УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.4

по предмету «Основы алгоритмизации и программирования»

Вариант 12

Выполнил:

Галуха П. А.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Дан массив А, состоящий из nхn-натуральных чисел. Выбрать элементы, встречающиеся более одного раза.

**Код программы Delphi:**

Program Lab24;

Uses

System.SysUtils;

Type

TMatrix = Array Of Array Of Integer;

TArr = Array Of Integer;

Const

MIN\_O = 1;

MAX\_O = 10;

MIN\_MAT = 1;

MAX\_MAT = 1000000;

YES = 1;

NO = 2;

Procedure PrintTask();

Begin

WriteLn('Данная программа подсчитывает число повторяющихся элементов в матрице.');

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 - ', YES, ' / Нет - ', NO, ')');

Try

ReadLn(IsFileInput);

IsCorrect := True;

Except

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

If IsCorrect Then

Begin

If IsFileInput = YES Then

Choose := True

Else If IsFileInput = NO 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 разряд не должен превышать

', MAX\_O, ', а её натуральные элементы должны лежать в пределе[', MIN\_MAT, ': ',

MAX\_MAT,']: ');

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;

K: Integer;

IsCorrect: Boolean;

Begin

Buf := ' ';

K := 0;

IsCorrect := True;

Reset(F);

Try

Read(F, K);

Except

IsCorrect := False;

End;

ReadLn(F, Buf);

If Buf <> #13 Then

IsCorrect := False;

If IsCorrect Then

IsCorrect := CheckArea(K, MIN\_O, MAX\_O);

While IsCorrect And Not EOF(F) Do

Begin

While IsCorrect And Not EOLN(F) Do

Begin

Try

Read(F, K);

Except

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckArea(K, MIN\_MAT, MAX\_MAT);

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);

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

End;

If IsCorrect Then

Begin

Readln(F);

Inc(Rows);

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

If Cols <> Order Then

IsCorrect := False;

End;

End;

CloseFile(F);

If IsCorrect And (Not 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 Order[', MIN\_O, '; ', MAX\_O, ']: ');

Try

Readln(Order);

IsCorrect := True;

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If IsCorrect Then

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

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 натуральный элемент[',

MIN\_MAT, '; ', MAX\_MAT, ']: ');

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\_MAT, MAX\_MAT);

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;

Procedure FillArr(Matrix: TMatrix; Order: Integer; Var Arr: TArr; Var ArrLen: Integer);

Var

I, Row, Col: Integer;

Begin

I := 0;

ArrLen := Order \* Order;

SetLength(Arr, ArrLen);

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

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

Begin

Arr[I] := Matrix[Row][Col];

Inc(I);

End;

End;

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

Var

I, J, PreviousMaxIndex, Buf: Integer;

Begin

PreviousMaxIndex := ArrLen - 2;

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

For J := Low(Arr) To PreviousMaxIndex - I Do

If Arr[J] > Arr[J + 1] Then

Begin

Buf := Arr[J];

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

Arr[J + 1] := Buf;

End;

End;

Procedure FindSame(Arr: TArr; ArrLen: Integer; Var ResMatrix: TMatrix);

Var

Count, I, J, PreviousMaxIndex: Integer;

Begin

Count := 1;

J := 0;

PreviousMaxIndex := ArrLen - 2;

SetLength(ResMatrix, ArrLen, ArrLen);

For I := Low(Arr) To PreviousMaxIndex Do

If Arr[I] = Arr[I + 1] Then

Inc(Count)

Else

Begin

ResMatrix[J][0] := Arr[I];

ResMatrix[J][1] := Count;

Inc(J);

Count:= 1;

End;

ResMatrix[J][0] := Arr[ArrLen - 1];

ResMatrix[J][1] := Count;

End;

Procedure FreeMemory(Var Matrix: TMatrix; Var Arr: TArr; Var ResMatrix: TMatrix);

Begin

Matrix := Nil;

Arr := Nil;

ResMatrix := Nil;

End;

Function ChooseFileOutput() : Boolean;

Var

IsFileOutput: Integer;

IsCorrect, Choose: Boolean;

Begin

IsFileOutput := 0;

IsCorrect := False;

Choose := False;

WriteLn;

Repeat

WriteLn('Вы хотите выводить матрицу через файл? (Да - ', YES, ' / Нет - ', NO, ')');

Try

ReadLn(IsFileOutput);

IsCorrect := True;

Except

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

If IsCorrect Then

Begin

If IsFileOutput = YES Then

Choose := True

Else If IsFileOutput = NO Then

Choose := False

Else

Begin

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

End;

Until IsCorrect;

ChooseFileOutput := Choose;

End;

Procedure PrintConsoleResult(ResMatrix: TMatrix; ArrLen: Integer);

Var

I: Integer;

Begin

I := 0;

WriteLn;

While (I < ArrLen) And (ResMatrix[I][1] <> 0) Do

Begin

WriteLn('Число ', ResMatrix[I][0], ' встречается ', ResMatrix[I][1], ' раз(а).');

Inc(I);

End;

End;

Procedure PrintFileResult(Var F: TextFile; ResMatrix: TMatrix; ArrLen: Integer);

Var

I: Integer;

Begin

Append(F);

I := 0;

WriteLn(F);

While (I < ArrLen) And (ResMatrix[I][1] <> 0) Do

Begin

WriteLn(F, 'Число ', ResMatrix[I][0], ' встречается ', ResMatrix[I][1], ' раз(а).');

Inc(I);

End;

CloseFile(F);

End;

Procedure PrintResult(ResMatrix: TMatrix; ArrLen: Integer);

Var

WF: TextFile;

Begin

If ChooseFileOutput() Then

Begin

GetFileNormalWriting(WF);

PrintFileResult(WF, ResMatrix, ArrLen);

End

Else

PrintConsoleResult(ResMatrix, ArrLen);

End;

Var

Matrix: TMatrix;

Order: Integer;

Arr: TArr;

ArrLen: Integer;

ResMatrix: TMatrix;

Begin

Order := 0;

ArrLen := 0;

PrintTask();

ReadMatrix(Matrix, Order);

FillArr(Matrix, Order, Arr, ArrLen);

SortArr(Arr, ArrLen);

FindSame(Arr, ArrLen, ResMatrix);

PrintResult(ResMatrix, ArrLen);

FreeMemory(Matrix, Arr, ResMatrix);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <string.h>

#include <fstream>

const int

MIN\_O = 1,

MAX\_O = 10,

MIN\_MAT = 1,

MAX\_MAT = 1000000,

YES = 1,

NO = 2;

void printTask()

{

std::cout << "Данная программа подсчитывает число повторяющихся элементов в матрице.\n\n";

}

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

{

matrix = new int\* [order];

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

matrix[i] = new int[order];

}

bool chooseFileInput()

{

int isFileInput;

bool isCorrect, choose;

isFileInput = 0;

choose = false;

do {

isCorrect = true;

std::cout << "Вы хотите вводить матрицу через файл? (Да - " << YES << " / Нет - " << NO << ")\n";

std::cin >> isFileInput;

if (std::cin.fail())

{

isCorrect = false;

std::cout << "Некорректный выбор!\n";

std::cin.clear();

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

}

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

{

isCorrect = false;

std::cout << "Некорректный выбор!\n";

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

}

if (isCorrect)

{

if (isFileInput == YES)

choose = true;

else if (isFileInput == NO)

choose = false;

else

{

isCorrect = false;

std::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)

{

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

isCorrect = false;

}

return isCorrect;

}

std::string readPathFile()

{

std::string pathToFile;

bool isCorrect;

int len;

pathToFile = "";

isCorrect = false;

len = 0;

do

{

std::cout << "Введите путь к файлу с расширением .txt с матрицей, у которой разряд не должен

превышать " << MAX\_O << ", а её натуральные элементы должны лежать в пределе[" <<

MIN\_MAT << ":" << MAX\_MAT << "]: ";

std::cin >> pathToFile;

len = pathToFile.length();

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

isCorrect = true;

else

{

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

isCorrect = false;

}

} while (!isCorrect);

return pathToFile;

}

bool isExists(std::string pathToFile)

{

bool isCorrect;

isCorrect = false;

std::ifstream file(pathToFile);

if (file.good())

isCorrect = true;

file.close();

return isCorrect;

}

bool isAbleToReading(std::string pathToFile)

{

bool isCorrect;

isCorrect = false;

std::ifstream file(pathToFile);

if (file.is\_open())

isCorrect = true;

file.close();

return isCorrect;

}

bool isAbleToWriting(std::string pathToFile)

{

bool isCorrect;

isCorrect = false;

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

if (file.is\_open())

isCorrect = true;

file.close();

return isCorrect;

}

bool isEmpty(std::string pathToFile)

{

bool isCorrect;

isCorrect = false;

std::ifstream file(pathToFile);

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

isCorrect = true;

file.close();

return isCorrect;

}

bool isRightFileNums(std::string pathToFile)

{

int k;

bool isCorrect;

k = 0;

isCorrect = true;

std::ifstream file(pathToFile);

file >> k;

if (file.fail())

{

isCorrect = false;

file.clear();

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

}

if (isCorrect)

{

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

{

isCorrect = false;

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

}

}

if (isCorrect)

isCorrect = checkArea(k, MIN\_O, MAX\_O);

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

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

{

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

{

file >> k;

if (file.fail())

{

isCorrect = false;

file.clear();

}

if (isCorrect)

isCorrect = checkArea(k, MIN\_MAT, MAX\_MAT);

}

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

}

file.close();

return isCorrect;

}

bool isOrdersEqual(std::string pathToFile)

{

int order, rows, cols, k;

bool isCorrect;

order = 0;

rows = 0;

cols = 0;

k = 0;

isCorrect = true;

std::ifstream file(pathToFile);

file >> order;

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

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

{

cols = 0;

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

{

file >> k;

cols++;

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

}

if (isCorrect)

{

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

rows++;

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

if (cols != order)

isCorrect = false;

}

}

file.close();

if (isCorrect && rows != order)

isCorrect = false;

return isCorrect;

}

void getFileNormalReading(std::string& pathToFile)

{

bool isCorrect;

do

{

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile))

{

isCorrect = false;

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

}

if (isCorrect && !isAbleToReading(pathToFile))

{

isCorrect = false;

std::cout << "Файл закрыт для чтения!\n";

}

if (isCorrect && !isAbleToWriting(pathToFile))

{

isCorrect = false;

std::cout << "Файл закрыт для записи!\n";

}

if (isCorrect && isEmpty(pathToFile))

{

isCorrect = false;

std::cout << "Файл пуст!\n";

}

if (isCorrect && !isRightFileNums(pathToFile))

{

isCorrect = false;

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

}

if (isCorrect && !isOrdersEqual(pathToFile))

{

isCorrect = false;

std::cout << "Значения порядков не равны!\n";

}

} while (!isCorrect);

}

void getFileNormalWriting(std::string& pathToFile)

{

bool isCorrect;

do

{

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile))

{

isCorrect = false;

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

}

if (isCorrect && !isAbleToWriting(pathToFile))

{

isCorrect = false;

std::cout << "Файл закрыт для записи!\n";

}

} while (!isCorrect);

}

int readFileOrder(std::string pathToFile)

{

int order;

order = 0;

std::ifstream file(pathToFile);

file >> order;

file.close();

return order;

}

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

{

int\*\* matrix;

int row, col;

setLengthMatrix(matrix, order);

std::ifstream file(pathToFile);

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

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

{

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

file >> matrix[row][col];

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

}

file.close();

return matrix;

}

int readConsoleOrder()

{

int order;

bool isCorrect;

order = 0;

do

{

isCorrect = true;

std::cout << "Введите порядок матрицы order[" << MIN\_O << "; " << MAX\_O << "]: ";

std::cin >> order;

if (std::cin.fail())

{

isCorrect = false;

std::cin.clear();

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

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

}

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

{

isCorrect = false;

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

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

}

if (isCorrect)

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

} 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;

std::cout << "Введите в " << (row + 1) << " строке " << (col + 1) << " столбце натуральный

элемент[" << MIN\_MAT << "; " << MAX\_MAT << "]: ";

std::cin >> matrix[row][col];

if (std::cin.fail())

{

isCorrect = false;

std::cin.clear();

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

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

}

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

{

isCorrect = false;

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

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

}

if (isCorrect)

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

} while (!isCorrect);

return matrix;

}

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

{

std::string pathToFile;

pathToFile = "";

if (chooseFileInput())

{

getFileNormalReading(pathToFile);

order = readFileOrder(pathToFile);

matrix = readFileMatrix(pathToFile, order);

}

else

{

order = readConsoleOrder();

matrix = readConsoleMatrix(order);

}

}

void fillArr(int\*\* matrix, int order, int\*& arr, int& arrLen)

{

int i;

i = 0;

arrLen = order \* order;

arr = new int[arrLen];

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

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

{

arr[i] = matrix[row][col];

i++;

}

}

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

{

int maxIndex, buf;

maxIndex = arrLen - 1;

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

for (int j = 0; j < maxIndex - i; j++)

if (arr[j] > arr[j + 1])

{

buf = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = buf;

}

}

void findSame(int\* arr, int arrLen, int\*\*& resMatrix)

{

int j, count, maxIndex;

count = 1;

j = 0;

maxIndex = arrLen - 1;

setLengthMatrix(resMatrix, arrLen);

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

if (arr[i] == arr[i + 1])

count++;

else

{

resMatrix[j][0] = arr[i];

resMatrix[j][1] = count;

j++;

count = 1;

}

resMatrix[j][0] = arr[arrLen - 1];

resMatrix[j][1] = count;

resMatrix[j + 1][1] = 0;

}

void freeMemory(int\*\*& matrix, int order, int\*& arr, int\*\*& resMatrix, int arrLen)

{

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

delete[] matrix[i];

delete[] matrix;

delete[] arr;

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

delete[] resMatrix[i];

delete[] resMatrix;

}

bool chooseFileOutput()

{

int isFileOutput;

bool isCorrect, choose;

isFileOutput = 0;

choose = false;

std::cout << "\n";

do {

isCorrect = true;

std::cout << "Вы хотите выводить матрицу через файл? (Да - " << YES << " / Нет - " << NO << ")\n";

std::cin >> isFileOutput;

if (std::cin.fail())

{

isCorrect = false;

std::cout << "Некорректный выбор!\n";

std::cin.clear();

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

}

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

{

isCorrect = false;

std::cout << "Некорректный выбор!\n";

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

}

if (isCorrect)

{

if (isFileOutput == YES)

choose = true;

else if (isFileOutput == NO)

choose = false;

else

{

isCorrect = false;

std::cout << "Некорректный выбор!\n";

}

}

} while (!isCorrect);

return choose;

}

void printConsoleResult(int\*\* resMatrix, int arrLen)

{

int i;

i = 0;

while (i < arrLen && resMatrix[i][1] != 0)

{

std::cout << "\nЧисло " << resMatrix[i][0] << " встречается " << resMatrix[i][1] << " раз(а).";

i++;

}

}

void printFileResult(std::string pathToFile, int\*\* resMatrix, int arrLen)

{

int i;

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

i = 0;

file << "\n";

while (i < arrLen && resMatrix[i][1] != 0)

{

file << "\nЧисло " << resMatrix[i][0] << " встречается " << resMatrix[i][1] << " раз(а).";

i++;

}

file.close();

}

void printResult(int\*\* resMatrix, int arrLen)

{

std::string pathToFile;

pathToFile = "";

if (chooseFileOutput())

{

getFileNormalWriting(pathToFile);

printFileResult(pathToFile, resMatrix, arrLen);

}

else

printConsoleResult(resMatrix, arrLen);

}

int main()

{

int\*\* matrix;

int order;

int\* arr;

int arrLen;

int\*\* resMatrix;

setlocale(LC\_ALL, "RU");

printTask();

readMatrix(matrix, order);

fillArr(matrix, order, arr, arrLen);

sortArr(arr, arrLen);

findSame(arr, arrLen, resMatrix);

printResult(resMatrix, arrLen);

freeMemory(matrix, order, arr, resMatrix, arrLen);

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\_O = 1,  
 MAX\_O = 10,  
 MIN\_MAT = -1000000,  
 MAX\_MAT = 1000000,  
 YES = 1,  
 NO = 2;  
 static Scanner scanConsole = new Scanner(System.in);  
 static Scanner scanFile;  
 static File file;  
 public static void printTask() {  
 System.out.println("Данная программа подсчитывает число повторяющихся элементов в матрице.\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 - " + YES + " / Нет - " +

NO + ")");  
 isCorrect = true;  
 try {  
 isFileInput = Integer.parseInt(scanConsole.nextLine());  
 } catch (NumberFormatException e) {  
 System.out.println("Некорректный выбор!");  
 isCorrect = false;  
 }  
 if (isCorrect) {  
 if (isFileInput == YES)  
 choose = true;  
 else if (isFileInput == NO)  
 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\_O + ", а её натуральные элементы должны лежать в

пределе[" + MIN\_MAT + ":" + MAX\_MAT + "]: ");  
 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 k;  
 boolean isCorrect;  
 isCorrect = true;  
 k = 0;  
 try {scanFile = new Scanner(file);} catch (FileNotFoundException e) {}  
 try {  
 k = 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(k, MIN\_O, MAX\_O);  
 scanFile.nextLine();  
 while (isCorrect && scanFile.hasNextInt()) {  
 try {  
 k = scanFile.nextInt();  
 } catch (NumberFormatException e) {  
 System.out.println("Некорректный тип данных внутри файла!");  
 isCorrect = false;  
 }  
 if (isCorrect)  
 isCorrect = checkArea(k, MIN\_MAT, MAX\_MAT);  
 }  
 scanFile.close();  
 return isCorrect;  
 }  
 public static boolean isOrdersEqual() {  
 int order, rows, cols, k;  
 boolean isCorrect;  
 order = 0;  
 rows = 0;  
 k = 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++;  
 isCorrect = checkArea(cols, MIN\_O, MAX\_O);  
 }  
 if (isCorrect) {  
 rows++;  
 isCorrect = checkArea(rows, MIN\_O, MAX\_O);  
 if (cols != order)  
 isCorrect = false;  
 }  
 }  
 scanFile.close();  
 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 N[" + MIN\_O + "; " + MAX\_O + "]: ");  
 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\_O, MAX\_O);  
 } 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

натуральный элемент[" + MIN\_MAT + "; " + MAX\_MAT + "]: ");  
 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\_MAT, MAX\_MAT);  
 } 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[] fillArr(int[][] matrix, int order) {  
 int[] arr;  
 int arrLen;  
 int i;  
 i = 0;  
 arrLen = order \* order;  
 arr = new int[arrLen];  
 for (int row = 0; row < order; row++)  
 for (int col = 0; col < order; col++)  
 {  
 arr[i] = matrix[row][col];  
 i++;  
 }  
 return arr;  
 }  
 public static void sortArr(int[] arr, int arrLen) {  
 int maxIndex, buf;  
 maxIndex = arrLen - 1;  
 for (int i = 0; i < arrLen; i++)  
 for (int j = 0; j < maxIndex - i; j++)  
 if (arr[j] > arr[j + 1]) {  
 buf = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = buf;  
 }  
 }  
 public static int[][] findSame(int[] arr, int arrLen) {  
 int[][] resMatrix;  
 int j, count, maxIndex;  
 count = 1;  
 j = 0;  
 maxIndex = arrLen - 1;  
 resMatrix = new int[arrLen][arrLen];  
 for (int i = 0; i < maxIndex; i++)  
 if (arr[i] == arr[i + 1])  
 count++;  
 else {  
 resMatrix[j][0] = arr[i];  
 resMatrix[j][1] = count;  
 j++;  
 count = 1;  
 }  
 resMatrix[j][0] = arr[arrLen - 1];  
 resMatrix[j][1] = count;  
 resMatrix[j + 1][1] = 0;  
 return resMatrix;  
 }  
 public static boolean chooseFileOutput() {  
 int isFileOutput;  
 boolean isCorrect, choose;  
 isFileOutput = 0;  
 choose = true;  
 do {  
 System.out.println("Вы хотите выводить матрицу через файл? (Да - " + YES + " / Нет - " +

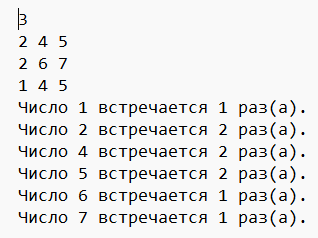
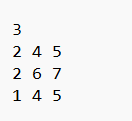
NO + ")");  
 isCorrect = true;  
 try {  
 isFileOutput = Integer.parseInt(scanConsole.nextLine());  
 } catch (NumberFormatException e) {  
 System.out.println("Некорректный выбор!");  
 isCorrect = false;  
 }  
 if (isCorrect) {  
 if (isFileOutput == YES)  
 choose = true;  
 else if (isFileOutput == NO)  
 choose = false;  
 else {  
 isCorrect = false;  
 System.out.println("Некорректный выбор!");  
 }  
 }  
 } while (!isCorrect);  
 return choose;  
 }  
 public static void printConsoleResult(int[][] resMatrix, int arrLen) {  
 int i;  
 i = 0;  
 while (i < arrLen && resMatrix[i][1] != 0)  
 {  
 System.out.print("\nЧисло " + resMatrix[i][0] + " встречается " + resMatrix[i][1] + "

раз(а).");  
 i++;  
 }  
 }  
 public static void printFileResult(int[][] resMatrix, int arrLen) {  
 int i;  
 i = 0;  
 try {  
 FileWriter writer = new FileWriter(file, true);  
 while (i < arrLen && resMatrix[i][1] != 0) {  
 writer.write("\nЧисло " + resMatrix[i][0] + " встречается " + resMatrix[i][1] + "

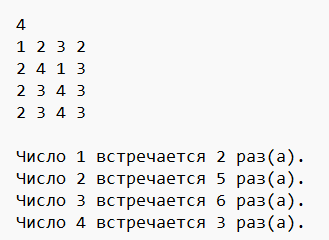
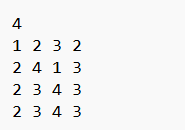
раз(а).");  
 i++;  
 }  
 writer.close();  
 } catch (IOException e) {}  
 }  
 public static void printResult(int[][] resMatrix, int arrLen) {  
 if (chooseFileOutput()) {  
 getFileNormalWriting();  
 printFileResult(resMatrix, arrLen);  
 }  
 else  
 printConsoleResult(resMatrix, arrLen);  
 }  
 public static void main(String[] args) {  
 int[][] matrix;  
 int order;  
 int[] arr;  
 int arrLen;  
 int[][] resMatrix;  
 printTask();  
 matrix = readMatrix();  
 order = matrix.length;  
 arr = fillArr(matrix, order);  
 arrLen = arr.length;  
 sortArr(arr, arrLen);  
 resMatrix = findSame(arr, arrLen);  
 printResult(resMatrix, arrLen);  
 scanConsole.close();  
 }  
}

**Скриншоты:**

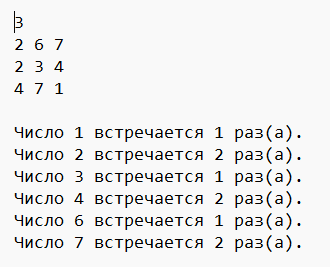
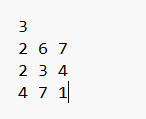
**Delphi:**



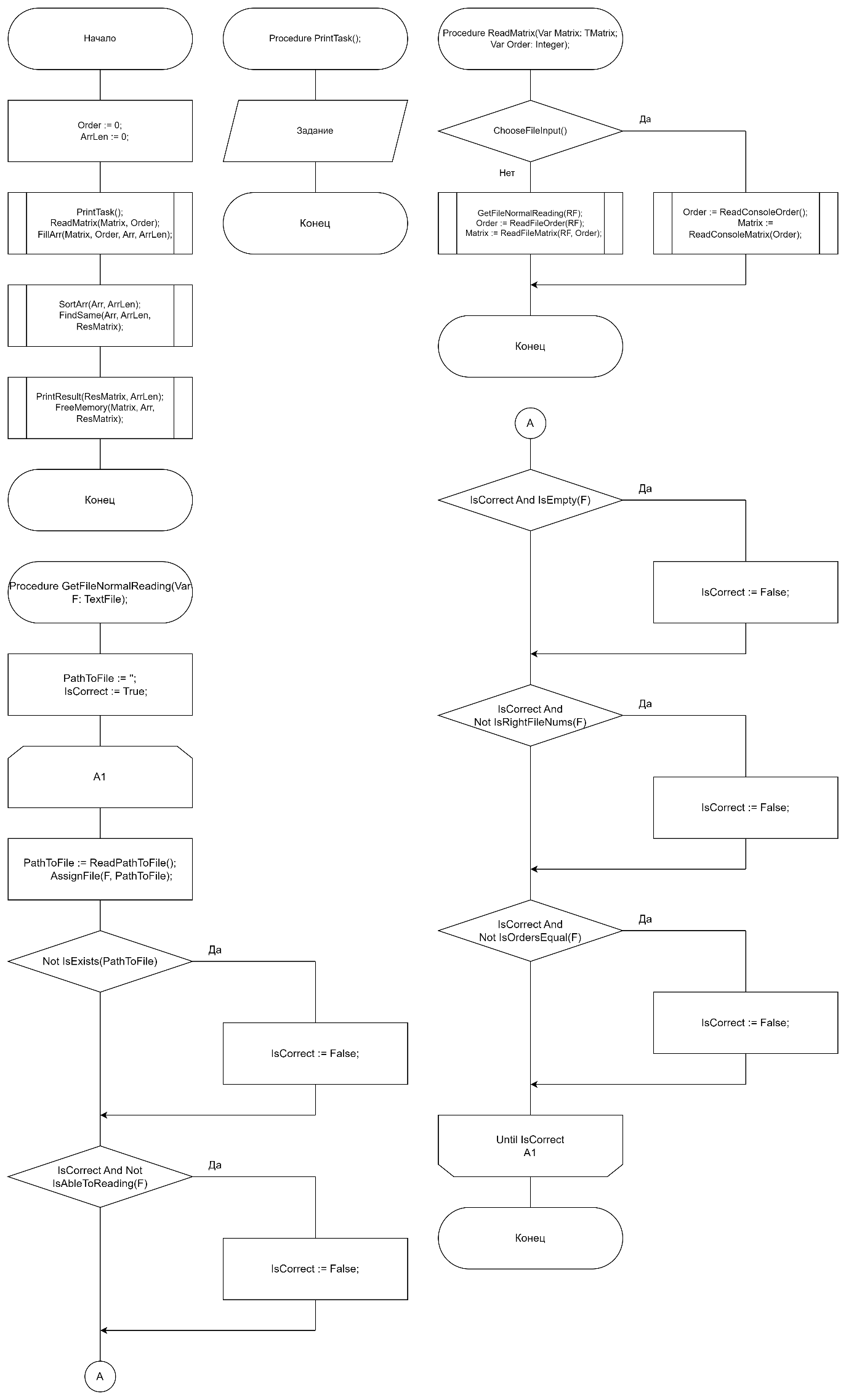
**C++:**

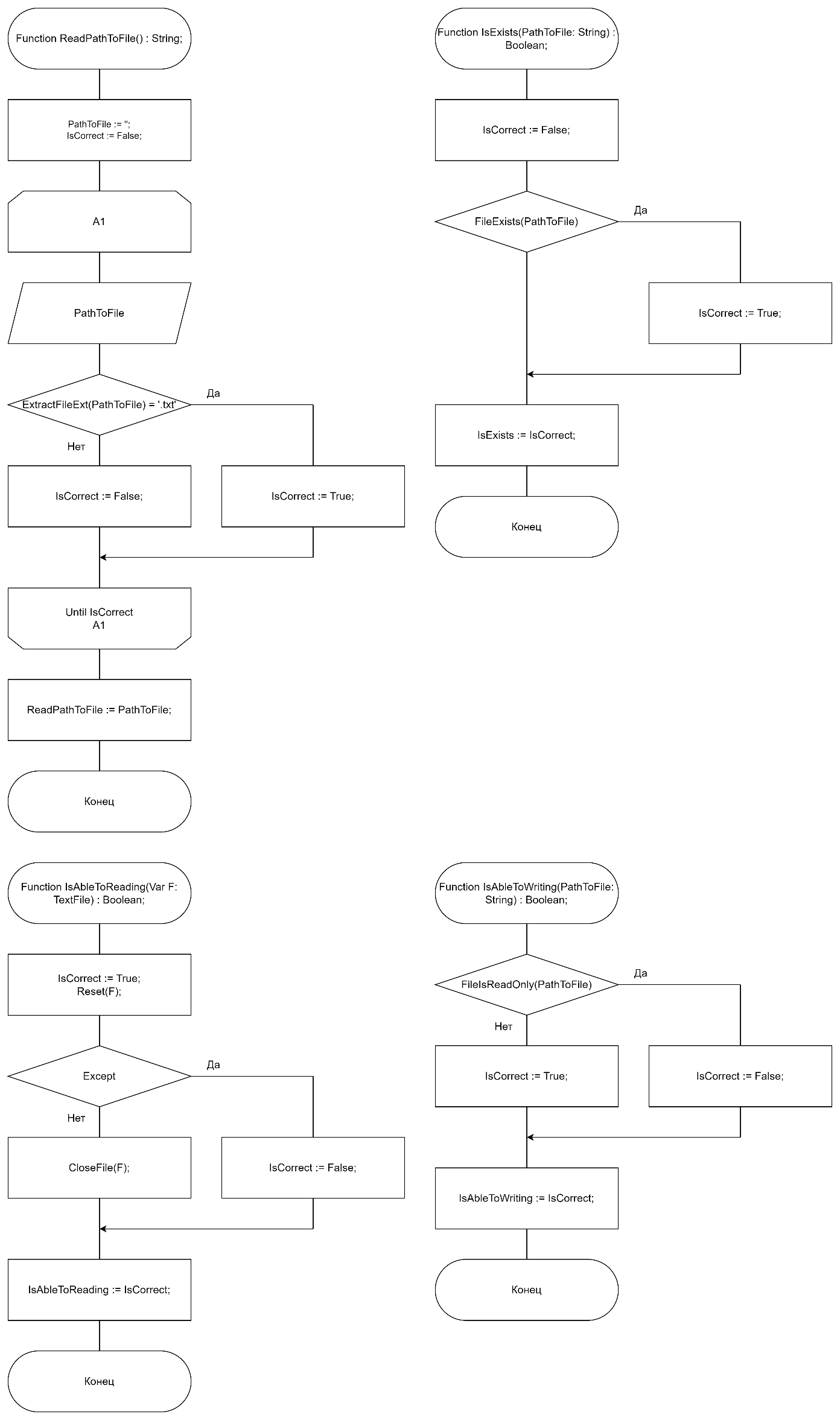


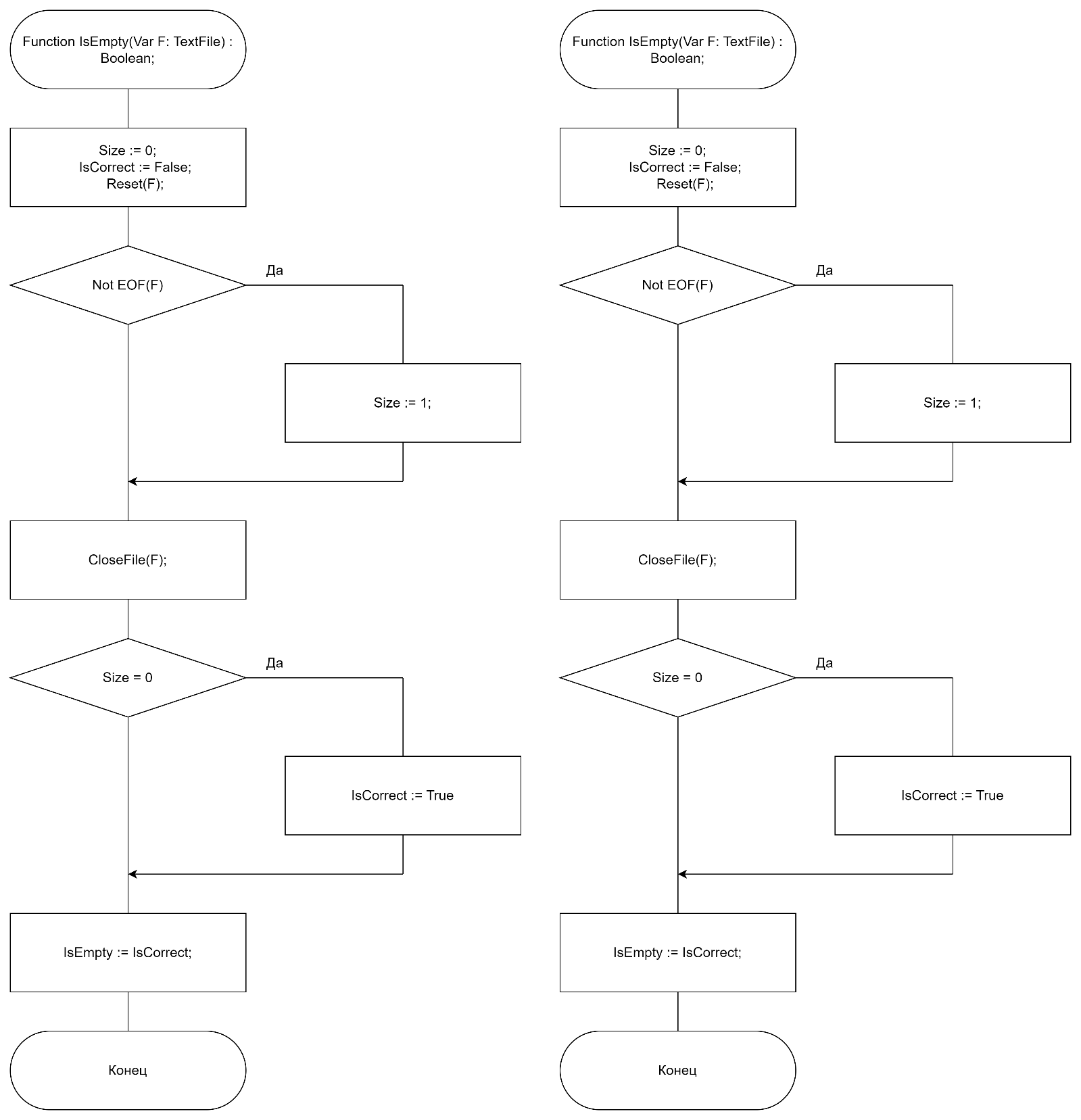
**Java:**

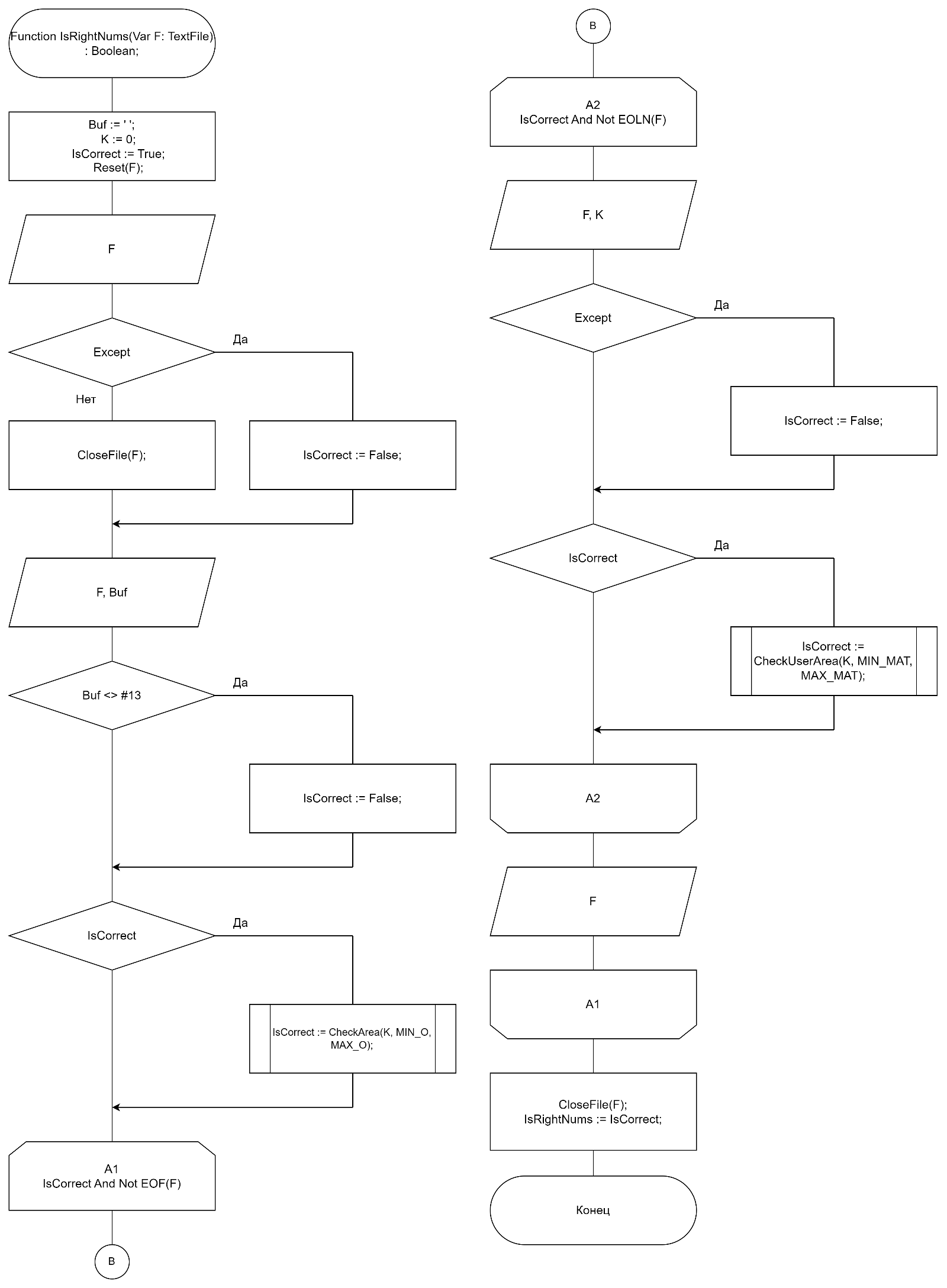


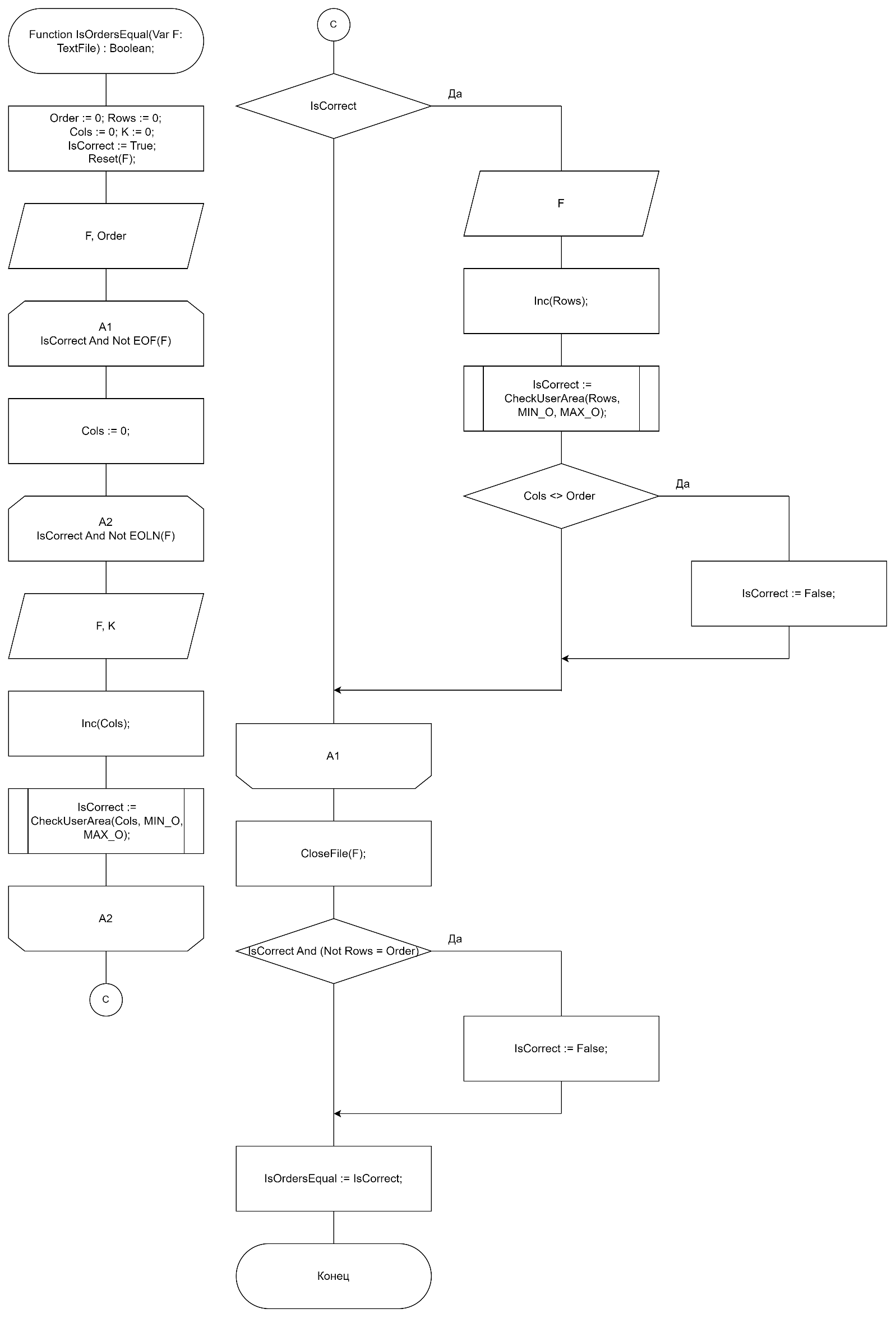
**Блок-схема:**

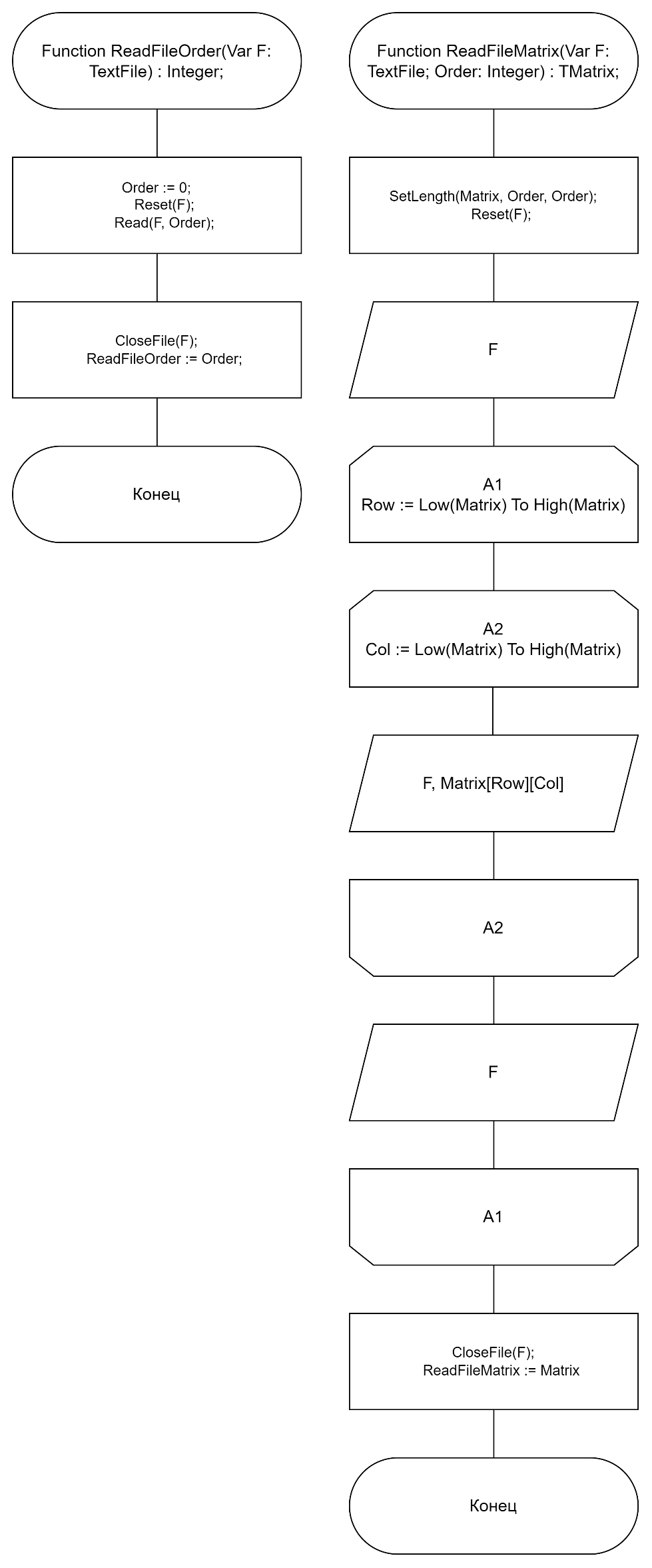
****

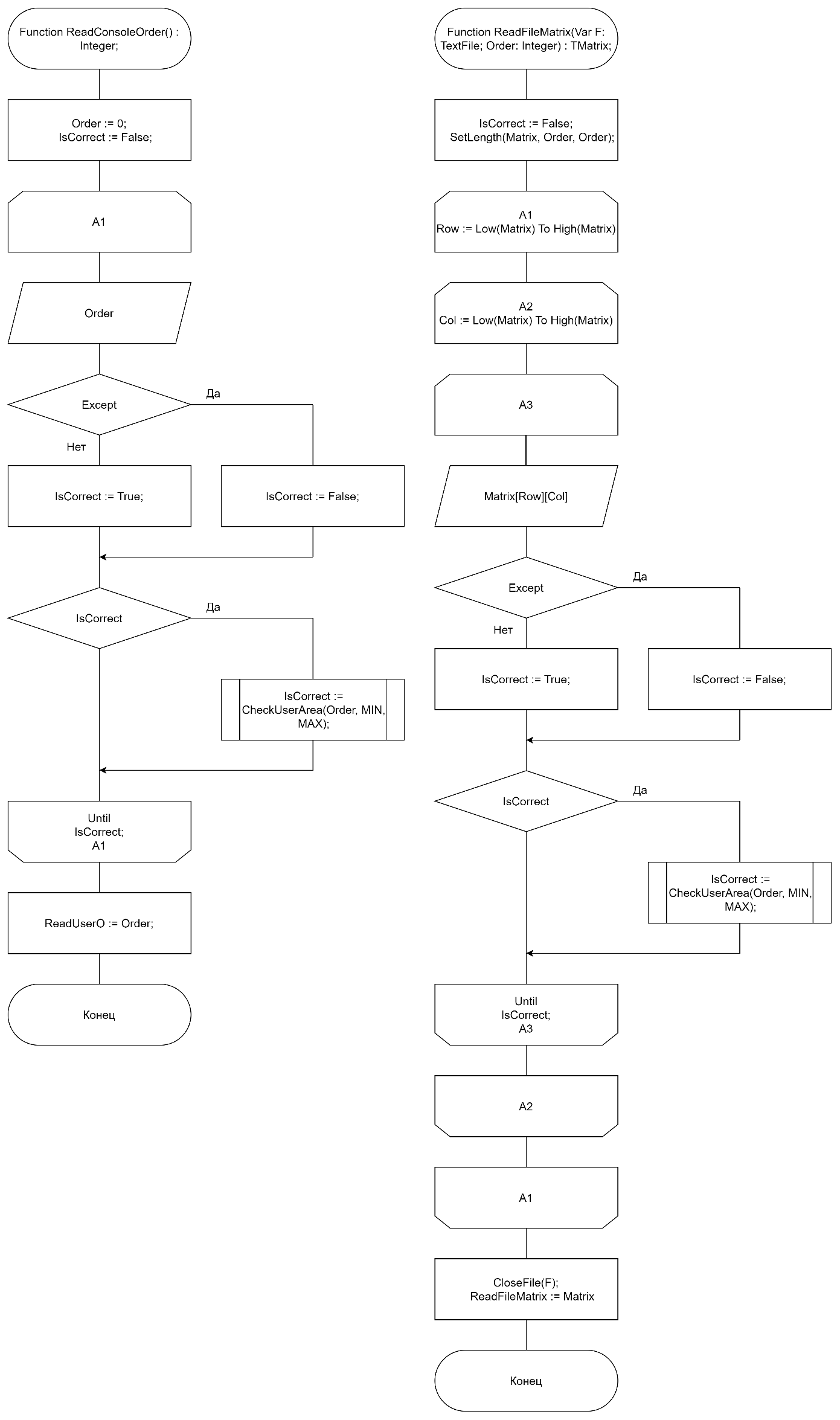
****

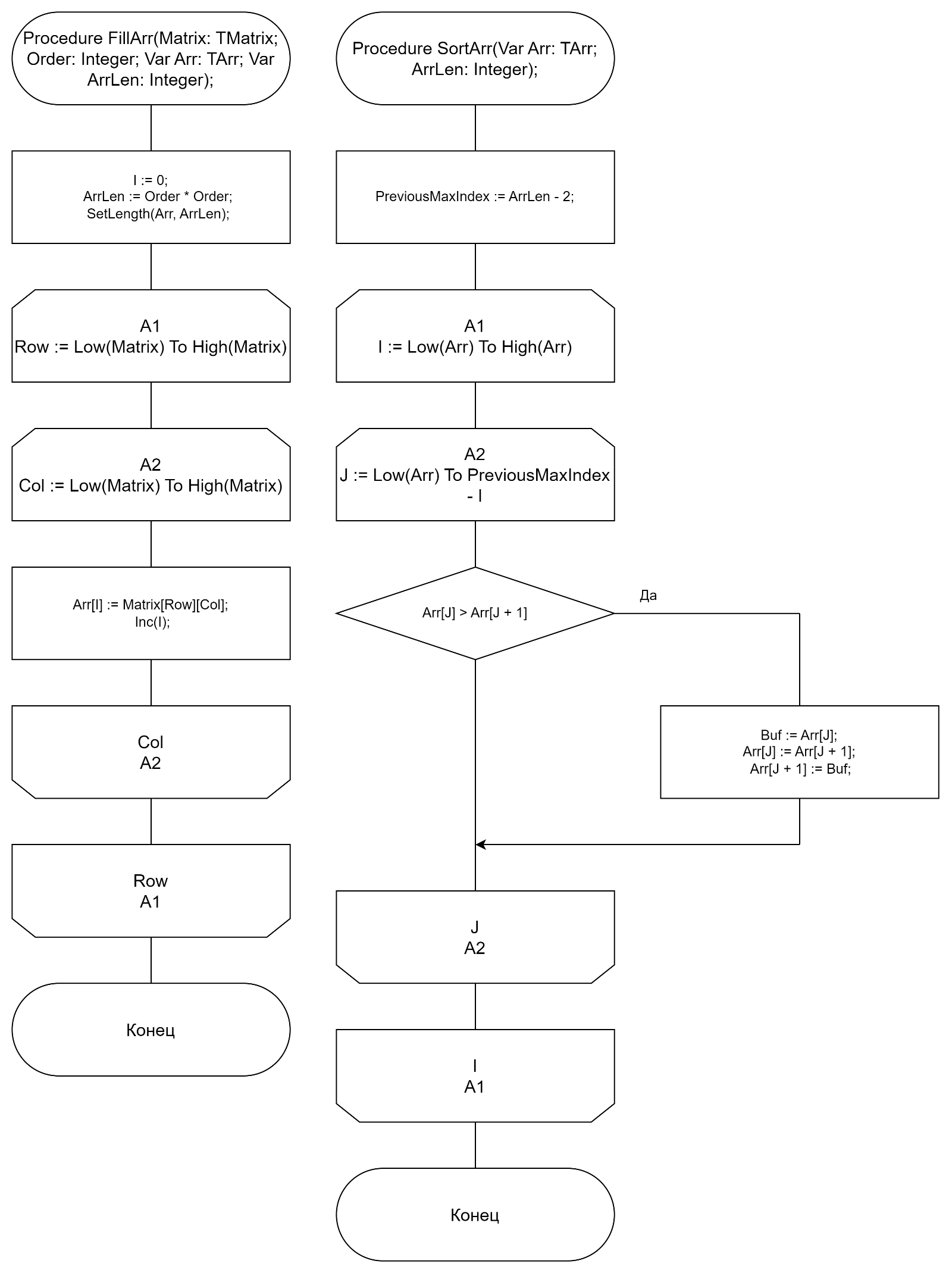
****

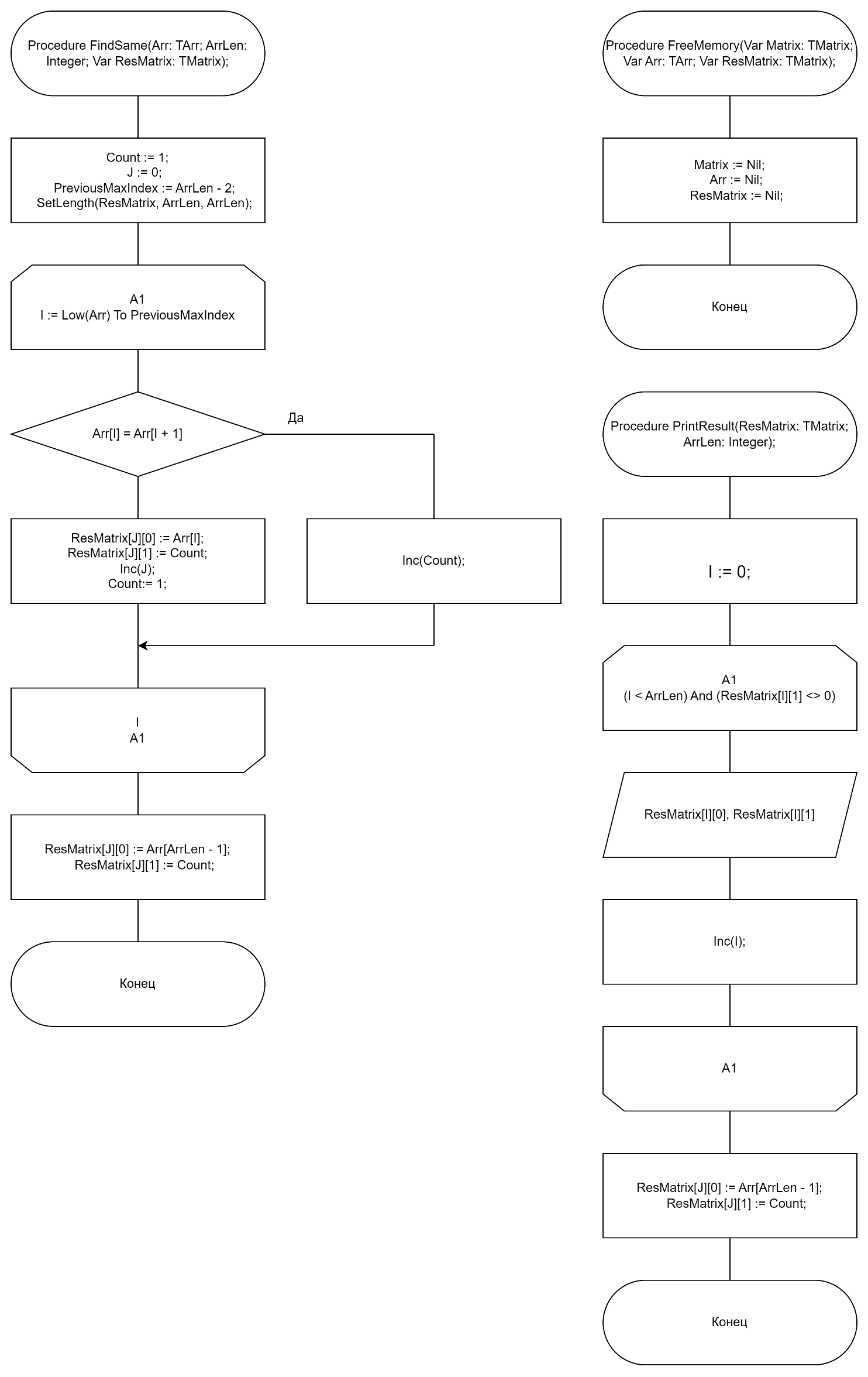
****

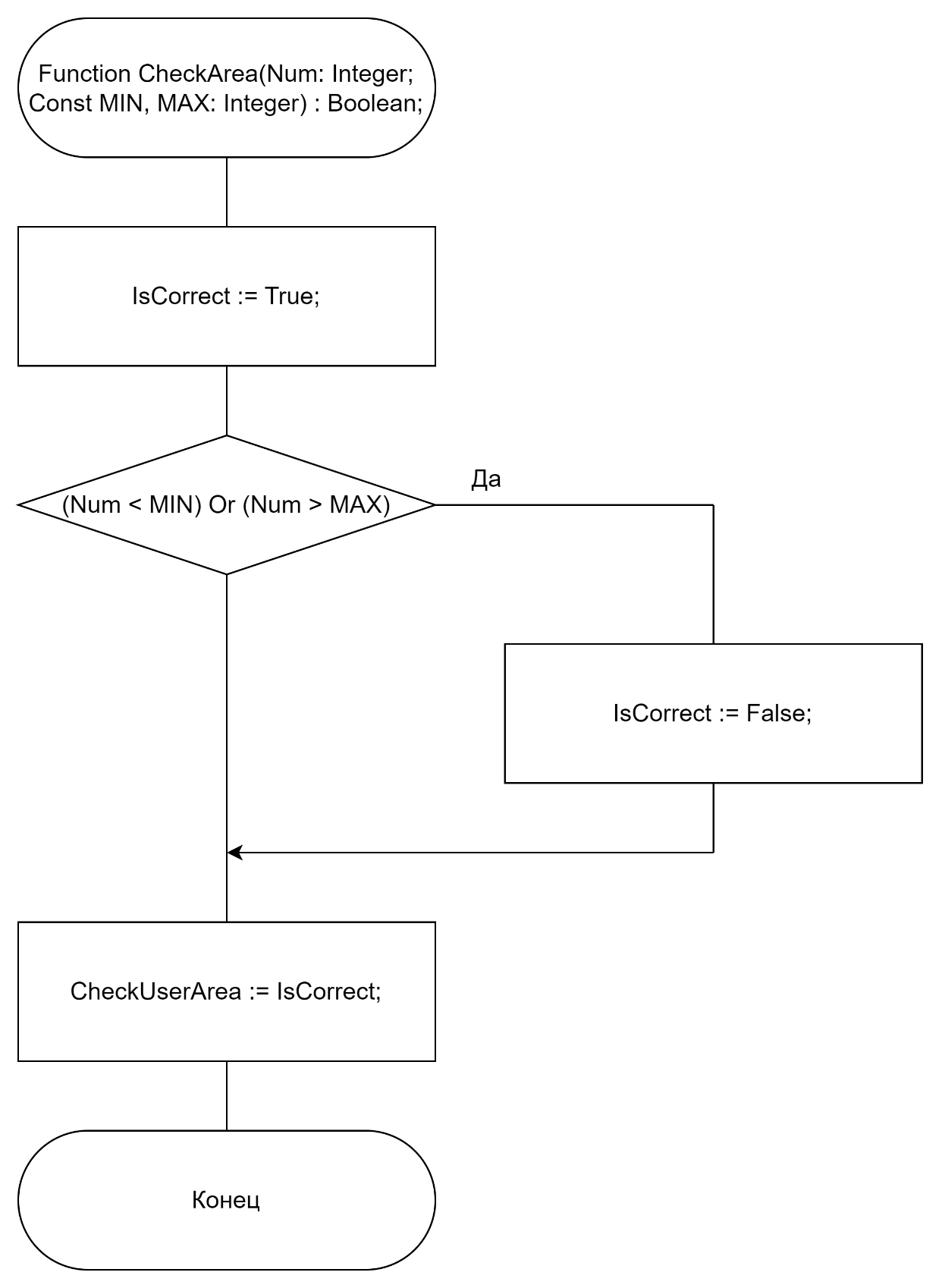
****

****

****

****

****

****