УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.5

по предмету «Основы алгоритмизации и программирования»

Вариант 5

Выполнил:

Бражалович А. И.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Дана квадратная матрица А порядка n. Найти седловую точку матрицы, т.е. элемент, который является наименьшим в своей строке и наибольшим в своем столбце.

**Код программы Delphi:**

Program Lab2\_5;

Uses

System.SysUtils;

Type

TMatrix = Array Of Array Of Integer;

TArr = Array Of Integer;

Const

MIN\_MATRIX = 2;

MAX\_MATRIX = 100;

MIN\_ELEMENT = -100000;

MAX\_ELEMENT = 100000;

Procedure PrintTask();

Begin

WriteLn('Данная программа находит седловую точку матрицы:', #13#10);

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 порядок не должен превышать ', MAX\_MATRIX, ', а её элементы должны лежать в пределе [', MIN\_ELEMENT, ' : ', MAX\_ELEMENT,']: ');

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

Buf: Char;

Order, Element: Integer;

IsCorrect: Boolean;

Begin

Buf := ' ';

Order := 0;

Element := 0;

IsCorrect := True;

Reset(F);

Try

Read(F, Order);

Except

IsCorrect := False;

End;

ReadLn(F, Buf);

If Buf <> #13 Then

IsCorrect := False;

If IsCorrect Then

IsCorrect := CheckArea(Order, MIN\_MATRIX, MAX\_MATRIX);

While IsCorrect And Not EOF(F) Do

Begin

While IsCorrect And Not EOLN(F) Do

Begin

Try

Read(F, Element);

Except

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckArea(Element, MIN\_ELEMENT, MAX\_ELEMENT);

End;

ReadLn(F);

End;

CloseFile(F);

IsRightFileNums := IsCorrect;

End;

Function IsOrdersEqual(Var F: TextFile) : Boolean;

Var

Order, Rows, Cols, K: Integer;

IsCorrect: Boolean;

Begin

Order := 0;

Rows := 0;

Cols := 0;

K := 0;

IsCorrect := True;

Reset(F);

Readln(F, Order);

While IsCorrect And Not EOF(F) Do

Begin

Cols := 0;

While IsCorrect And Not EOLN(F) Do

Begin

Read(F, K);

Inc(Cols);

End;

If IsCorrect Then

Begin

Readln(F);

Inc(Rows);

If Cols = Order Then

IsCorrect := True;

End;

End;

CloseFile(F);

If IsCorrect Then

Begin

IsCorrect := CheckArea(Cols, MIN\_MATRIX, MAX\_MATRIX);

IsCorrect := CheckArea(Rows, MIN\_MATRIX, MAX\_MATRIX);

End;

If IsCorrect And (Rows <> Order) Then

IsCorrect := False;

IsOrdersEqual := 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;

If IsCorrect And Not IsOrdersEqual(F) Then

Begin

IsCorrect := False;

Writeln('Значения порядков не равны!');

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 ReadFileOrder(Var F: TextFile) : Integer;

Var

Order: Integer;

Begin

Order := 0;

Reset(F);

Read(F, Order);

CloseFile(F);

ReadFileOrder := Order;

End;

Function ReadFileMatrix(Var F: TextFile; Order: Integer) : TMatrix;

Var

Matrix: TMatrix;

Row, Col: Integer;

Begin

SetLength(Matrix, Order, Order);

Reset(F);

Readln(F);

For Row := Low(Matrix) To High(Matrix) Do

Begin

For Col := Low(Matrix[Row]) To High(Matrix[Row]) Do

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

Readln(F);

End;

CloseFile(F);

ReadFileMatrix := Matrix

End;

Function ReadConsoleOrder() : Integer;

Var

Order: Integer;

IsCorrect: Boolean;

Begin

Order := 0;

IsCorrect := False;

Repeat

Write('Введите порядок матрицы [', MIN\_MATRIX, ' : ', MAX\_MATRIX, ']: ');

Try

Readln(Order);

IsCorrect := True;

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckArea(Order, MIN\_MATRIX, MAX\_MATRIX);

Until IsCorrect;

ReadConsoleOrder := Order;

End;

Function ReadConsoleMatrix(Order: Integer) : TMatrix;

Var

Matrix: TMatrix;

Row, Col: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := False;

SetLength(Matrix, Order, Order);

For Row := Low(Matrix) To High(Matrix) Do

For Col := Low(Matrix) To High(Matrix) Do

Repeat

Write('Введите в ', (Row + 1), ' строке ', (Col + 1), ' столбце элемент [', MIN\_ELEMENT, ' : ', MAX\_ELEMENT, ']: ');

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\_ELEMENT, MAX\_ELEMENT);

Until IsCorrect;

ReadConsoleMatrix := Matrix;

End;

Procedure ReadMatrix(Var Matrix: TMatrix; Var Order: Integer);

Var

RF: TextFile;

Begin

If ChooseFileInput() Then

Begin

GetFileNormalReading(RF);

Order := ReadFileOrder(RF);

Matrix := ReadFileMatrix(RF, Order);

End

Else

Begin

Order := ReadConsoleOrder();

Matrix := ReadConsoleMatrix(Order);

End;

End;

Function FindMinInLine(Matrix: TMatrix; Order: Integer): TArr;

Var

MinInLine: TArr;

Line, Column: Integer;

Begin

SetLength(MinInLine, Order);

For Line := Low(Matrix) To High(Matrix) Do

Begin

MinInLine[Line] := Matrix[Line][0];

For Column := Low(Matrix) To High(Matrix) Do

Begin

If Matrix[Line][Column] < MinInLine[Line] Then

MinInLine[line] := Matrix[Line][Column];

End;

End;

FindMinInLine := MinInLine;

End;

Function FindMaxInColumn(Matrix: TMatrix; Order: Integer): TArr;

Var

Line, Column: Integer;

MaxInColumn: TArr;

Begin

Line := 0;

Column := 0;

SetLength(MaxInColumn, Order);

For Column := Low(Matrix) To High(Matrix) Do

Begin

MaxInColumn[Column] := Matrix[0][Column];

For Line := Low(Matrix) To High(Matrix) Do

Begin

If Matrix[Line][Column] > MaxInColumn[Column] Then

MaxInColumn[Column] := Matrix[Line][Column];

End;

End;

FindMaxInColumn := MaxInColumn;

End;

Procedure FreeMemory(Var Matrix: TMatrix; Var MinInLine: TArr; Var MaxInLine: TArr);

Begin

Matrix := Nil;

MinInLine := Nil;

MaxInLine := Nil;

End;

Function ChooseFileOutput() : Boolean;

Var

IsFileOutput: Integer;

IsCorrect, Choose: Boolean;

Begin

IsFileOutput := 0;

IsCorrect := False;

Choose := False;

WriteLn;

Repeat

WriteLn('Вы хотите выводить ответ 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(MinInLine: TArr; MaxInColumn: TArr);

Var

Line, Column: Integer;

SaddlePoint: Boolean;

Begin

Line := 0;

Column := 0;

SaddlePoint := False;

For Line := Low(MaxInColumn) To High(MaxInColumn) Do

Begin

For Column := Low(MaxInColumn) To High(MaxInColumn) Do

Begin

If MinInLine[line] = MaxInColumn[Column] Then

Begin

Writeln('Седловая точка находиться на месте: ', line + 1, ' ', column + 1);

Saddlepoint := True;

End;

End;

End;

If Not SaddlePoint Then

Writeln('Нет седловой точки!');

End;

Procedure PrintFileResult(Var F: TextFile; MinInLine: TArr; MaxInColumn: TArr);

Var

Line, Column: Integer;

SaddlePoint: Boolean;

Begin

SaddlePoint := False;

Line := 0;

Column := 0;

Append(F);

WriteLn(F);

For Line := Low(MaxInColumn) To High(MaxInColumn) Do

Begin

For Column := Low(MaxInColumn) To High(MaxInColumn) Do

Begin

If MinInLine[line] = MaxInColumn[Column] Then

Begin

Writeln(F, 'Седловая точка находиться на месте: ', line + 1, ' ', column + 1);

SaddlePoint := True;

End;

End;

End;

If Not SaddlePoint Then

Writeln(F,'Нет седловой точки!');

CloseFile(F);

End;

Procedure PrintResult(MinInLine, MaxInColumn: TArr);

Var

WF: TextFile;

Begin

If ChooseFileOutput() Then

Begin

GetFileNormalWriting(WF);

PrintFileResult(WF, MinInLine, MaxInColumn);

End

Else

PrintConsoleResult(MinInLine, MaxInColumn);

End;

Var

Matrix: TMatrix;

Order: Integer;

MinInLine, MaxInColumn: TArr;

Begin

Order := 0;

PrintTask();

ReadMatrix(Matrix, Order);

MaxInColumn := FindMaxInColumn(Matrix, Order);

MinInLine := FindMinInLine(Matrix, Order);

PrintResult(MinInLine, MaxInColumn);

FreeMemory(Matrix, MinInLine, MaxInColumn);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <string.h>

#include <fstream>

using namespace std;

const int MIN\_MATRIX = 2;

const int MAX\_MATRIX = 100;

const int MIN\_ELEMENT = -100000;

const int MAX\_ELEMENT = 100000;

void printTask()

{

cout << "Данная программа находит седловую точку матрицы:\n\n";

}

void setLengthMatrix(int\*\*& matrix, int order)

{

matrix = new int\* [order];

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

matrix[i] = new int[order];

}

void setLegnthMinInLine(int\*& minInLine, int order)

{

minInLine = new int [order];

}

void setLegnthMaxInColumn(int\*& maxInColumn, int order)

{

maxInColumn = new int[order];

}

bool chooseFileInput()

{

int isFileInput;

bool isCorrect, 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 ( 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 разряд не должен превышать " << MAX\_MATRIX << ", а её натуральные элементы должны лежать в пределе[" << MIN\_ELEMENT << " : " << MAX\_ELEMENT << "]: ";

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 element;

int order;

bool isCorrect;

element = 0;

order = 0;

isCorrect = true;

ifstream file(pathToFile);

file >> order;

if (file.fail())

{

isCorrect = false;

file.clear();

cout << "Проверьте порядок матрицы!\n";

}

if (isCorrect)

{

if (file.peek() != '\n')

{

isCorrect = false;

cout << "Неправильный порядок матрицы!\n";

}

}

if (isCorrect)

isCorrect = checkArea(order, MIN\_MATRIX, MAX\_MATRIX);

file.get() != '\n';

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

{

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

{

file >> element;

if (file.fail())

{

isCorrect = false;

file.clear();

}

if (isCorrect)

isCorrect = checkArea(element, MIN\_ELEMENT, MAX\_ELEMENT);

}

file.get() != '\n';

}

file.close();

return isCorrect;

}

bool isOrdersEqual(string pathToFile)

{

int order, rows, cols, k;

bool isCorrect;

order = 0;

rows = 0;

cols = 0;

k = 0;

isCorrect = true;

ifstream file(pathToFile);

file >> order;

file.get() != '\n';

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

{

cols = 0;

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

{

file >> k;

cols++;

}

if (isCorrect)

{

file.get() != '\n';

rows++;

if (cols != order)

isCorrect = false;

}

}

file.close();

if (isCorrect)

{

isCorrect = checkArea(cols, MIN\_MATRIX, MAX\_MATRIX);

isCorrect = checkArea(rows, MIN\_MATRIX, MAX\_MATRIX);

}

if (isCorrect && rows != order)

isCorrect = false;

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";

}

if (isCorrect && !isOrdersEqual(pathToFile))

{

isCorrect = false;

cout << "Значения порядков не равны!\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 readFileOrder(string pathToFile)

{

int order;

order = 0;

ifstream file(pathToFile);

file >> order;

file.close();

return order;

}

int\*\* readFileMatrix(string pathToFile, int order)

{

int\*\* matrix;

int row, col;

setLengthMatrix(matrix, order);

ifstream file(pathToFile);

file.get() != '\n';

for (int row = 0; row < order; row++)

{

for (int col = 0; col < order; col++)

file >> matrix[row][col];

file.get() != '\n';

}

file.close();

return matrix;

}

int readConsoleOrder()

{

int order;

bool isCorrect;

order = 0;

do

{

isCorrect = true;

cout << "Введите порядок матрицы [" << MIN\_MATRIX << " : " << MAX\_MATRIX << "]: ";

cin >> order;

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

}

if (isCorrect)

isCorrect = checkArea(order, MIN\_MATRIX, MAX\_MATRIX);

} while (!isCorrect);

return order;

}

int\*\* readConsoleMatrix(int order)

{

int\*\* matrix;

bool isCorrect;

setLengthMatrix(matrix, order);

for (int row = 0; row < order; row++)

for (int col = 0; col < order; col++)

do

{

isCorrect = true;

cout << "Введите в " << (row + 1) << " строке " << (col + 1) << " столбце элемент [" << MIN\_ELEMENT << " : " << MAX\_ELEMENT << "]: ";

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

}

if (isCorrect)

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

} while (!isCorrect);

return matrix;

}

void readMatrix(int\*\*& matrix, int& order)

{

string pathToFile;

pathToFile = "";

if (chooseFileInput())

{

getFileNormalReading(pathToFile);

order = readFileOrder(pathToFile);

matrix = readFileMatrix(pathToFile, order);

}

else

{

order = readConsoleOrder();

matrix = readConsoleMatrix(order);

}

}

int\* findMinInLine(int\*\* matrix, int order)

{

int\* minInLine;

int line;

int column;

line = 0;

column = 0;

setLegnthMinInLine(minInLine, order);

for (line = 0; line < order; line++)

{

minInLine[line] = matrix[line][0];

for (column = 0; column < order; column++)

if (matrix[line][column] < minInLine[line])

minInLine[line] = matrix[line][column];

}

return minInLine;

}

int\* findMaxInColumn(int\*\* matrix, int order)

{

int\* maxInColumn;

int line;

int column;

setLegnthMaxInColumn(maxInColumn, order);

line = 0;

column = 0;

for (column = 0; column < order; column++)

{

maxInColumn[column] = matrix[0][column];

for (line = 0; line < order; line++)

if (matrix[line][column] > maxInColumn[column])

maxInColumn[column] = matrix[line][column];

}

return maxInColumn;

}

void freeMemory(int\*\*& matrix, int order, int\*& minInLine, int\*& maxInColumn)

{

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

delete[] matrix[i];

delete[] matrix;

delete[] minInLine;

delete[] maxInColumn;

}

bool chooseFileOutput()

{

int isFileOutput;

bool isCorrect, 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\* minInLine, int\* maxInColumn, int order)

{

int line;

int column;

bool saddlePoint;

line = 0;

column = 0;

saddlePoint = false;

for (line = 0; line < order; line++)

for (column = 0; column < order; column++)

if (minInLine[line] == maxInColumn[column])

{

cout << "Седловая точка находиться на месте: " << line + 1 << " " << column + 1;

saddlePoint = true;

}

if (!saddlePoint)

cout << "Нет седловой точки!";

}

void printFileResult(string pathToFile, int\* minInLine, int\* maxInColumn, int order)

{

int line;

int column;

bool saddlePoint;

line = 0;

column = 0;

saddlePoint = false;

ofstream file(pathToFile, ios::app);

file << "\n";

for (line = 0; line < order; line++)

for (column = 0; column < order; column++)

if (minInLine[line] == maxInColumn[column])

{

file << "Седловая точка находиться на месте: " << line + 1 << " " << column + 1;

saddlePoint = true;

}

if (!saddlePoint)

file << "Нет седловой точки!";

file.close();

}

void printResult(int\* minInLine, int\* maxInColumn, int order)

{

string pathToFile;

pathToFile = "";

if (chooseFileOutput())

{

getFileNormalWriting(pathToFile);

printFileResult(pathToFile, minInLine, maxInColumn, order);

}

else

printConsoleResult(minInLine, maxInColumn, order);

}

int main()

{

int\*\* matrix;

int order;

int\* minInLine;

int\* maxInColumn;

setlocale(LC\_ALL, "RU");

printTask();

readMatrix(matrix, order);

maxInColumn = findMaxInColumn(matrix, order);

minInLine = findMinInLine(matrix, order);

printResult(minInLine, maxInColumn, order);

freeMemory(matrix, order, minInLine, maxInColumn);

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\_MATRIX = 2;

public static final int MAX\_MATRIX = 100;

public static final int MIN\_ELEMENT = -100000;

public static final int MAX\_ELEMENT = 100000;

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

public static Scanner scanFile;

public static File file;

public static void printTask() {

System.out.println("Данная программа находит седловую точку матрицы:\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!");

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 " + MAX\_MATRIX + ", а её элементы должны лежать в пределе [" + MIN\_ELEMENT + ":" + MAX\_ELEMENT + "]: ");

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 element;

int order;

boolean isCorrect;

isCorrect = true;

element = 0;

order = 0;

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

try {

order = scanFile.nextInt();

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect) {

if (!scanFile.hasNextLine()) {

isCorrect = false;

System.out.println("Неправильный порядок матрицы!");

}

}

if (isCorrect)

isCorrect = checkArea(order, MIN\_MATRIX, MAX\_MATRIX);

scanFile.nextLine();

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

try {

element = scanFile.nextInt();

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

isCorrect = checkArea(element, MIN\_ELEMENT, MAX\_ELEMENT);

}

scanFile.close();

return isCorrect;

}

public static boolean isOrdersEqual() {

int order, rows, cols, k;

boolean isCorrect;

order = 0;

rows = 0;

k = 0;

cols = 0;

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

order = scanFile.nextInt();

scanFile.nextLine();

isCorrect = true;

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

Scanner lineScanFile = new Scanner(scanFile.nextLine());

cols = 0;

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

k = lineScanFile.nextInt();

cols++;

}

if (isCorrect) {

rows++;

if (cols != order)

isCorrect = false;

}

}

scanFile.close();

if (isCorrect) {

isCorrect = checkArea(cols, MIN\_MATRIX, MAX\_MATRIX);

isCorrect = checkArea(rows, MIN\_MATRIX, MAX\_MATRIX);

}

if (isCorrect && rows != order)

isCorrect = false;

return isCorrect;

}

public static void getFileNormalReading() {

boolean isCorrect;

String pathToFile;

int order;

order = 0;

pathToFile = "";

do {

isCorrect = true;

pathToFile = readPathFile();

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!");

}

if (isCorrect && !isOrdersEqual()) {

isCorrect = false;

System.out.println("Значения порядков не равны!");

}

} 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 readFileOrder() {

int order;

order = 0;

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

order = scanFile.nextInt();

scanFile.close();

return order;

}

public static int[][] readFileMatrix(int order) {

int[][] matrix;

matrix = new int[order][order];

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

scanFile.nextLine();

for (int row = 0; row < order; row++)

for (int col = 0; col < order; col++)

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

scanFile.close();

return matrix;

}

public static int readConsoleOrder() {

int order;

boolean isCorrect;

order = 0;

do {

System.out.print("Введите порядок матрицы [" + MIN\_MATRIX + ":" + MAX\_MATRIX + "]: ");

isCorrect = true;

try {

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

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

isCorrect = checkArea(order, MIN\_MATRIX, MAX\_MATRIX);

} while (!isCorrect);

return order;

}

public static int[][] readConsoleMatrix(int order) {

int[][] matrix;

boolean isCorrect;

matrix = new int[order][order];

for (int row = 0; row < order; row++)

for (int col = 0; col < order; col++)

do {

System.out.print("Введите в " + (row + 1) + " строке " + (col + 1) + " столбце элемент[" + MIN\_ELEMENT + ":" + MAX\_ELEMENT + "]: ");

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\_ELEMENT, MAX\_ELEMENT);

} while (!isCorrect);

return matrix;

}

public static int[][] readMatrix() {

int[][] matrix;

int order;

if (chooseFileInput()) {

getFileNormalReading();

order = readFileOrder();

matrix = readFileMatrix(order);

}

else {

order = readConsoleOrder();

matrix = readConsoleMatrix(order);

}

return matrix;

}

public static int[] findMinInLine(int[][] matrix, int order) {

int[] minInLine;

int line;

int column;

line = 0;

column = 0;

minInLine = new int [order];

for (line = 0; line < order; line++) {

minInLine[line] = matrix[line][0];

for (column = 0; column < order; column++)

if (matrix[line][column] < minInLine[line])

minInLine[line] = matrix[line][column];

}

return minInLine;

}

public static int[] findMaxInColumn(int[][] matrix, int order) {

int[] maxInColumn;

int line;

int column;

line = 0;

column = 0;

maxInColumn = new int [order];

for (column = 0; column < order; column++) {

maxInColumn[column] = matrix[0][column];

for (line = 0; line < order; line++)

if (matrix[line][column] > maxInColumn[column])

maxInColumn[column] = matrix[line][column];

}

return maxInColumn;

}

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[] minInLine, int[] maxInColumn, int order) {

int line;

int column;

boolean saddlePoint;

line = 0;

column = 0;

saddlePoint = false;

for (line = 0; line < order; line++)

for (column = 0; column < order; column++)

if (minInLine[line] == maxInColumn[column]) {

System.out.print("Седловая точка находится на месте:" + (line + 1) + " " + (column + 1) + "\n");

saddlePoint = true;

}

if (!saddlePoint)

System.out.print("Нет седловой точки!");

}

public static void printFileResult(int[] minInLine, int[] maxInColumn, int order) {

int line;

int column;

boolean saddlePoint;

line = 0;

column = 0;

saddlePoint = false;

try {

FileWriter writer = new FileWriter(file, true);

for (line = 0; line < order; line++)

for (column = 0; column < order; column++)

if (minInLine[line] == maxInColumn[column])

{

writer.write("Седловая точка находится на месте:" + (line + 1) + " " + (column + 1) + "\n");

saddlePoint = true;

}

if (!saddlePoint)

writer.write("Нет седловой точки!");

writer.close();

} catch (IOException e) {}

}

public static void printResult(int[] minInLine, int[] maxInColumn, int order) {

if (chooseFileOutput()) {

getFileNormalWriting();

printFileResult(minInLine, maxInColumn, order);

}

else

printConsoleResult(minInLine, maxInColumn, order);

}

public static void main(String[] args) {

int[][] matrix;

int order;

int[] minInLine;

int[] maxInColumn;

printTask();

matrix = readMatrix();

order = matrix.length;

maxInColumn = findMaxInColumn(matrix, order);

minInLine = findMinInLine(matrix, order);

printResult(minInLine, maxInColumn, order);

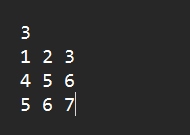
scanConsole.close();

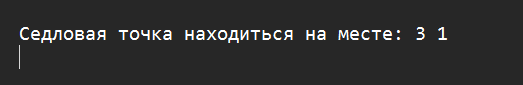
}

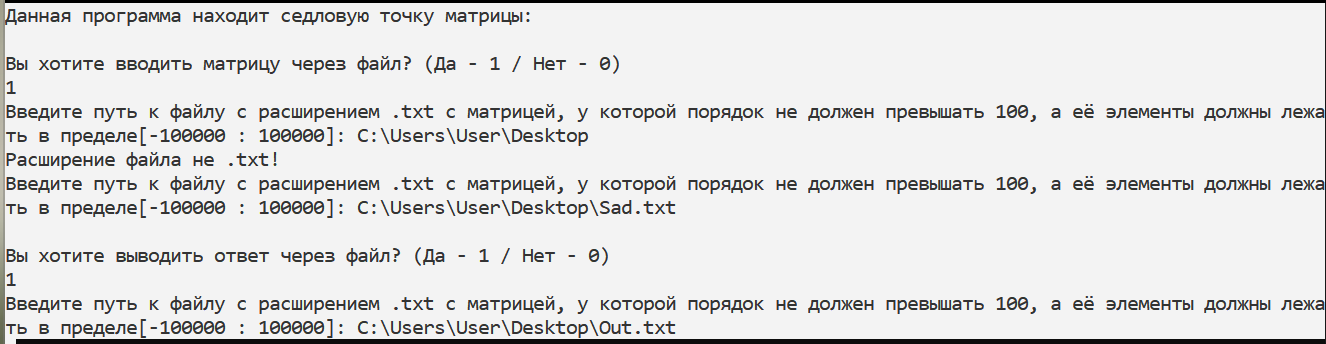
}

**Скриншоты:**

**Delphi:**

****

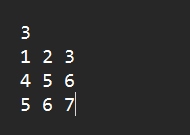
****

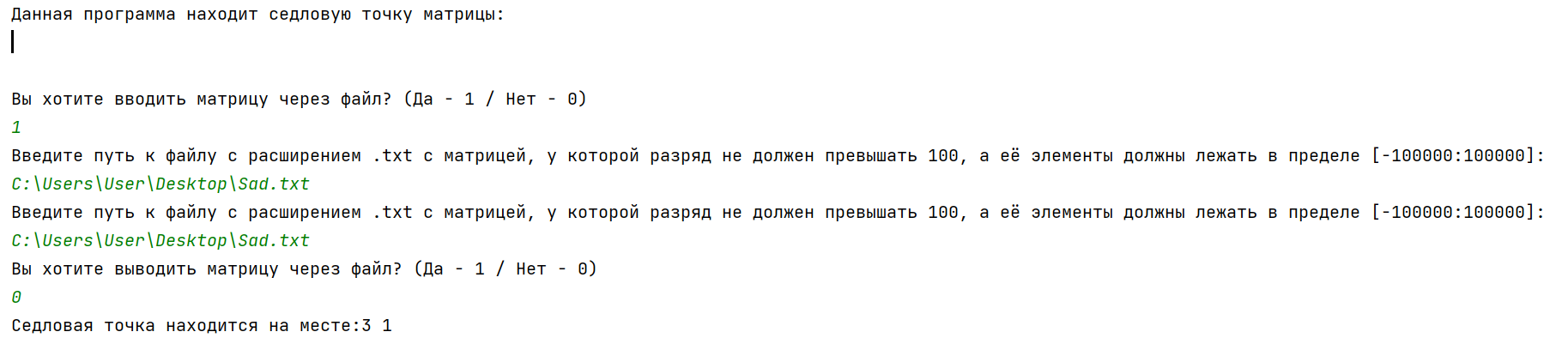
****

**C++:**

****

**Java:**

****

****

**Блок-схема:**

















