УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.3

по предмету «Основы алгоритмизации и программирования»

Вариант 5

Выполнил:

Бражалович А. И.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Дана матрица А, имеющая 30 строк и 10 столбцов, содержащая оценки группы за первый семестр. Напечатать номера отличников (оценки не ниже 8).

**Код программы Delphi:**

Program Lab2\_3;

Uses

System.SysUtils;

Type

TMatrix = Array [1..30, 1..10] Of Integer;

TGoodMen = Array [1..30] of Integer;

Const

MIN\_MARK = 0;

MAX\_MARK = 10;

FIRST\_STUDENT = 1;

LAST\_STUDENT = 30;

LAST\_COLUMN = 10;

FIRST\_COLUMN = 1;

Procedure PrintTask();

Begin

WriteLn('Данная программа выводит номера отличников за семестр.');

WriteLn;

End;

Function ChooseFileInput(): Boolean;

Var

IsFileInput: Integer;

IsCorrect, Choose: Boolean;

Begin

IsFileInput := 0;

IsCorrect := False;

Choose := False;

Repeat

WriteLn('Вы хотите вводить матрицу через файл? (Да - ', 1, ' / Нет - ', 0, ')');

Try

ReadLn(IsFileInput);

IsCorrect := True;

Except

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

If IsCorrect Then

Begin

If IsFileInput = 1 Then

Choose := True

Else If IsFileInput = 0 Then

Choose := False

Else

Begin

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

End;

Until IsCorrect;

ChooseFileInput := Choose;

End;

Function CheckArea(Num: Integer; Const MIN, MAX: Integer) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

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

Begin

Writeln('Значение не попадает в диапазон!');

IsCorrect := False;

End;

CheckArea := IsCorrect;

End;

Function ReadPathToFile() : String;

Var

PathToFile: String;

IsCorrect: Boolean;

Begin

PathToFile := '';

IsCorrect := False;

Repeat

Write('Введите путь к файлу с расширением .txt с матрицей, у которой числа должны соответствовать оценкам[', MIN\_MARK, ':', MAX\_MARK,']: ');

ReadLn(PathToFile);

If ExtractFileExt(PathToFile) = '.txt' Then

IsCorrect := True

Else

Begin

WriteLn('Расширение файла не .txt!');

IsCorrect := False;

End;

Until IsCorrect;

ReadPathToFile := PathToFile;

End;

Function IsExists(PathToFile: String) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := False;

If FileExists(PathToFile) Then

IsCorrect := True;

IsExists := IsCorrect;

End;

Function IsAbleToReading(Var F: TextFile) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

Try

Reset(F);

CloseFile(F);

Except

IsCorrect := False;

End;

IsAbleToReading := IsCorrect;

End;

Function IsAbleToWriting(PathToFile: String) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

If FileIsReadOnly(PathToFile) Then

IsCorrect := False;

IsAbleToWriting := IsCorrect;

End;

Function IsEmpty(Var F: TextFile) : Boolean;

Var

Size: Integer;

IsCorrect: Boolean;

Begin

Size := 0;

IsCorrect := False;

Reset(F);

If Not EOF(F) Then

Size := 1;

CloseFile(F);

If Size = 0 Then

IsCorrect := True;

IsEmpty := IsCorrect;

End;

Function IsRightFileNums(Var F: TextFile) : Boolean;

Var

Mark: Integer;

IsCorrect: Boolean;

Begin

Mark := 0;

IsCorrect := True;

Reset(F);

While IsCorrect And Not EOF(F) Do

Begin

While IsCorrect And Not EOLN(F) Do

Begin

Try

Read(F, Mark);

Except

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckArea(Mark, MIN\_MARK, MAX\_MARK);

End;

ReadLn(F);

End;

CloseFile(F);

IsRightFileNums := IsCorrect;

End;

Procedure GetFileNormalReading(Var F: TextFile);

Var

PathToFile: String;

IsCorrect: Boolean;

Begin

PathToFile := '';

IsCorrect := True;

Repeat

PathToFile := ReadPathToFile();

AssignFile(F, PathToFile);

If Not IsExists(PathToFile) Then

Begin

IsCorrect := False;

Writeln('Проверьте корректность ввода пути к файлу!');

End;

If IsCorrect And Not IsAbleToReading(F) Then

Begin

IsCorrect := False;

Writeln('Файл закрыт для чтения!');

End;

If IsCorrect And IsEmpty(F) Then

Begin

IsCorrect := False;

WriteLn('Файл пуст!');

End;

If IsCorrect And Not IsRightFileNums(F) Then

Begin

IsCorrect := False;

WriteLn('Некорректный тип данных внутри файла!');

End;

Until IsCorrect;

End;

Procedure GetFileNormalWriting(Var F: TextFile);

Var

PathToFile: String;

IsCorrect: Boolean;

Begin

PathToFile := '';

IsCorrect := True;

Repeat

PathToFile := ReadPathToFile();

AssignFile(F, PathToFile);

If Not IsExists(PathToFile) Then

Begin

IsCorrect := False;

Writeln('Проверьте корректность ввода пути к файлу!');

End;

If IsCorrect And Not IsAbleToWriting(PathToFile) Then

Begin

IsCorrect := False;

WriteLn('Файл закрыт для записи!');

End;

Until IsCorrect;

End;

Function ReadFileMatrix(Var F: TextFile) : TMatrix;

Var

Matrix: TMatrix;

Row, Col: Integer;

Begin

Reset(F);

Readln(F);

For Row := FIRST\_STUDENT To LAST\_STUDENT Do

Begin

For Col := FIRST\_COLUMN To LAST\_COLUMN Do

Read(F, Matrix[Row][Col]);

Readln(F);

End;

CloseFile(F);

ReadFileMatrix := Matrix

End;

Function ReadConsoleMatrix() : TMatrix;

Var

Matrix: TMatrix;

Row, Col: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := False;

For Row := FIRST\_STUDENT To LAST\_STUDENT Do

For Col := FIRST\_COLUMN To LAST\_COLUMN Do

Begin

Repeat

Write('Введите оценку ', (Row), ' человека за ', (Col), ' день:');

Try

Readln(Matrix[Row][Col]);

IsCorrect := True;

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckArea(Matrix[Row][Col], MIN\_MARK, MAX\_MARK);

Until IsCorrect;

End;

ReadConsoleMatrix := Matrix;

End;

Procedure ReadMatrix(Var Matrix: TMatrix);

Var

RF: TextFile;

Begin

If ChooseFileInput() Then

Begin

GetFileNormalReading(RF);

Matrix := ReadFileMatrix(RF);

End

Else

Matrix := ReadConsoleMatrix();

End;

Function FindGoodMen (Var Matrix: TMatrix; Var GoodMen: TGoodMen): TGoodMen;

Var

GoodMarks: Boolean;

Row, Col: Integer;

Begin

Row := 0;

Col := 0;

GoodMarks := True;

For Row := FIRST\_STUDENT To LAST\_STUDENT Do

Begin

For Col := FIRST\_COLUMN To LAST\_COLUMN Do

Begin

If Matrix[Row][Col] < 9 Then

GoodMarks := False;

End;

If GoodMarks Then

GoodMen[Row] := Row + 1;

GoodMarks := True;

End;

FindGoodMen := GoodMen;

End;

Function ChooseFileOutput() : Boolean;

Var

IsFileOutput: Integer;

IsCorrect, Choose: Boolean;

Begin

IsFileOutput := 0;

IsCorrect := False;

Choose := False;

Repeat

WriteLn('Вы хотите выводить матрицу через файл? (Да - ', 1, ' / Нет - ', 0, ')');

Try

ReadLn(IsFileOutput);

IsCorrect := True;

Except

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

If IsCorrect Then

Begin

If IsFileOutput = 1 Then

Choose := True

Else If IsFileOutput = 0 Then

Choose := False

Else

Begin

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

End;

Until IsCorrect;

ChooseFileOutput := Choose;

End;

Procedure PrintConsoleResult(GoodMen: TGoodMen);

Var

Row: Integer;

Begin

Row := 0;

WriteLn('Номера отличников:');

For Row := FIRST\_STUDENT To LAST\_STUDENT Do

Begin

If GoodMen[Row] > 0 Then

Write(GoodMen[Row], ' ');

End;

End;

Procedure PrintFileResult(Var F: TextFile; GoodMen: TGoodMen);

Var

Row: Integer;

Begin

Append(F);

Row := 0;

WriteLn(F, 'Номера отличников:');

For Row := FIRST\_STUDENT To LAST\_STUDENT Do

Begin

If GoodMen[Row] > 0 Then

Write(F, GoodMen[Row], ' ');

End;

CloseFile(F);

End;

Procedure PrintResult(Var GoodMen: TGoodMen);

Var

WF: TextFile;

Begin

If ChooseFileOutput() Then

Begin

GetFileNormalWriting(WF);

PrintFileResult(WF, GoodMen);

End

Else

PrintConsoleResult(GoodMen);

End;

Var

Matrix: TMatrix;

GoodMen: TGoodMen;

Begin

PrintTask();

ReadMatrix(Matrix);

FindGoodMen(Matrix, GoodMen);

PrintResult(GoodMen);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <string.h>

#include <fstream>

using std::cin;

using std::cout;

using std::string;

using std::fstream;

using std::ifstream;

using std::ofstream;

using std::ios;

using std::numeric\_limits;

using std::streamsize;

const int

MIN\_MARK = 0,

MAX\_MARK = 10,

FIRST\_STUDENT = 1,

LAST\_STUDENT = 30,

LAST\_COLUMN = 10,

FIRST\_COLUMN = 1;

void printTask()

{

cout << "Данная программа выводит номера отличников за семестр.\n\n";

}

void createMatrix(int\*\*& matrix)

{

matrix = new int\* [LAST\_STUDENT];

for (int i = 0; i < LAST\_STUDENT; i++)

matrix[i] = new int[LAST\_COLUMN];

}

void createGoodMen(int\*& goodMen)

{

goodMen = new int [LAST\_STUDENT];

}

bool chooseFileInput()

{

int isFileInput;

bool isCorrect;

bool choose;

isFileInput = 0;

choose = false;

do {

isCorrect = true;

cout << "Вы хотите вводить матрицу через файл? (Да - " << 1 << " / Нет - " << 0 << ")\n";

cin >> isFileInput;

if (cin.fail())

{

isCorrect = false;

cout << "Некорректный выбор!\n";

cin.clear();

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

}

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

{

isCorrect = false;

cout << "Некорректный выбор!\n";

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

}

if (isCorrect)

{

if (isFileInput == 1)

choose = true;

else if (isFileInput == 0)

choose = false;

else

{

isCorrect = false;

cout << "Некорректный выбор!\n";

}

}

} while (!isCorrect);

return choose;

}

bool checkArea(int num, const int MIN, const int MAX)

{

bool isCorrect;

isCorrect = true;

if (num < MIN || num > MAX)

{

cout << "Значение не попадает в диапазон!\n";

isCorrect = false;

}

return isCorrect;

}

string readPathFile()

{

string pathToFile;

bool isCorrect;

int len;

pathToFile = "";

isCorrect = false;

len = 0;

do

{

cout << "Введите путь к файлу с расширением .txt с матрицей, у которой числа должны соответствовать оценкам[" << MIN\_MARK << ":" << MAX\_MARK << "]: ";

cin >> pathToFile;

len = pathToFile.length();

if (len > 4 && pathToFile.substr(len - 4, 4) == ".txt")

isCorrect = true;

else

{

cout << "Расширение файла не .txt!\n";

isCorrect = false;

}

} while (!isCorrect);

return pathToFile;

}

bool isExists(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ifstream file(pathToFile);

if (file.good())

isCorrect = true;

file.close();

return isCorrect;

}

bool isAbleToReading(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ifstream file(pathToFile);

if (file.is\_open())

isCorrect = true;

file.close();

return isCorrect;

}

bool isAbleToWriting(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ofstream file(pathToFile, ios::app);

if (file.is\_open())

isCorrect = true;

file.close();

return isCorrect;

}

bool isEmpty(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ifstream file(pathToFile);

if (file.peek() == ifstream::traits\_type::eof())

isCorrect = true;

file.close();

return isCorrect;

}

bool isRightFileNums(string pathToFile)

{

int mark;

bool isCorrect;

mark = 0;

isCorrect = true;

ifstream file(pathToFile);

while (isCorrect && !file.eof())

{

while (isCorrect && file.peek() != '\n' && !file.eof())

{

file >> mark;

if (file.fail())

{

isCorrect = false;

file.clear();

}

if (isCorrect)

isCorrect = checkArea(mark, MIN\_MARK, MAX\_MARK);

}

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

}

file.close();

return isCorrect;

}

void getFileNormalReading(string& pathToFile)

{

bool isCorrect;

do

{

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile))

{

isCorrect = false;

cout << "Проверьте корректность ввода пути к файлу!\n";

}

if (isCorrect && !isAbleToReading(pathToFile))

{

isCorrect = false;

cout << "Файл закрыт для чтения!\n";

}

if (isCorrect && !isAbleToWriting(pathToFile))

{

isCorrect = false;

cout << "Файл закрыт для записи!\n";

}

if (isCorrect && isEmpty(pathToFile))

{

isCorrect = false;

cout << "Файл пуст!\n";

}

if (isCorrect && !isRightFileNums(pathToFile))

{

isCorrect = false;

cout << "Некорректный тип данных внутри файла!\n";

}

} while (!isCorrect);

}

void getFileNormalWriting(string& pathToFile)

{

bool isCorrect;

do

{

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile))

{

isCorrect = false;

cout << "Проверьте корректность ввода пути к файлу!\n";

}

if (isCorrect && !isAbleToWriting(pathToFile))

{

isCorrect = false;

cout << "Файл закрыт для записи!\n";

}

} while (!isCorrect);

}

int\*\* readFileMatrix(string pathToFile)

{

int\*\* matrix;

createMatrix(matrix);

ifstream file(pathToFile);

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

for (int row = 0; row < LAST\_STUDENT; row++)

{

for (int col = 0; col < LAST\_COLUMN; col++)

file >> matrix[row][col];

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

}

file.close();

return matrix;

}

int\*\* readConsoleMatrix()

{

int\*\* matrix;

bool isCorrect;

createMatrix(matrix);

for (int row = 0; row < LAST\_STUDENT ; row++)

for (int col = 0; col < LAST\_COLUMN; col++)

do

{

isCorrect = true;

cout <<" Введите оценку " << row + 1 << " человека за " << col + 1 << " день: ";

cin >> matrix[row][col];

if (cin.fail())

{

isCorrect = false;

cin.clear();

cout << "Проверьте корректность ввода данных!\n";

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

}

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

{

isCorrect = false;

cout << "Проверьте корректность ввода данных!\n";

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

}

if (isCorrect)

isCorrect = checkArea(matrix[row][col], MIN\_MARK, MAX\_MARK);

} while (!isCorrect);

return matrix;

}

void readMatrix(int\*\*& matrix)

{

string pathToFile;

pathToFile = "";

if (chooseFileInput())

{

getFileNormalReading(pathToFile);

matrix = readFileMatrix(pathToFile);

}

else

matrix = readConsoleMatrix();

}

int\* findGoodMen(int\*\*& matrix, int\*& goodMen)

{

int row;

int col;

bool goodMarks;

row = 0;

col = 0;

goodMarks = true;

for (row = FIRST\_STUDENT; row < LAST\_STUDENT; row++)

{

for (col = FIRST\_COLUMN; col < LAST\_COLUMN; col++)

{

if (matrix[row][col] < 9)

goodMarks = false;

}

if (goodMarks)

goodMen[row] = row + 2;

goodMarks = true;

}

return goodMen;

}

bool chooseFileOutput()

{

int isFileOutput;

bool isCorrect;

bool choose;

isFileOutput = 0;

choose = false;

cout << "\n";

do {

isCorrect = true;

cout << "Вы хотите выводить матрицу через файл? (Да - " << 1 << " / Нет - " << 0 << ")\n";

cin >> isFileOutput;

if (cin.fail())

{

isCorrect = false;

cout << "Некорректный выбор!\n";

cin.clear();

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

}

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

{

isCorrect = false;

cout << "Некорректный выбор!\n";

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

}

if (isCorrect)

{

if (isFileOutput == 1)

choose = true;

else if (isFileOutput == 0)

choose = false;

else

{

isCorrect = false;

cout << "Некорректный выбор!\n";

}

}

} while (!isCorrect);

return choose;

}

void printConsoleResult(int\* goodMen)

{

int row;

row = 0;

cout << "Номера отличников: ";

for (row = 0; row < LAST\_STUDENT; row++)

{

if (goodMen[row] > 0)

cout << goodMen[row] << " ";

}

}

void printFileResult(string pathToFile, int\* goodMen)

{

int row;

ofstream file(pathToFile, ios::app);

row = 0;

file << "Номера отличников:\n";

for (row = 0; row < LAST\_STUDENT; row++)

{

if (goodMen[row] > 0)

file << goodMen[row] << " ";

}

file.close();

}

void printResult(int\* goodMen)

{

string pathToFile;

pathToFile = "";

if (chooseFileOutput())

{

getFileNormalWriting(pathToFile);

printFileResult(pathToFile, goodMen);

}

else

printConsoleResult(goodMen);

}

void freeMemory(int\*\*& matrix, int\*& goodMen)

{

for (int i = 0; i < LAST\_COLUMN; i++)

delete[] matrix[i];

delete[] matrix;

delete[] goodMen;

}

int main()

{

int\*\* matrix;

int\* goodMen;

createGoodMen(goodMen);

setlocale(LC\_ALL, "RU");

printTask();

readMatrix(matrix);

findGoodMen(matrix, goodMen);

printResult(goodMen);

freeMemory(matrix, goodMen);

return 0;

}

**Код программы Java:**

import java.util.Scanner;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

public class Main {

public static final int

MIN\_MARK = 0,

MAX\_MARK = 10,

FIRST\_STUDENT = 1,

LAST\_STUDENT = 30,

LAST\_COLUMN = 10,

FIRST\_COLUMN = 1;

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

static Scanner scanFile;

static File file;

public static void printTask() {

System.out.println("Данная программа выводит номера отличников за семестр.\n\n");

}

public static boolean chooseFileInput() {

int isFileInput;

boolean isCorrect, choose;

isFileInput = 0;

choose = true;

do {

System.out.println("Вы хотите вводить матрицу через файл? (Да - " + 1 + " / Нет - " + 0 + ")");

isCorrect = true;

try {

isFileInput = Integer.parseInt(scanConsole.nextLine());

} catch (NumberFormatException e) {

System.out.println("Некорректный выбор!");

isCorrect = false;

}

if (isCorrect) {

if (isFileInput == 1)

choose = true;

else if (isFileInput == 0)

choose = false;

else {

isCorrect = false;

System.out.println("Некорректный выбор!");

}

}

} while (!isCorrect);

return choose;

}

public static boolean checkArea(int num, final int MIN, final int MAX) {

boolean isCorrect;

isCorrect = true;

if (num < MIN || num > MAX) {

System.out.println("Значение не попадает в диапазон!\n");

isCorrect = false;

}

return isCorrect;

}

public static String readPathFile() {

boolean isCorrect;

int len;

String pathToFile;

isCorrect = false;

len = 0;

pathToFile = "";

do {

System.out.println("Введите путь к файлу с расширением .txt с матрицей, у которой разряд не должен превышать , а её натуральные элементы должны лежать в пределе: ");

pathToFile = scanConsole.nextLine();

len = pathToFile.length();

if (len > 4 && pathToFile.substring(len - 4).equals(".txt"))

isCorrect = true;

else

{

isCorrect = false;

System.out.println("Расширение файла не .txt!");

}

} while (!isCorrect);

return pathToFile;

}

public static boolean isExists(String pathToFile) {

boolean isCorrect;

isCorrect = false;

file = new File(pathToFile);

if (file.exists())

isCorrect = true;

return isCorrect;

}

public static boolean isAbleToReading() {

boolean isCorrect;

isCorrect = false;

if (file.canRead())

isCorrect = true;

return isCorrect;

}

public static boolean isAbleToWriting() {

boolean isCorrect;

isCorrect = false;

if (file.canWrite())

isCorrect = true;

return isCorrect;

}

public static boolean isEmpty() {

boolean isCorrect;

isCorrect = false;

if (file.length() == 0)

isCorrect = true;

return isCorrect;

}

public static boolean isRightFileNums() {

int mark;

boolean isCorrect;

isCorrect = true;

mark = 0;

try {scanFile = new Scanner(file);} catch (FileNotFoundException e) {}

try {

mark = scanFile.nextInt();

} catch (NumberFormatException e) {

System.out.println("Некорректный тип данных внутри файла!");

isCorrect = false;

}

if (isCorrect)

isCorrect = checkArea(mark, MIN\_MARK, MAX\_MARK);

scanFile.nextLine();

while (isCorrect && scanFile.hasNextInt()) {

try {

mark = scanFile.nextInt();

} catch (NumberFormatException e) {

System.out.println("Некорректный тип данных внутри файла!");

isCorrect = false;

}

if (isCorrect)

isCorrect = checkArea(mark, MIN\_MARK, MAX\_MARK);

}

scanFile.close();

return isCorrect;

}

public static void getFileNormalReading() {

boolean isCorrect;

String pathToFile;

pathToFile = "";

do {

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile)) {

isCorrect = false;

System.out.println("Проверьте корректность ввода пути к файлу!");

}

if (isCorrect && !isAbleToReading()) {

isCorrect = false;

System.out.println("Файл закрыт для чтения!");

}

if (isCorrect && !isAbleToWriting()) {

isCorrect = false;

System.out.println("Файл закрыт для записи!");

}

if (isCorrect && isEmpty()) {

isCorrect = false;

System.out.println("Файл пуст!");

}

if (isCorrect && !isRightFileNums()) {

isCorrect = false;

System.out.println("Некорректный тип данных внутри файла!");

}

} while (!isCorrect);

}

public static void getFileNormalWriting() {

boolean isCorrect;

String pathToFile;

pathToFile = "";

do {

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile)) {

isCorrect = false;

System.out.println("Проверьте корректность ввода пути к файлу!");

}

if (isCorrect && !isAbleToWriting()) {

isCorrect = false;

System.out.println("Файл закрыт для записи!");

}

} while (!isCorrect);

}

public static int[][] readFileMatrix() {

int[][] matrix;

matrix = new int[LAST\_STUDENT][LAST\_COLUMN];

try {scanFile = new Scanner(file);} catch (FileNotFoundException e) {}

for (int row = 0; row < LAST\_STUDENT; row++)

for (int col = 0; col < LAST\_COLUMN; col++)

matrix[row][col] = scanFile.nextInt();

scanFile.close();

return matrix;

}

public static int[][] readConsoleMatrix() {

int[][] matrix;

boolean isCorrect;

matrix = new int[LAST\_STUDENT][LAST\_COLUMN];

for (int row = 0; row < LAST\_STUDENT; row++)

for (int col = 0; col < LAST\_COLUMN; col++)

do {

System.out.print("Введите оценку " + (row + 1) + " человека за " + (col + 1) + " за " );

isCorrect = true;

try {

matrix[row][col] = Integer.parseInt(scanConsole.nextLine());

} catch (NumberFormatException e) {

System.out.println("Проверьте корректность ввода данных!");

isCorrect = false;

}

if (isCorrect)

isCorrect = checkArea(matrix[row][col], MIN\_MARK, MAX\_MARK);

} while (!isCorrect);

return matrix;

}

public static int[][] readMatrix() {

int[][] matrix;

int order;

if (chooseFileInput()) {

getFileNormalReading();

matrix = readFileMatrix();

}

else {

matrix = readConsoleMatrix();

}

return matrix;

}

public static int[] findGoodMen(int[][] matrix, int[] goodMen) {

boolean goodMarks;

goodMarks = true;

for (int row = 0; row < LAST\_STUDENT; row++) {

for (int col = 0; col < LAST\_COLUMN; col++) {

if (matrix[row][col] < 9)

goodMarks = false;

}

if (goodMarks)

goodMen[row] = row + 1;

goodMarks = true;

}

return goodMen;

}

public static boolean chooseFileOutput() {

int isFileOutput;

boolean isCorrect, choose;

isFileOutput = 0;

choose = true;

do {

System.out.println("Вы хотите выводить матрицу через файл? (Да - " + 1 + " / Нет - " + 0 + ")");

isCorrect = true;

try {

isFileOutput = Integer.parseInt(scanConsole.nextLine());

} catch (NumberFormatException e) {

System.out.println("Некорректный выбор!");

isCorrect = false;

}

if (isCorrect) {

if (isFileOutput == 1)

choose = true;

else if (isFileOutput == 0)

choose = false;

else {

isCorrect = false;

System.out.println("Некорректный выбор!");

}

}

} while (!isCorrect);

return choose;

}

public static void printConsoleResult(int[] goodMen) {

int row;

row = 0;

System.out.print("Номера отличников:\n");

for (row = 0; row < LAST\_STUDENT; row++)

if (goodMen[row] > 0)

System.out.print(goodMen[row] + " ");

}

public static void printFileResult(int[] goodMen) {

int row;

row = 0;

try {

FileWriter writer = new FileWriter(file, true);

writer.write("Номера отличников:\n");

while (row < LAST\_STUDENT) {

if (goodMen[row] > 0)

writer.write(goodMen[row] + " ");

row++;

}

writer.close();

} catch (IOException e) {}

}

public static void printResult(int[] goodMen) {

if (chooseFileOutput()) {

getFileNormalWriting();

printFileResult(goodMen);

}

else

printConsoleResult(goodMen);

}

public static void main(String[] args) {

int[][] matrix;

int[] goodMen = new int [LAST\_STUDENT];

int arrLen;

int[][] resMatrix;

printTask();

matrix = readMatrix();

findGoodMen(matrix, goodMen);

printResult(goodMen);

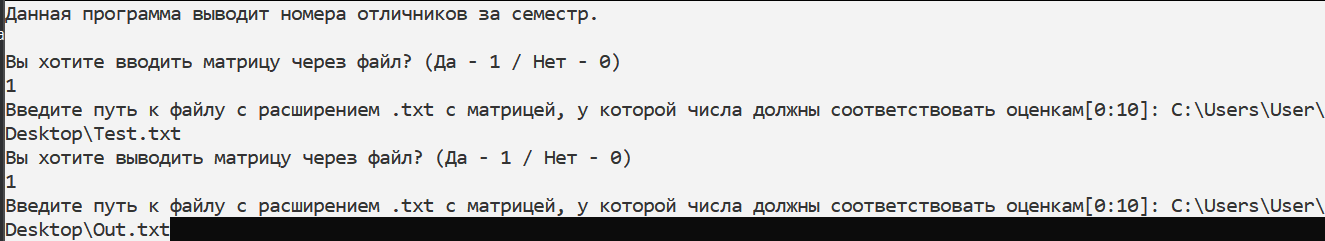
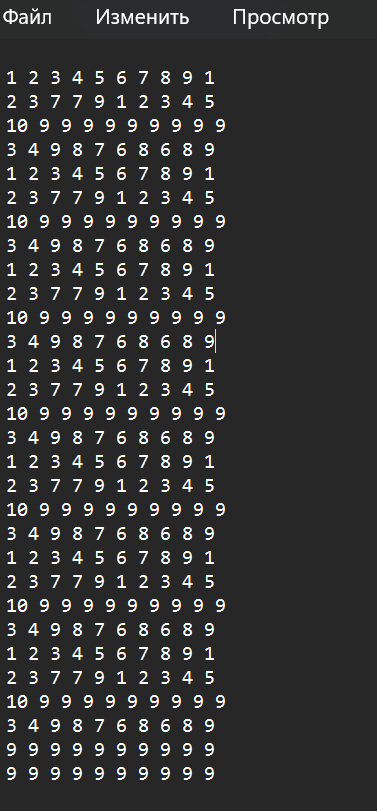
scanConsole.close();

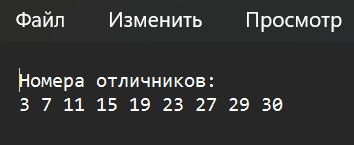
}

}

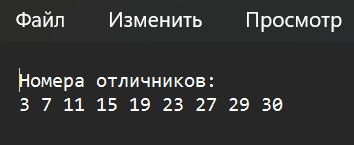
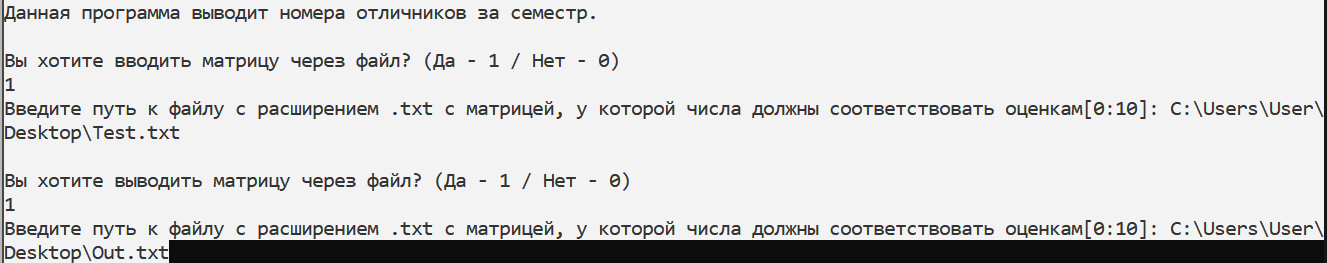
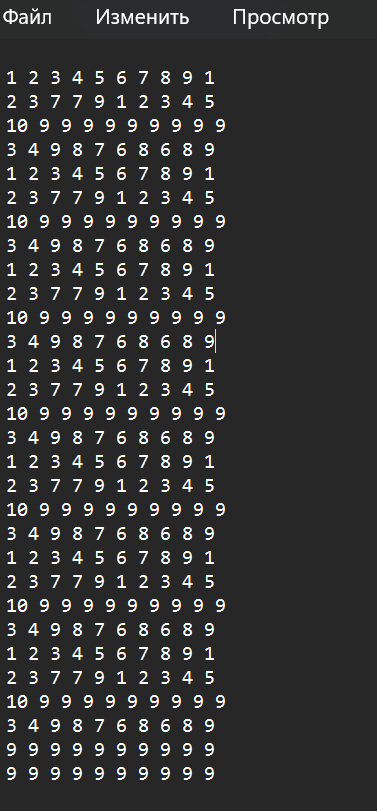
**Скриншоты:**

**Delphi:**

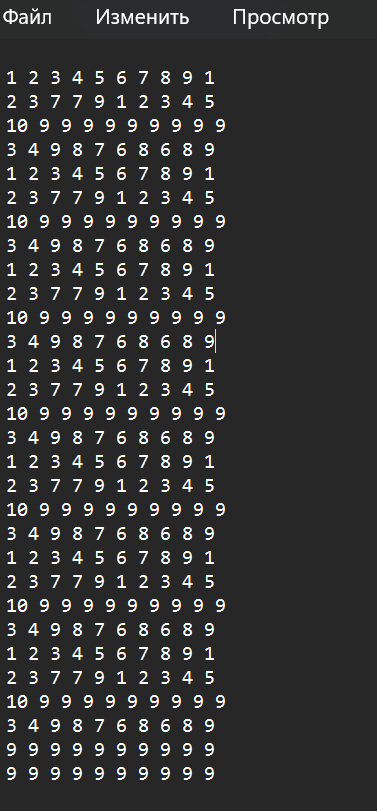
****

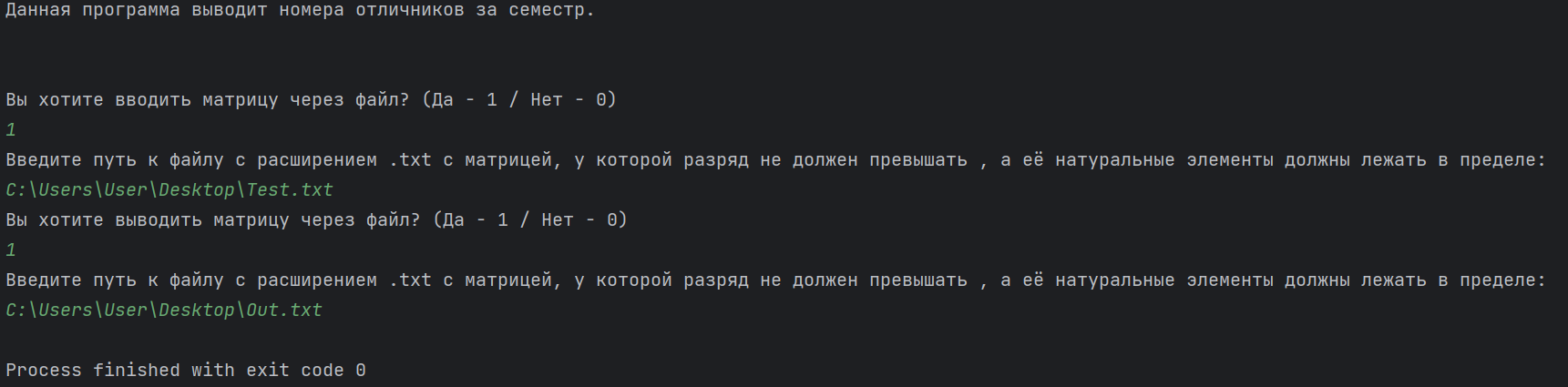
****

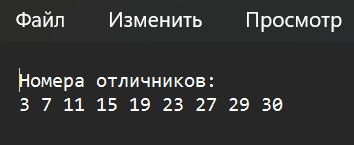
**C++:**

****

**Java:**

****

****

****

**Блок-схема:**









