileInf := Arr;

End;

Procedure WriteInfFile(Name: String; Var DefaultArr, SortedArr: TwoSizeArr;

Size: Integer);

Var

OutFile: TextFile;

I, J: Integer;

Begin

AssignFile(OutFile, Name);

Rewrite(OutFile);

Writeln(OutFile, 'Default array');

For I := 0 To High(DefaultArr) Do

Begin

For J := 0 To High(DefaultArr) Do

Write(OutFile, DefaultArr[I][J], ' ');

Write(OutFile, #13#10);

End;

Writeln(OutFile, 'Sorted array');

For I := 0 To High(SortedArr) Do

Begin

For J := 0 To High(SortedArr) Do

Write(OutFile, SortedArr[I][J], ' ');

Write(OutFile, #13#10);

End;

Close(OutFile);

Writeln('Writing is successfull');

End;

Function UserChoice(): Integer;

Var

Choice: Integer;

Begin

Writeln('Choose a way of input/output of data', #13#10, '1 -- Console',

#13#10, '2 -- File');

Choice := InpValidNum(1, 2);

UserChoice := Choice;

End;

Procedure InputFromConsole(Var Size: Integer; Var Arr: TwoSizeArr);

Begin

Writeln('Enter size of array, please');

Size := InpValidNum(MIN\_SIZE, MAX\_SIZE);

Writeln('Now enter the elements');

Arr := EnterArr(Size, Size);

End;

Procedure InputFromFile(Var Size: Integer; Var Arr: TwoSizeArr);

Var

IsCorrect: Boolean;

Name: String;

Begin

Repeat

IsCorrect := True;

Writeln('Please enter the full path to file');

Readln(Name);

If IsFileOk(Name) Then

Arr := ReadValidFileInf(Name, Size)

Else

IsCorrect := False;

Until IsCorrect;

End;

Function InputInf(Var Size: Integer): TwoSizeArr;

Var

Arr: TwoSizeArr;

ChoiceInp: Integer;

Begin

ChoiceInp := UserChoice();

If ChoiceInp = 1 Then

InputFromConsole(Size, Arr)

Else

InputFromFile(Size, Arr);

InputInf := Arr;

End;

Procedure OutputInConsole(DefaultArr, SortedArr: TwoSizeArr; Size: Integer);

Begin

Writeln('Default Array');

PrintArr(DefaultArr, Size, Size);

Writeln('Sorted Array');

PrintArr(SortedArr, Size, Size);

End;

Procedure OutputInFile(DefaultArr, SortedArr: TwoSizeArr; Size: Integer);

Var

IsCorrect: Boolean;

Name: String;

Begin

Repeat

IsCorrect := True;

Writeln('Please enter the full path to file');

Readln(Name);

If IsFileOk(Name) Then

WriteInfFile(Name, DefaultArr, SortedArr, Size)

Else

IsCorrect := False;

Until IsCorrect;

End;

Procedure OutputInf(DefaultArr, SortedArr: TwoSizeArr; Size: Integer);

Var

ChoiceOut: Integer;

Begin

If (Length(DefaultArr) > 1) Then

Begin

ChoiceOut := UserChoice();

If (ChoiceOut = 1) Then

OutputInConsole(DefaultArr, SortedArr, Size)

Else

OutputInFile(DefaultArr, SortedArr, Size);

End;

End;

Var

Size, ChoiceInp, ChoiceOut, I, J: Integer;

ArrOfNum, SortedArr, Arr: TwoSizeArr;

FileName: String;

Begin

PrintInf;

ArrOfNum := InputInf(Size);

SortedArr := SortEvenRow(ArrOfNum, Size);

OutputInf(ArrOfNum, SortedArr, Size);

Readln;

End.

**Код программы С++:**

#include <iostream>

#include <fstream>

#include <string>

using std::string;

using std::cin;

using std::cout;

using std::ifstream;

using std::ofstream;

const int MAX\_SIZE = 100;

const int MIN\_SIZE = 2;

const int MIN\_ELEM = -2000000000;

const int MAX\_ELEM = 2000000000;

void printInf()

{

cout << "Program sort even rows of square matrix from larger to smaller\n";

}

int inputNum(const int MIN, const int MAX)

{

int number;

bool isIncorrect;

do

{

isIncorrect = false;

cin >> number;

if (cin.fail())

{

cin.clear();

cout << "Data is not correct, or number is too large\n";

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

isIncorrect = true;

cout << "Please, enter again\n";

}

if (!isIncorrect && cin.get() != '\n')

{

cin.clear();

cout << "Data is not correct, or number is too large\n";

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

isIncorrect = true;

cout << "Please, enter again\n";

}

if (!isIncorrect && (number > MAX || number < MIN))

{

cout << "Error, number should be from " << MIN << " to " << MAX << '\n';

isIncorrect = true;

cout << "Please, enter again\n";

}

} while (isIncorrect);

return number;

}

int \*\*enterArr(int size)

{

int \*\*arr = new int\*[size];

int i, j;

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

{

arr[i] = new int[size];

}

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

{

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

{

cout << "Enter a" << i+1 << j+1 << " element\n";

arr[i][j] = inputNum(MIN\_ELEM, MAX\_ELEM);

}

}

return arr;

}

void printArr(int \*\*arr, int size)

{

int i, j;

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

{

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

{

cout << arr[i][j] << " ";

}

cout << '\n';

}

}

void sortArr(int \*arr, int size)

{

bool isNotSorted;

int i, j, buffer;

buffer = 0;

isNotSorted = true;

while (isNotSorted)

{

isNotSorted = false;

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

{

for (j = i; j < size; j++)

{

if (arr[j - 1] < arr[j])

{

isNotSorted = true;

buffer = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = buffer;

}

}

}

}

}

int\*\* copyArr(int\*\* mainArr, int size)

{

int i, j;

int\*\* copiedArr = new int\* [size];

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

{

copiedArr[i] = new int[size];

}

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

{

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

{

copiedArr[i][j] = mainArr[i][j];

}

}

return copiedArr;

}

int \*\*sortEvenRow(int \*\*arrOfNum, int size)

{

int i;

int \*\*arr = copyArr(arrOfNum, size);

for (i = 1; i < size; i += 2)

{

sortArr(arr[i], size);

}

return arr;

}

bool thisIsTxtFile(string fileName)

{

string lastFourChar = fileName.substr(fileName.length() - 4);

if (lastFourChar == ".txt")

{

return true;

}

else

{

cout << "it is not a .txt file\n";

return false;

}

}

bool isFileExist(string nameOfFile)

{

ifstream file(nameOfFile);

if (file.is\_open())

{

file.close();

return true;

}

else

{

cout << "this file is not exist\n";

file.close();

return false;

}

}

int readSizeFromFile(ifstream &file)

{

int size;

char next;

bool isCorrect = true;

file >> size;

if (file.fail())

{

cout << "size is incorrect";

file.clear();

isCorrect = false;

size = 0;

}

next = file.get();

if (isCorrect && ((next != ' ') && (next != '\n')))

{

size = 0;

isCorrect = false;

cout << "size is incorrect, remove other simbols or whitespase";

file.clear();

}

if (isCorrect && (size < MIN\_SIZE) || (size > MAX\_SIZE))

{

size = 0;

cout << "Size of array should be from " << MIN\_SIZE << " to " << MAX\_SIZE;

}

return size;

}

void chekAnyException(int &size, bool isCorrect, ifstream &file, bool isElemIncorrect)

{

if (isElemIncorrect)

{

size = 0;

cout << "One of the element is incorrect or out of range [ " << MIN\_ELEM << ", " << MAX\_ELEM << " ]";

}

else if (file.eof() && isCorrect)

cout << "Reading is successfull\n";

else if (isCorrect)

{

size = 0;

cout << "Count of element is too a lot\n";

}

else

{

size = 0;

cout << "Count of element is not enough\n";

}

}

int \*\*readFile(int &size, string name)

{

int i, j;

bool isCorrect, isElemIncorrect;

ifstream file(name);

size = readSizeFromFile(file);

isCorrect = size > 1;

isElemIncorrect = false;

int \*\*arr = new int\*[size];

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

{

arr[i] = new int[size];

}

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

{

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

{

if (file.eof())

isCorrect = false;

else

{

file >> arr[i][j];

if (file.fail())

isElemIncorrect = true;

else if (arr[i][j] < MIN\_ELEM || arr[i][j] > MAX\_ELEM)

isElemIncorrect = true;

}

}

}

chekAnyException(size, isCorrect, file, isElemIncorrect);

file.close();

return arr;

}

void writeFile(int \*\*defaultArr, int \*\*sortedArr, int size, string name)

{

int i, j;

ofstream file(name);

file << "Default array\n";

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

{

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

{

file << defaultArr[i][j] << " ";

}

file << '\n';

}

file << "Sorted array\n";

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

{

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

{

file << sortedArr[i][j] << " ";

}

file << '\n';

}

file.close();

cout << "Writing is successfull";

}

bool isFileOk(string name)

{

return isFileExist(name) && thisIsTxtFile(name);

}

int userChoice()

{

int choice;

cout << "Choose a way of input/output of data\n"

<< "1 -- Console\n"

<< "2 -- File\n";

choice = inputNum(1, 2);

return choice;

}

int \*\*inputFromConsole(int &size)

{

cout << "Enter size of array, please\n";

size = inputNum(MIN\_SIZE, MAX\_SIZE);

int\*\* arr = new int\* [size];

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

{

arr[i] = new int[size];

}

cout << "Now enter the elements\n";

arr = enterArr(size);

return arr;

}

int \*\*inputFromFile(int &size)

{

bool isIncorrect;

int \*\*arr = new int\*;

string name;

do

{

isIncorrect = false;

cout << "Enter full path to file\n";

cin >> name;

if (isFileOk(name))

{

arr = readFile(size, name);

}

else

{

isIncorrect = true;

}

} while (isIncorrect);

return arr;

}

int \*\*inputInf(int &size)

{

int \*\*arr = new int\*;

int choiceInp = userChoice();

if (choiceInp == 1)

arr = inputFromConsole(size);

else

arr = inputFromFile(size);

return arr;

}

void outputInConsole(int \*\*defaultArr, int \*\*sortedArr, int size)

{

cout << "Default array\n";

printArr(defaultArr, size);

cout << "Sorted array\n";

printArr(sortedArr, size);

}

void outputInFile(int\*\* defaultArr, int\*\* sortedArr, int size)

{

bool isIncorrect;

string name;

do

{

isIncorrect = false;

cout << "Enter full path to file\n";

cin >> name;

if (isFileOk(name))

{

writeFile(defaultArr, sortedArr, size, name);

}

else

{

isIncorrect = true;

}

} while (isIncorrect);

}

void outputInf(int \*\*defaultArr, int \*\*sortedArr, int size)

{

if (size > 1)

{

int choiceOut = userChoice();

if (choiceOut == 1)

{

outputInConsole(defaultArr, sortedArr, size);

}

else

{

outputInFile(defaultArr, sortedArr, size);

}

}

}

int main()

{

int \*\*arrOfNum;

int \*\*sortedArr;

int size;

printInf();

arrOfNum = inputInf(size);

sortedArr = sortEvenRow(arrOfNum, size);

outputInf(arrOfNum, sortedArr, size);

return 0;

}

**Код программы Java:**

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Scanner;

public class Main {

static final int MAX\_SIZE = 100;

static final int MIN\_SIZE = 2;

static final int MIN\_ELEM = -2000000000;

static final int MAX\_ELEM = 2000000000;

static void printInf() {

System.out.println("Program sort even rows of square matrix from larger to smaller");

}

static int inputNum(Scanner input, final int MIN, final int MAX) {

int number = 0;

boolean isIncorrect;

do {

isIncorrect = false;

try {

number = Integer.parseInt(input.next());

} catch (NumberFormatException e) {

isIncorrect = true;

System.err.println("Data is not correct, or number is too large");

System.out.println("Please, enter again");

}

if (!isIncorrect && (number < MIN || number > MAX)) {

isIncorrect = true;

System.err.println("Error, number should be from " + MIN + " to " + MAX);

System.out.println("Please, enter again");

}

} while (isIncorrect);

return number;

}

static int[][] enterArr(Scanner input, int row, int col) {

int[][] arr = new int[row][col];

int i, j;

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

for (j = 0; j < col; j++) {

System.out.println("Enter a" + (i+1) + (j+1) + " element");

arr[i][j] = inputNum(input, MIN\_ELEM, MAX\_ELEM);

}

}

return arr;

}

static void printArr(int[][] arr) {

int i, j;

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

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

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

}

System.out.println();

}

}

static void sortArr(int[] arr) {

boolean isNotSorted;

int i, j, buffer;

isNotSorted = true;

while (isNotSorted) {

isNotSorted = false;

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

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

if (arr[j - 1] < arr[j]) {

isNotSorted = true;

buffer = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = buffer;

}

}

}

}

}

static int[][] sortEvenRow(int[][] arrOfNum)

{

int i;

int[][] arr = copyArr(arrOfNum);

for (i = 1; i < arr.length; i += 2)

{

sortArr(arr[i]);

}

return arr;

}

static int[][] copyArr(int[][] mainArr)

{

int i, j;

int size = mainArr.length;

int[][] copiedArr = new int[size][size];

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

{

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

{

copiedArr[i][j] = mainArr[i][j];

}

}

return copiedArr;

}

static boolean thisIsTxtFile(String fileName) {

if (fileName.endsWith(".txt")) {

return true;

} else {

System.err.println("This is not a .txt file");

return false;

}

}

static boolean isFileExist(String fileName) {

File file = new File(fileName);

if (file.exists()) {

return true;

} else {

System.err.println("This file is not exist");

return false;

}

}

static int readSizeFromFile(Scanner file) {

int size = 0;

boolean isCorrect = true;

try {

size = Integer.parseInt(file.next());

} catch (NumberFormatException e) {

System.err.println("Data is not correct, or number is too large");

isCorrect = false;

}

if (isCorrect && (size < MIN\_SIZE || size > MAX\_SIZE)) {

System.err.println("Error, number should be from " + MIN\_SIZE + " to " + MAX\_SIZE);

size = 0;

}

return size;

}

static boolean isAnyException(boolean isCorrect, boolean isElemIncorrect) {

boolean anyException = !isCorrect;

if (isElemIncorrect) {

System.err.println("One of the element is incorrect or out of range [ " + MIN\_ELEM + ", " + MAX\_ELEM + " ]");

anyException = true;

} else if (isCorrect) {

System.out.println("Reading is successfull");

}

return anyException;

}

static int[][] readFile(String fileName) throws IOException{

int size;

boolean isCorrect;

Path path = Paths.get(fileName);

Scanner file = new Scanner(path);

size = readSizeFromFile(file);

isCorrect = size > 1;

int i, j;

boolean isElemIncorrect = false;

int[][] arr = new int[size][size];

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

for ( j = 0; j < size; j++) {

try {

arr[i][j] = Integer.parseInt(file.next());

} catch (NumberFormatException e) {

isElemIncorrect = true;

}

if (!isElemIncorrect && (arr[i][j] > MAX\_ELEM || arr[i][j] < MIN\_ELEM)) {

isElemIncorrect = true;

}

}

}

if (isAnyException(isCorrect, isElemIncorrect)) {

return new int[0][0];

} else {

return arr;

}

}

static int userChoice(Scanner input) {

int choice;

System.out.println("Choose a way of input/output of data\n"

+ "1 -- Console\n"

+ "2 -- File");

choice = inputNum(input,1, 2);

return choice;

}

static String getFileName(Scanner input) {

boolean isIncorrect;

String fileName;

fileName = input.nextLine();

do {

System.out.println("Enter full path to file");

fileName = input.nextLine();

isIncorrect = !thisIsTxtFile(fileName) || !isFileExist(fileName);

} while (isIncorrect);

return fileName;

}

static int[][]inputFromConsole(Scanner input) {

boolean isIncorrect;

int size;

int[][] arr;

System.out.println("Enter size of array, please");

size = inputNum(input, MIN\_SIZE, MAX\_SIZE);

System.out.println("Now enter the elements");

arr = enterArr(input, size, size);

return arr;

}

static int[][] inputFromFile(Scanner input) {

boolean isIncorrect;

int size;

String fileName = getFileName(input);

int[][] arr = null;

try {

arr = readFile(fileName);

} catch (IOException e) {

System.err.println("IOException");

}

return arr;

}

static int[][] inputInf(Scanner input) {

int choice = userChoice(input);

int[][] arr;

if (choice == 1) {

arr = inputFromConsole(input);

} else {

arr = inputFromFile(input);

}

return arr;

}

static void writeFile(int[][] defaultArr, int[][] sortedArr, String fileName) throws IOException {

int i, j;

PrintWriter file = new PrintWriter(fileName);

file.println("Default array");

for (i = 0; i < defaultArr.length; i++){

for (j = 0; j < defaultArr[i].length; j++) {

file.print(defaultArr[i][j] + " ");

}

file.println();

}

file.println("Sorted array");

for (i = 0; i < sortedArr.length; i++){

for (j = 0; j < sortedArr[i].length; j++) {

file.print(sortedArr[i][j] + " ");

}

file.println();

}

file.close();

System.out.println("Writing is successfull");

}

static void outputInConsole(int[][] defaultArr, int[][] sortedArr) {

System.out.println("Default array");

printArr(defaultArr);

System.out.println("Sorted array");

printArr(sortedArr);

}

static void outputInFile(int[][] defaultArr, int[][] sortedArr, Scanner input) {

String fileName = getFileName(input);

try {

writeFile(defaultArr, sortedArr, fileName);

} catch(IOException e) {

System.err.println("Write error");

}

}

static void outputInf(int[][] defaultArr, int[][] sortedArr, Scanner input) {

if (defaultArr.length > 1) {

int choice = userChoice(input);

if (choice == 1) {

outputInConsole(defaultArr, sortedArr);

} else {

outputInFile(defaultArr, sortedArr, input);

}

}

}

public static void main(String[] args) throws IOException {

Scanner input = new Scanner(System.in);

int[][] arrOfNum;

int[][] sortedArr;

printInf();

arrOfNum = inputInf(input);

sortedArr = sortEvenRow(arrOfNum);

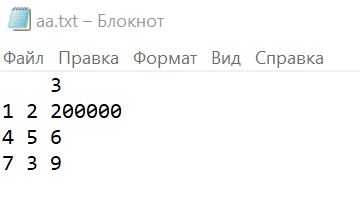
outputInf(arrOfNum, sortedArr, input);

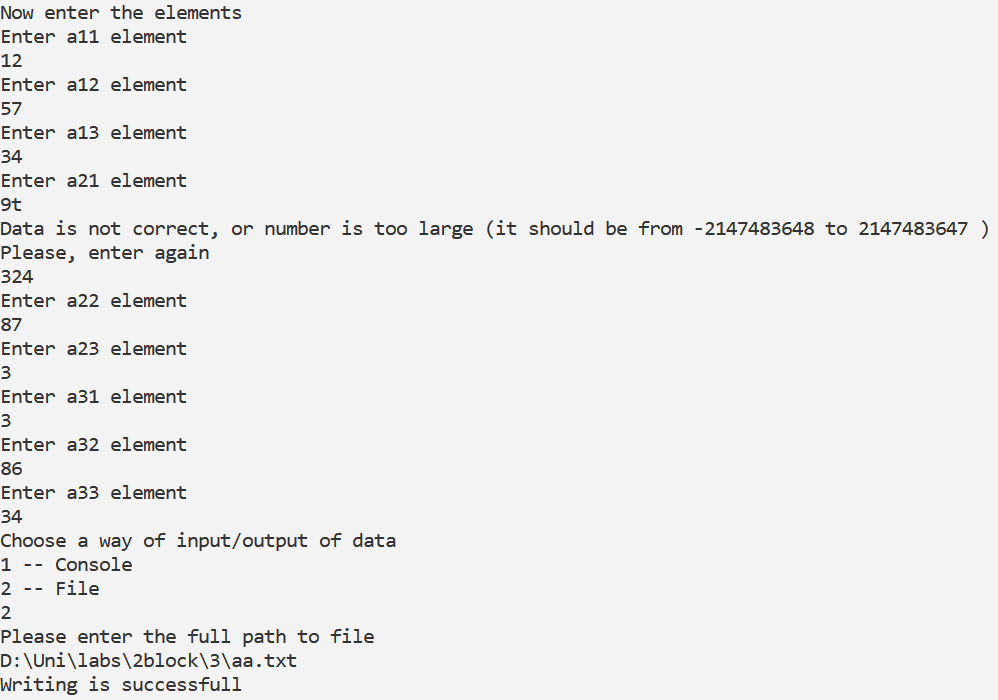
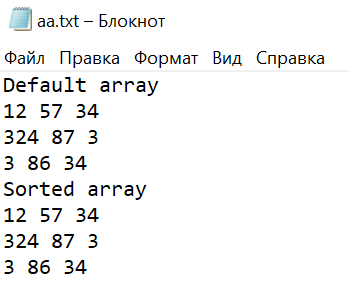
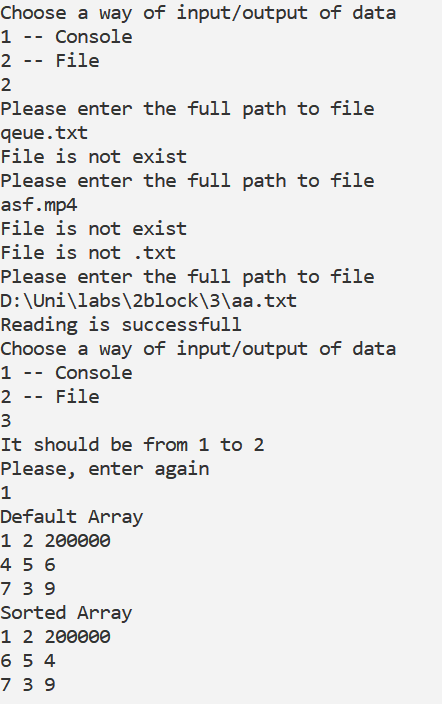
input.close();

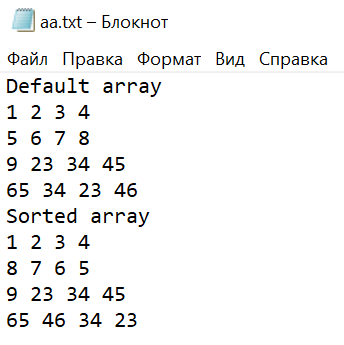
}

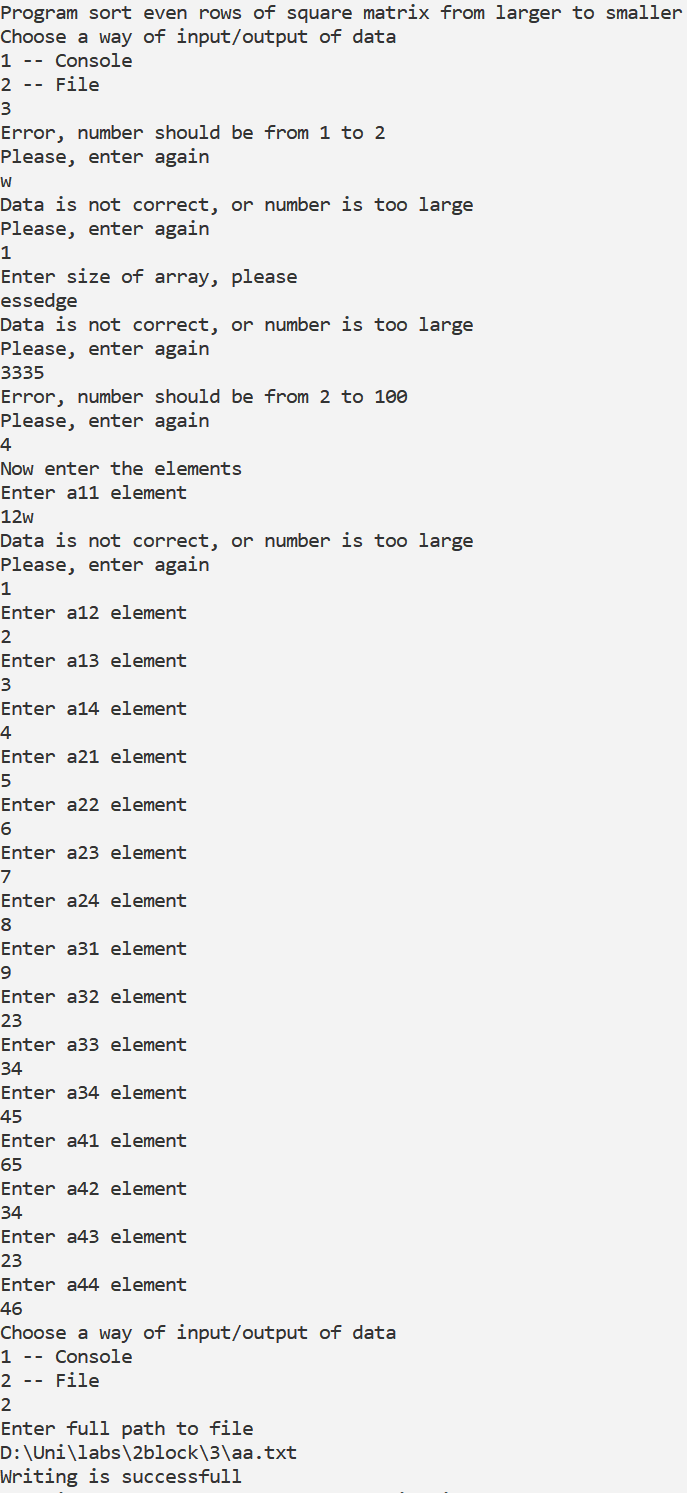
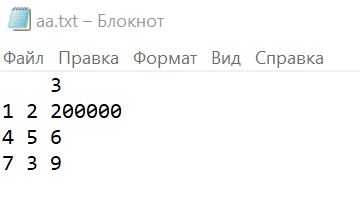
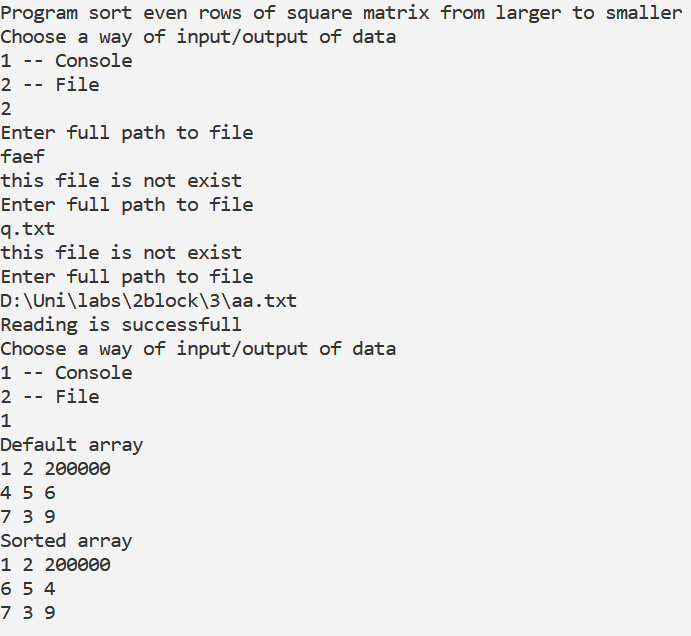
}

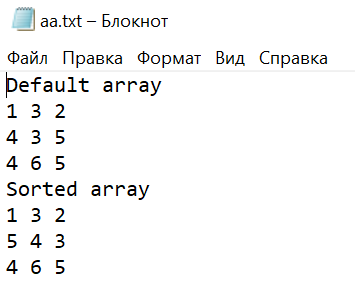
**Скриншоты:**

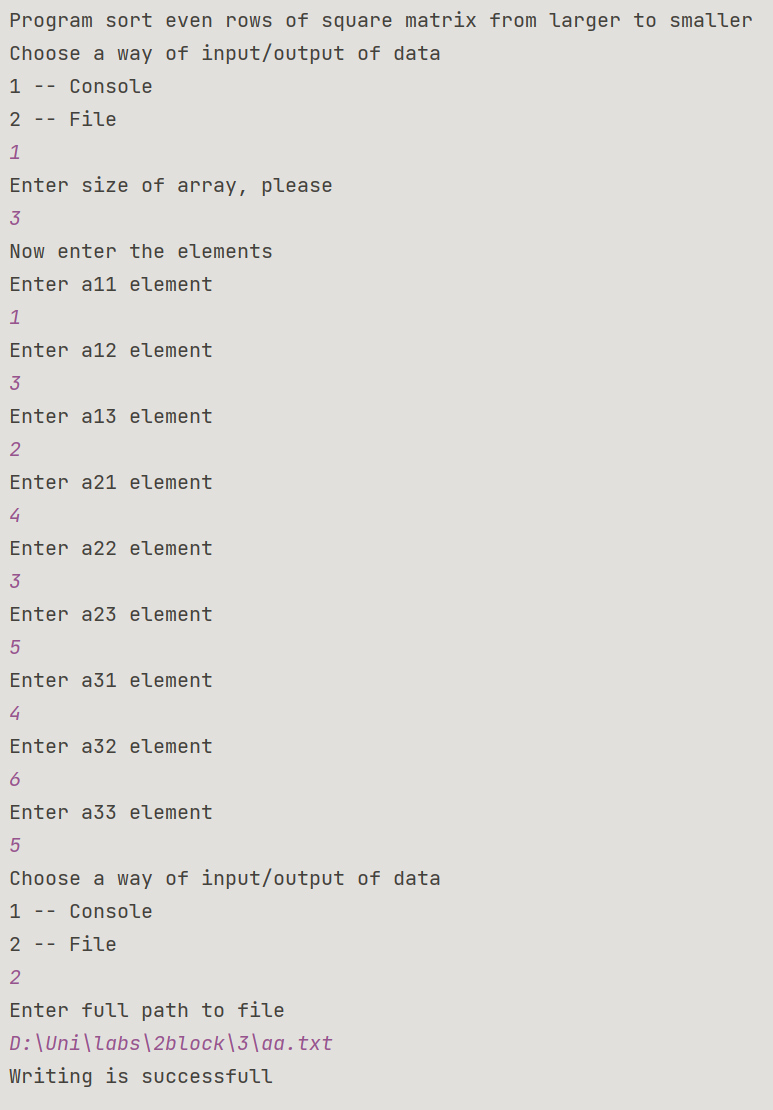
**Delphi:**

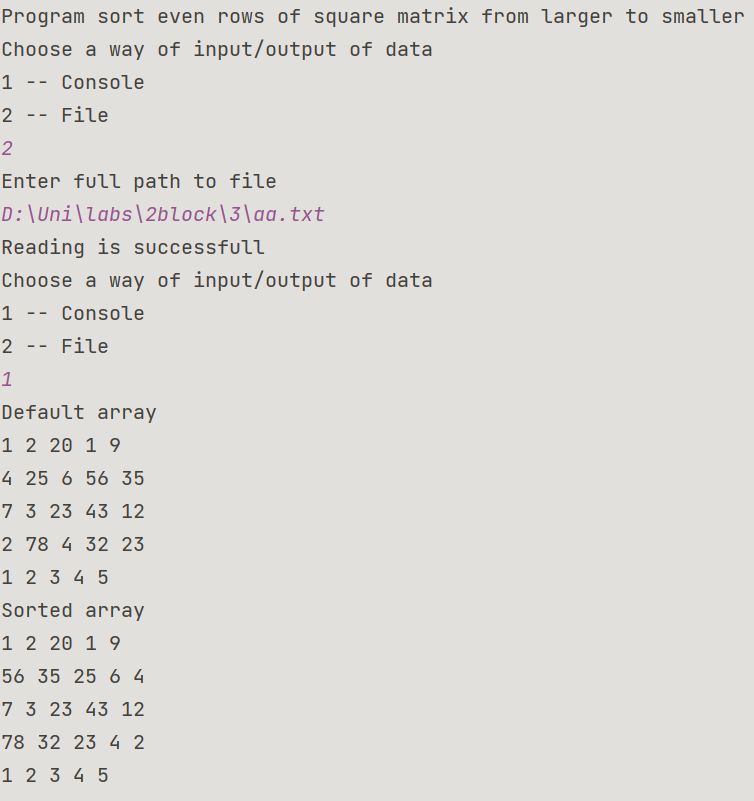
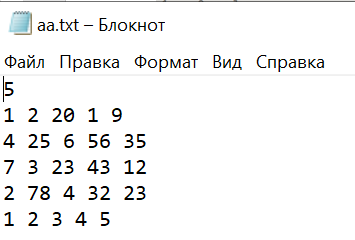
****

**C++:**

****

**Java:**

****

****

**Блок-схема:**













