УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.3

по предмету «Основы алгоритмизации и программирования»

Вариант 12

Выполнил:

Галуха П. А.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Дана матрица А порядка n. Подсчитать количество строк матрицы, элементы которых представляют перестановки чисел от 1 до n.

**Код программы Delphi:**

Program Lab23;

Uses

System.SysUtils;

Type

TMatrix = Array Of Array Of Integer;

Const

MIN\_O = 1;

MAX\_O = 10;

MIN\_MAT = -1000000;

MAX\_MAT = 1000000;

YES = 1;

NO = 2;

Procedure PrintTask();

Begin

WriteLn('Данная программа подсчитывает число строк в матрице, которые составлены из

перестановок чисел от 1 до порядка матрицы.');

WriteLn;

End;

Function ChooseFileInput() : Boolean;

Var

IsFileInput: Integer;

IsCorrect, Choose: Boolean;

Begin

Repeat

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

Try

ReadLn(IsFileInput);

IsCorrect := True;

Except

WriteLn('Некорректный выбор!');

End;

If IsCorrect Then

Begin

If IsFileInput = YES Then

Choose := True

Else If IsFileInput = NO Then

Choose := False

Else

Begin

IsCorrect := False;

WriteLn('Некорректный выбор!');

End;

End;

Until IsCorrect;

ChooseFileInput := Choose;

End;

Function CheckUserArea(Num, MIN, MAX: Integer) : Boolean;

Var

IsCorrect: Boolean;

Begin

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

Begin

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

IsCorrect := False;

End

Else

IsCorrect := True;

CheckUserArea := IsCorrect;

End;

Function ReadUserO(Num: Integer; MIN, MAX: Integer) : Integer;

Var

IsCorrect: Boolean;

Begin

Repeat

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

Try

Readln(Num);

IsCorrect := True;

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckUserArea(Num, MIN, MAX);

Until IsCorrect;

ReadUserO := Num;

End;

Function ReadUserMatrix(Num: Integer; Row, Col, MIN, MAX: Integer) : Integer;

Var

IsCorrect: Boolean;

Begin

Repeat

Write('Введите в ', (Row + 1), ' строке ', (Col + 1), ' столбце элемент[', MIN, '; ',

MAX, ']: ');

Try

Readln(Num);

IsCorrect := True;

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckUserArea(Num, MIN, MAX);

Until IsCorrect;

ReadUserMatrix := Num;

End;

Function ReadPathToFile() : String;

Var

IsCorrect: Boolean;

PathToFile: String;

Begin

Repeat

Write('Введите путь к файлу с расширением .txt с матрицей, у которой разряд не должен

превышать ' , MAX\_O, ', а её 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

If FileExists(PathToFile) Then

IsCorrect := True

Else

IsCorrect := False;

IsExists := IsCorrect;

End;

Function IsAbleToReading(Var F: TextFile) : Boolean;

Var

IsCorrect: Boolean;

Begin

Try

Reset(F);

CloseFile(F);

IsCorrect := True;

Except

IsCorrect := False;

End;

IsAbleToReading := IsCorrect;

End;

Function IsAbleToWriting(PathToFile: String) : Boolean;

Var

IsCorrect: Boolean;

Begin

If FileIsReadOnly(PathToFile) Then

IsCorrect := False

Else

IsCorrect := True;

IsAbleToWriting := IsCorrect;

End;

Function IsEmpty(Var F: TextFile) : Boolean;

Var

Buf: Char;

Size: Integer;

IsCorrect: Boolean;

Begin

Reset(F);

Size := 0;

If Not EOF(F) Then

Size := 1;

CloseFile(F);

If Size = 0 Then

IsCorrect := True

Else

IsCorrect := False;

IsEmpty := IsCorrect;

End;

Function ReadOrder(Var F: TextFile) : Integer;

Var

Buf: Char;

IsCorrect: Boolean;

Order: Integer;

Begin

IsCorrect := True;

Reset(F);

Try

Read(F, Order);

Except

Writeln('Неправильный порядок матрицы!');

IsCorrect := False;

End;

If IsCorrect Then

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

If IsCorrect Then

Begin

Readln(F, Buf);

If Buf <> #13 Then

Begin

IsCorrect := False;

Writeln('Неправильный порядок матрицы!');

End;

End;

CloseFile(F);

If Not IsCorrect Then

Order := -1;

ReadOrder := Order;

End;

Function IsRightNums(Var F: TextFile) : Boolean;

Var

K: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := True;

Reset(F);

Readln(F);

While IsCorrect And Not EOF(F) Do

Begin

While IsCorrect And Not EOLN(F) Do

Begin

Try

Read(F, K);

Except

IsCorrect := False;

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

End;

If IsCorrect Then

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

End;

ReadLn(F);

End;

CloseFile(F);

IsRightNums := IsCorrect;

End;

Function IsOrdersEqual(Var F: TextFile; Order: Integer) : Boolean;

Var

Rows, Cols, K: Integer;

IsCorrect: Boolean;

Begin

Rows := 0;

Reset(F);

Readln(F);

IsCorrect := True;

While IsCorrect And Not EOF(F) Do

Begin

Cols := 0;

While IsCorrect And Not EOLN(F) Do

Begin

Read(F, K);

Inc(Cols);

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

End;

If IsCorrect Then

Begin

Readln(F);

Inc(Rows);

IsCorrect := CheckUserArea(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;

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 ReadFile(Var F: TextFile) : TMatrix;

Var

IsCorrect: Boolean;

PathToFile: String;

Order: Integer;

Matrix: TMatrix;

Begin

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 Not IsAbleToWriting(PathToFile) Then

Begin

IsCorrect := False;

WriteLn('Файл закрыт для записи!');

End;

If IsCorrect And IsEmpty(F) Then

Begin

IsCorrect := False;

WriteLn('Файл пуст!');

End;

If IsCorrect Then

Begin

Order := ReadOrder(F);

If Order = -1 Then

IsCorrect := False;

End;

If IsCorrect And Not IsRightNums(F) Then

IsCorrect := False;

If IsCorrect And Not IsOrdersEqual(F, Order) Then

Begin

IsCorrect := False;

Writeln('Значения порядков не равны!');

End;

Until IsCorrect;

Matrix := ReadFileMatrix(F, Order);

ReadFile := Matrix;

End;

Procedure AddAnswerToFile(Var F: TextFile; CountCorrectRows: Integer);

Begin

Append(F);

WriteLn(F);

WriteLn(F, 'Количество строк, имеющих перестановки элементов от 1 до порядка матрицы: ',

CountCorrectRows);

CloseFile(F);

End;

Function ReadConsole() : TMatrix;

Var

Row, Col, Order: Integer;

Matrix: TMatrix;

Begin

Order := ReadUserO(Order, MIN\_O, MAX\_O);

SetLength(Matrix, Order, Order);

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

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

Matrix[Row][Col] := ReadUserMatrix(Matrix[Row][Col], Row, Col, MIN\_MAT, MAX\_MAT);

ReadConsole := Matrix;

End;

Function Sort(Matrix: TMatrix) : TMatrix;

Var

Row, I, J, PreviousMaxIndex, Buf: Integer;

Begin

PreviousMaxIndex := High(Matrix) - 1;

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

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

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

If Matrix[Row][J] > Matrix[Row][J + 1] Then

Begin

Buf := Matrix[Row][J];

Matrix[Row][J] := Matrix[Row][J + 1];

Matrix[Row][J + 1] := Buf;

End;

Sort := Matrix;

End;

Function CalcCorrectRows(Matrix: TMatrix) : Integer;

Var

Count, I, Row: Integer;

IsCorrect: Boolean;

Begin

Count := 0;

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

Begin

IsCorrect := True;

I := 0;

While (IsCorrect) And (I <= High(Matrix)) Do

Begin

If (Matrix[Row][I] <> I + 1) Then

IsCorrect := False;

Inc(I);

End;

If IsCorrect Then

Inc(Count);

End;

CalcCorrectRows := Count;

End;

Procedure FreeMemory(Matrix: TMatrix);

Begin

Matrix := Nil;

End;

Function GetResultFromFile() : Integer;

Var

F: TextFile;

Matrix: TMatrix;

CountCorrectRows: Integer;

Begin

Matrix := ReadFile(F);

Matrix := Sort(Matrix);

CountCorrectRows := CalcCorrectRows(Matrix);

FreeMemory(Matrix);

AddAnswerToFile(F, CountCorrectRows);

GetResultFromFile := CountCorrectRows;

End;

Function GetResultFromConsole() : Integer;

Var

Matrix: TMatrix;

CountCorrectRows: Integer;

Begin

Matrix := ReadConsole();

Matrix := Sort(Matrix);

CountCorrectRows := CalcCorrectRows(Matrix);

FreeMemory(Matrix);

GetResultFromConsole := CountCorrectRows;

End;

Procedure PrintResult(Count: Integer);

Begin

WriteLn;

WriteLn('Количество строк, имеющих перестановки элементов от 1 до N: ', Count);

End;

Var

CountCorrectRows: Integer;

Begin

PrintTask();

If ChooseFileInput() Then

CountCorrectRows := GetResultFromFile()

Else

CountCorrectRows := GetResultFromConsole();

PrintResult(CountCorrectRows);

Readln;

End.

**Код программы С++:**

#include <iostream>

#include <string.h>

#include <fstream>

const int

MIN\_O = 1,

MAX\_O = 10,

MIN\_MAT = -1000000,

MAX\_MAT = 1000000,

YES = 1,

NO = 2;

void printTask()

{

std::cout << "Данная программа подсчитывает число строк в матрице, которые составлены из

перестановок чисел от 1 до порядка матрицы.\n\n";

}

bool chooseFileInput()

{

int isFileInput;

bool isCorrect, choose;

do {

std::cout << "Вы хотите вводить матрицу через файл? (Да - " << YES << " / Нет - "

<< NO << ")\n";

isCorrect = true;

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 == 1)

choose = true;

else if (isFileInput == 2)

choose = false;

else

{

isCorrect = false;

std::cout << "Некорректный выбор!\n";

}

}

} while (!isCorrect);

return choose;

}

bool checkUserArea(int num, int MIN, int MAX)

{

bool isCorrect;

if (num < MIN || num > MAX)

{

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

isCorrect = false;

}

else

isCorrect = true;

return isCorrect;

}

int readUserN(int num, int MIN, int MAX)

{

bool isCorrect;

do

{

std::cout << "Введите порядок матрицы N[" << MIN << "; " << MAX << "]: ";

isCorrect = true;

std::cin >> num;

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 = checkUserArea(num, MIN, MAX);

} while (!isCorrect);

return num;

}

int readUserMatrix(int num, int row, int col, int MIN, int MAX)

{

bool isCorrect;

do

{

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

<< MIN << "; " << MAX << "]: ";

isCorrect = true;

std::cin >> num;

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 = checkUserArea(num, MIN, MAX);

} while (!isCorrect);

return num;

}

std::string readPathFile()

{

bool isCorrect;

int len;

std::string pathToFile;

do

{

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

не должен превышать " << MAX\_O << ", а её 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

{

isCorrect = false;

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

}

} while (!isCorrect);

return pathToFile;

}

bool isExists(std::string pathToFile)

{

bool isCorrect;

std::ifstream file(pathToFile);

if (file.good())

isCorrect = true;

else

isCorrect = false;

file.close();

return isCorrect;

}

bool isAbleToReading(std::string pathToFile)

{

bool isCorrect;

std::ifstream file(pathToFile);

if (file.is\_open())

isCorrect = true;

else

isCorrect = false;

file.close();

return isCorrect;

}

bool isAbleToWriting(std::string pathToFile)

{

bool isCorrect;

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

if (file.is\_open())

isCorrect = true;

else

isCorrect = false;

file.close();

return isCorrect;

}

bool isEmpty(std::string pathToFile)

{

bool isCorrect;

std::ifstream file(pathToFile);

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

isCorrect = true;

else

isCorrect = false;

file.close();

return isCorrect;

}

int readOrder(std::string pathToFile)

{

int order;

bool isCorrect;

isCorrect = true;

std::ifstream file(pathToFile);

file >> order;

if (file.fail())

{

isCorrect = false;

file.clear();

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

}

if (isCorrect)

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

if (isCorrect)

{

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

{

isCorrect = false;

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

}

}

file.close();

if (!isCorrect)

order = -1;

return order;

}

bool isRightNums(std::string pathToFile)

{

int k;

bool isCorrect;

std::ifstream file(pathToFile);

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

isCorrect = true;

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

{

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

{

file >> k;

if (file.fail())

{

isCorrect = false;

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

file.clear();

}

if (isCorrect)

isCorrect = checkUserArea(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)

{

int rows, cols, k;

bool isCorrect;

rows = 0;

std::ifstream file(pathToFile);

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

isCorrect = true;

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

{

cols = 0;

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

{

file >> k;

cols++;

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

}

if (isCorrect)

{

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

rows++;

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

if (cols != order)

isCorrect = false;

}

}

file.close();

if (isCorrect && rows != order)

isCorrect = false;

return isCorrect;

}

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

{

int row, col;

int\*\* matrix;

row = 0;

col = 0;

std::ifstream file(pathToFile);

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

matrix = new int\* [order];

while (row < order)

{

matrix[row] = new int[order];

while (col < order)

{

file >> matrix[row][col];

col++;

}

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

col = 0;

row++;

}

file.close();

return matrix;

}

int\*\* readFile(std::string& pathToFile, int& order)

{

bool isCorrect;

int\*\* matrix = new int\*[0];

isCorrect = true;

do

{

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)

{

order = readOrder(pathToFile);

if (order == -1)

isCorrect = false;

}

if (isCorrect && !isRightNums(pathToFile))

isCorrect = false;

if (isCorrect && !isOrdersEqual(pathToFile, order))

{

isCorrect = false;

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

}

} while (!isCorrect);

matrix = readFileMatrix(pathToFile, order);

return matrix;

}

void addAnswerToFile(std::string pathToFile, int countCorrectRows)

{

std::cout << pathToFile;

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

file << "\nКоличество строк, имеющих перестановки элементов от 1 до порядка матрицы: "

<< countCorrectRows;

file.close();

}

int\*\* readConsole(int& order)

{

order = readUserN(order, MIN\_O, MAX\_O);

int\*\* matrix = new int\* [order];

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

matrix[i] = new int[order];

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

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

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

return matrix;

}

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

{

int previousMaxIndex, buf;

previousMaxIndex = order - 1;

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

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

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

if (matrix[row][j] > matrix[row][j + 1])

{

buf = matrix[row][j];

matrix[row][j] = matrix[row][j + 1];

matrix[row][j + 1] = buf;

}

return matrix;

}

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

{

int i, count;

bool isCorrect;

isCorrect = true;

count = 0;

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

{

i = 0;

while (isCorrect && i < order)

{

if (matrix[row][i] != i + 1)

isCorrect = false;

i++;

}

if (isCorrect)

count++;

isCorrect = true;

}

return count;

}

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

{

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

delete[] matrix[i];

delete[] matrix;

}

int getResultFromFile()

{

int order;

std::string pathToFile;

int\*\* matrix;

int countCorrectRows;

matrix = readFile(pathToFile, order);

matrix = sort(matrix, order);

countCorrectRows = calcCorrectRows(matrix, order);

freeMemory(matrix, order);

addAnswerToFile(pathToFile, countCorrectRows);

return countCorrectRows;

}

int getResultFromConsole()

{

int order;

int\*\* matrix;

int countCorrectRows;

matrix = readConsole(order);

matrix = sort(matrix, order);

countCorrectRows = calcCorrectRows(matrix, order);

freeMemory(matrix, order);

return countCorrectRows;

}

void printResult(int count)

{

std::cout << "\nКоличество строк, имеющих перестановки элементов от 1 до N: " << count;

}

int main()

{

setlocale(LC\_ALL, "RU");

int countCorrectRows;

printTask();

if (chooseFileInput())

countCorrectRows = getResultFromFile();

else

countCorrectRows = getResultFromConsole();

printResult(countCorrectRows);

return 0;

}

**Код программы Java:**

import java.io.File;

import java.util.Scanner;

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 строк в матрице, которые составлены

из перестановок чисел от 1 до порядка матрицы.\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 == 1)

choose = true;

else if (isFileInput == 2)

choose = false;

else {

isCorrect = false;

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

}

}

} while (!isCorrect);

return choose;

}

public static boolean checkUserArea(int num, int MIN, int MAX) {

boolean isCorrect;

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

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

isCorrect = false;

}

else

isCorrect = true;

return isCorrect;

}

public static int readUserO(int num, int MIN, int MAX) {

boolean isCorrect;

do {

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

isCorrect = true;

try {

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

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

isCorrect = checkUserArea(num, MIN, MAX);

} while (!isCorrect);

return num;

}

public static int readUserMatrix(int num, int row, int col, int MIN, int MAX) {

boolean isCorrect;

do {

System.out.print("Введите в " + (row + 1) + " строке " + (col + 1) + " столбце элемент

[" + MIN + "; " + MAX + "]: ");

isCorrect = true;

try {

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

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

isCorrect = checkUserArea(num, MIN, MAX);

} while (!isCorrect);

return num;

}

public static String readPathFile() {

boolean isCorrect;

int len;

String pathToFile;

do {

System.out.println("Введите путь к файлу с расширением .txt с матрицей, у которой

разряд не должен превышать " + MAX\_O + ", а её 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;

file = new File(pathToFile);

if (file.exists())

isCorrect = true;

else

isCorrect = false;

return isCorrect;

}

public static boolean isAbleToReading() {

boolean isCorrect;

if (file.canRead())

isCorrect = true;

else

isCorrect = false;

return isCorrect;

}

public static boolean isAbleToWriting() {

boolean isCorrect;

if (file.canWrite())

isCorrect = true;

else

isCorrect = false;

return isCorrect;

}

public static boolean isEmpty() {

boolean isCorrect;

if (file.length() == 0)

isCorrect = true;

else

isCorrect = false;

return isCorrect;

}

public static int readOrder() {

int order;

boolean isCorrect;

order = 0;

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

isCorrect = true;

try {

order = scanFile.nextInt();

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

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

if (isCorrect) {

if (!scanFile.hasNextLine()) {

isCorrect = false;

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

}

}

scanFile.close();

if (!isCorrect)

order = -1;

return order;

}

public static boolean isRightNums() {

int k;

boolean isCorrect;

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

scanFile.nextLine();

isCorrect = true;

k = 0;

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

try {

k = scanFile.nextInt();

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

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

}

scanFile.close();

return isCorrect;

}

public static boolean isOrdersEqual(int order) {

int rows, cols, k;

boolean isCorrect;

rows = 0;

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

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 = checkUserArea(cols, MIN\_O, MAX\_O);

}

if (isCorrect) {

rows++;

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

if (cols != order)

isCorrect = false;

}

}

scanFile.close();

if (isCorrect && rows != order)

isCorrect = false;

return isCorrect;

}

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

int row, col;

int[][] matrix = new int[order][order];

row = 0;

col = 0;

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

scanFile.nextLine();

while (row < order) {

matrix[row] = new int[order];

while (col < order) {

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

col++;

}

col = 0;

row++;

}

scanFile.close();

return matrix;

}

public static int[][] readFile() {

String pathToFile;

int order;

boolean isCorrect;

int[][] matrix;

order = 0;

isCorrect = true;

do {

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) {

order = readOrder();

if (order == -1)

isCorrect = false;

}

if (isCorrect && !isRightNums())

isCorrect = false;

if (isCorrect && !isOrdersEqual(order)) {

isCorrect = false;

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

}

} while (!isCorrect);

matrix = readFileMatrix(order);

return matrix;

}

public static void addAnswerToFile(int countCorrectRows) {

try {

FileWriter writer = new FileWriter(file, true);

writer.write("\nКоличество строк, имеющих перестановки элементов от 1 до N: " +

countCorrectRows);

writer.close();

} catch (IOException e) {}

}

public static int[][] readConsole() {

int order;

order = 0;

order = readUserO(order, MIN\_O, MAX\_O);

int[][] matrix = new int[order][order];

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

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

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

return matrix;

}

public static int[][] sort(int[][] matrix) {

int order;

int previousMaxIndex, buf;

order = matrix.length;

previousMaxIndex = order - 1;

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

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

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

if (matrix[row][j] > matrix[row][j + 1]) {

buf = matrix[row][j];

matrix[row][j] = matrix[row][j + 1];

matrix[row][j + 1] = buf;

}

return matrix;

}

public static int calcCorrectRows(int[][] matrix) {

int i, count, order;

boolean isCorrect;

isCorrect = true;

order = matrix.length;

count = 0;

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

i = 0;

while (isCorrect && i < order) {

if (matrix[row][i] != i + 1)

isCorrect = false;

i++;

}

if (isCorrect)

count++;

isCorrect = true;

}

return count;

}

public static int getResultFromFile() {

int[][] matrix;

int countCorrectRows;

matrix = readFile();

matrix = sort(matrix);

countCorrectRows = calcCorrectRows(matrix);

addAnswerToFile(countCorrectRows);

return countCorrectRows;

}

public static int getResultFromConsole() {

int[][] matrix;

int countCorrectRows;

matrix = readConsole();

matrix = sort(matrix);

countCorrectRows = calcCorrectRows(matrix);

return countCorrectRows;

}

public static void printResult(int count) {

System.out.println("\nКоличество строк, имеющих перестановки элементов от 1 до N: " + count);

}

public static void main(String[] args) {

int countCorrectRows;

countCorrectRows = 0;

printTask();

if (chooseFileInput())

countCorrectRows = getResultFromFile();

else

countCorrectRows = getResultFromConsole();

scanConsole.close();

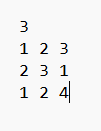
printResult(countCorrectRows);

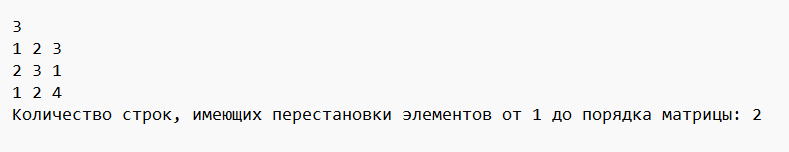
}

}

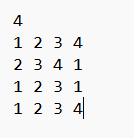
**Скриншоты:**

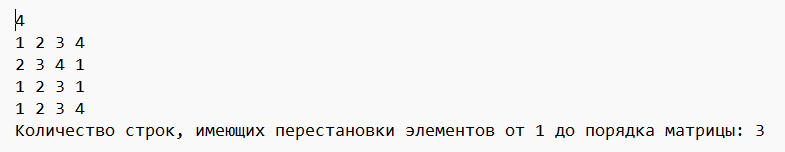
**Delphi:**





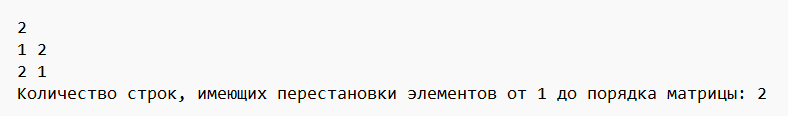
**C++:**



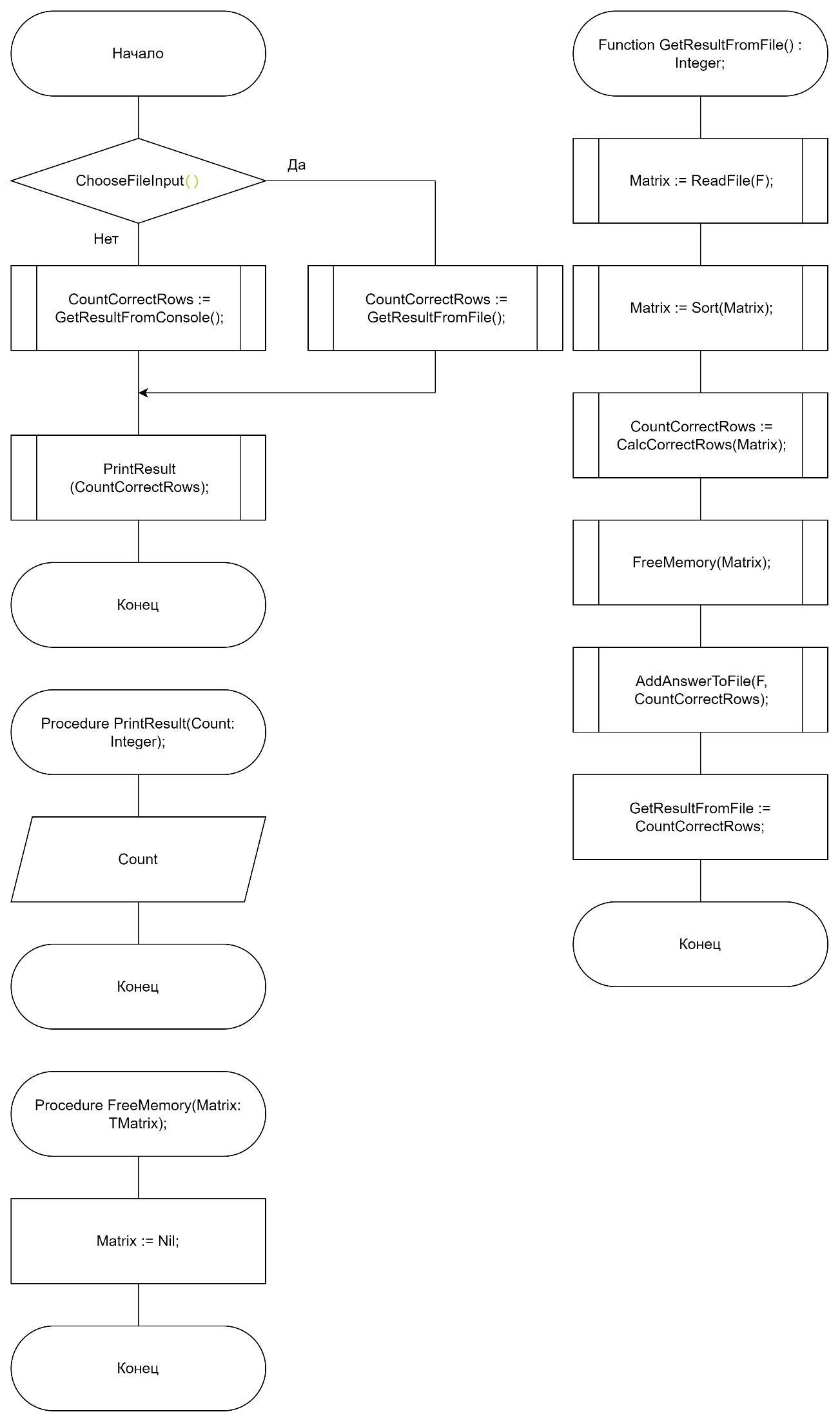


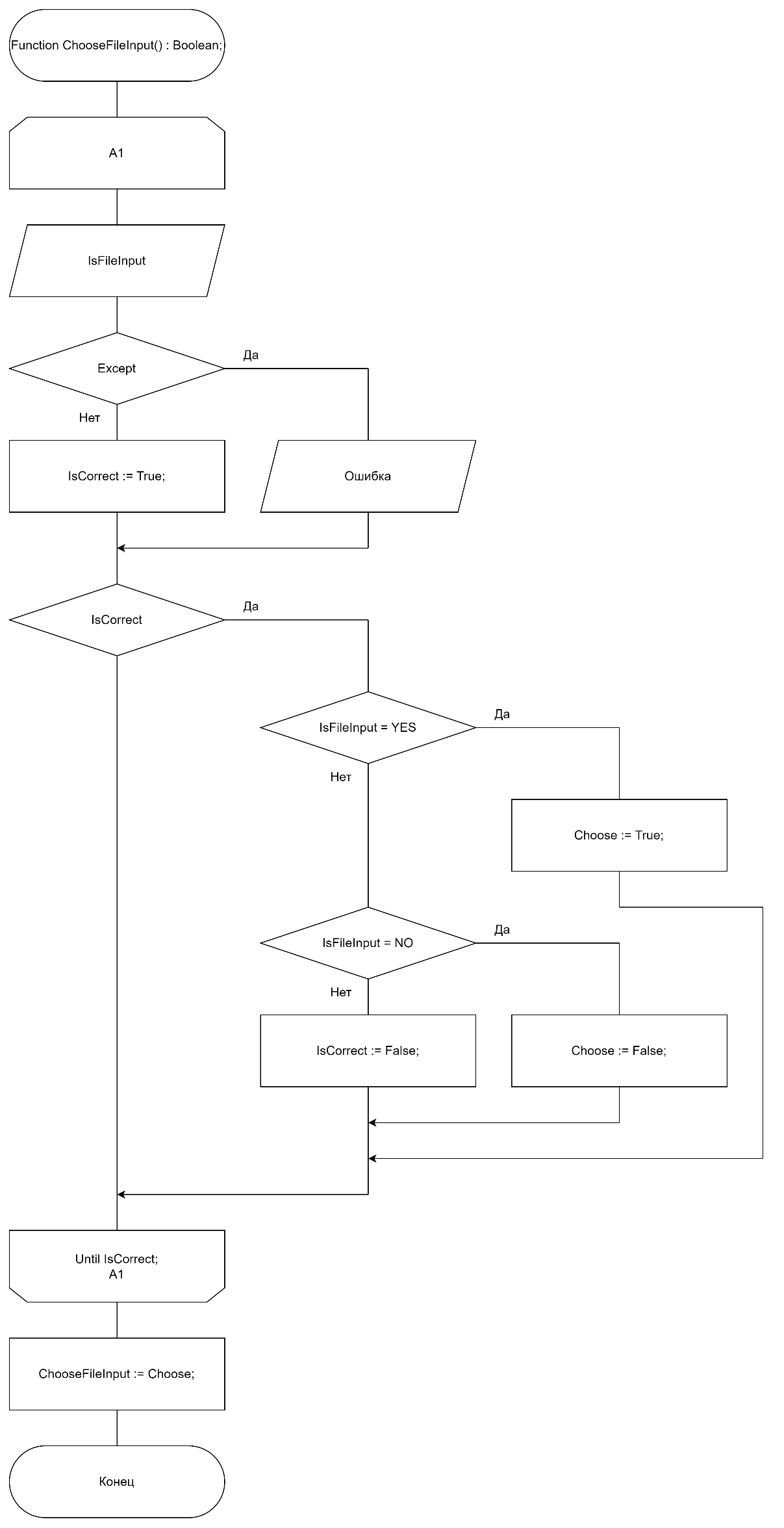
**Java:**

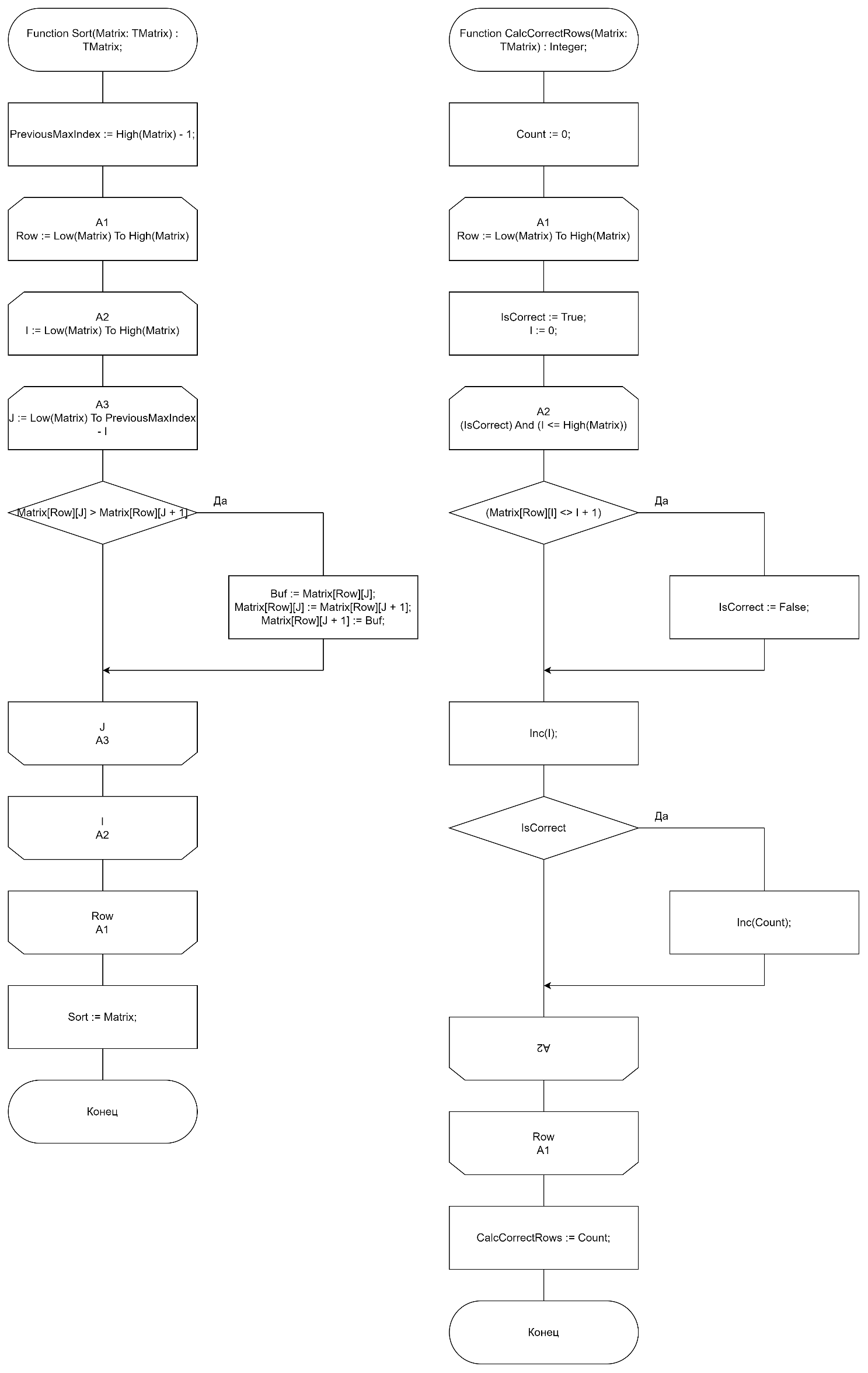
****

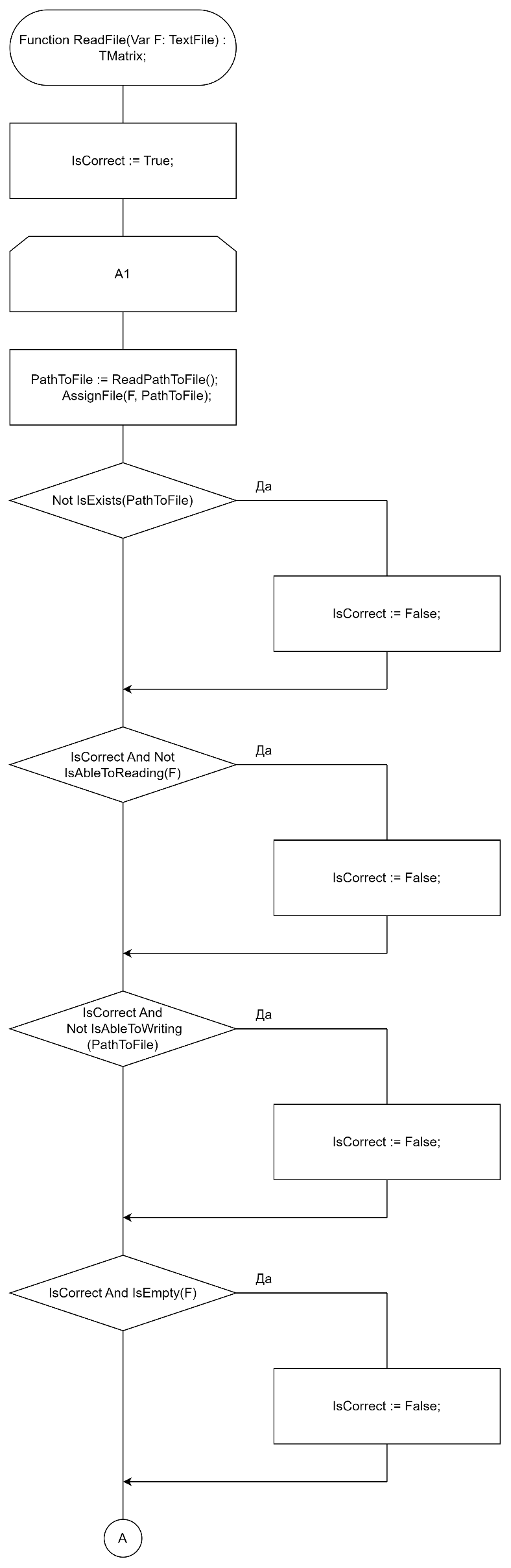
****

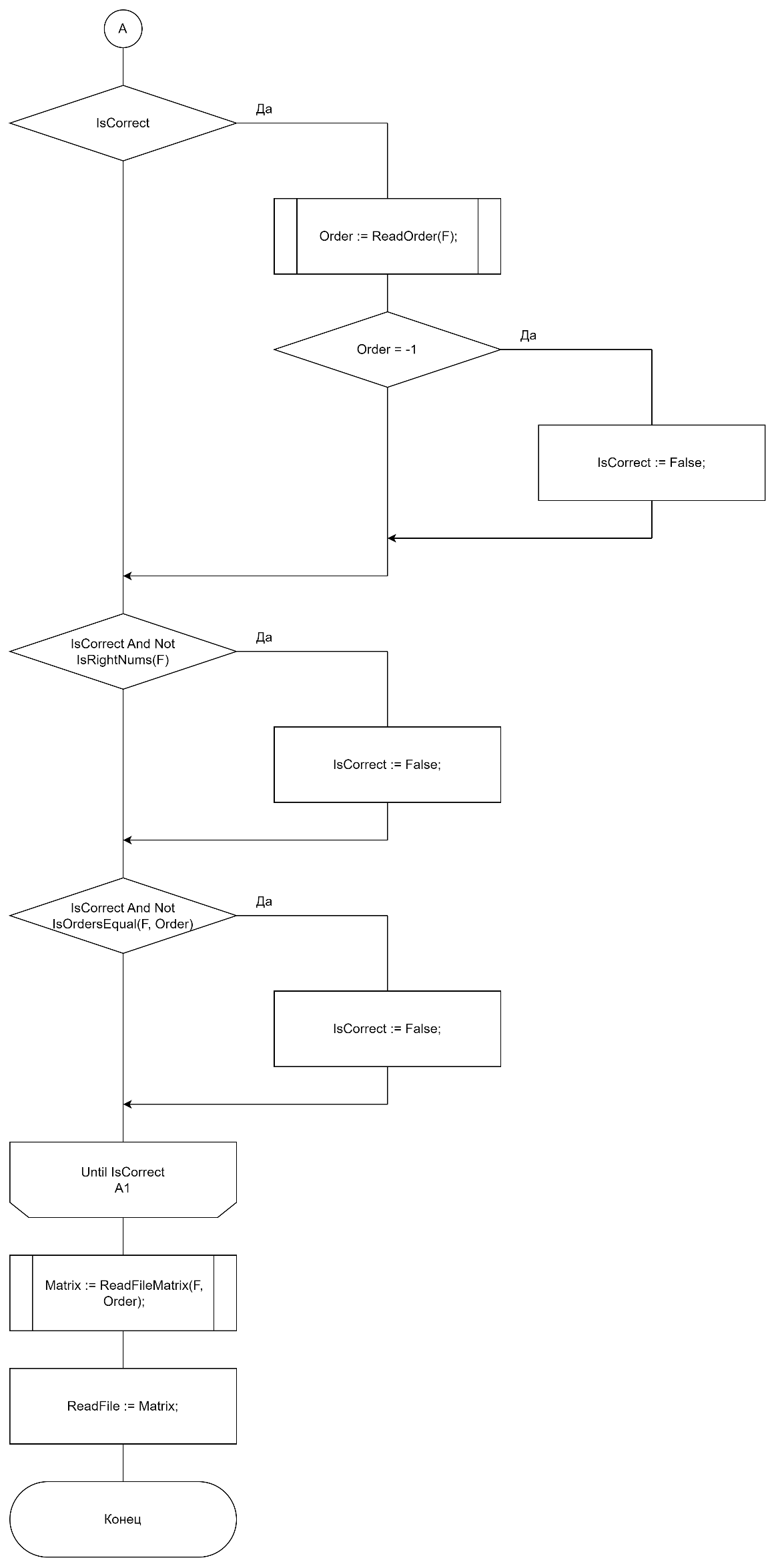
**Блок-схема:**

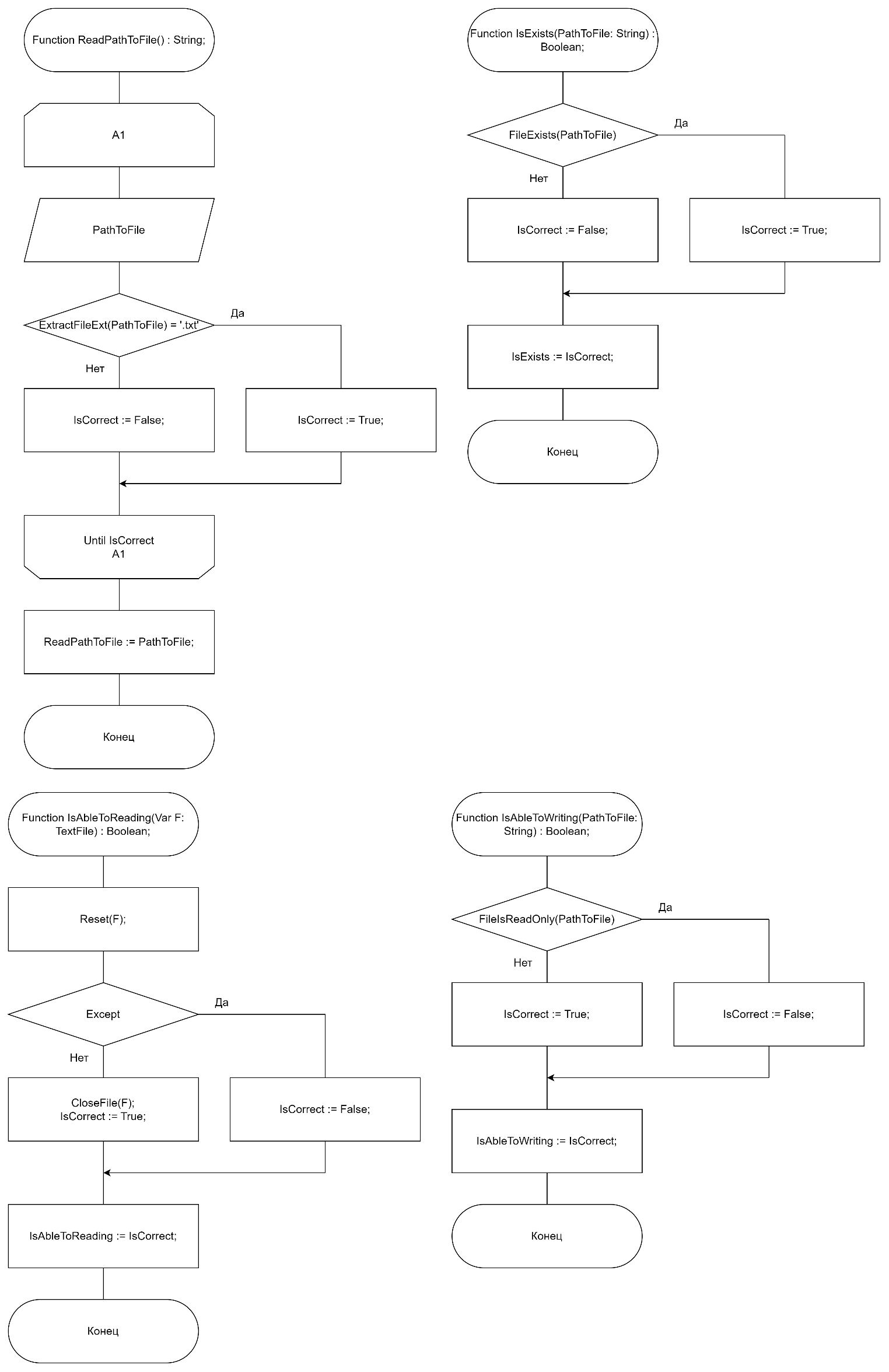


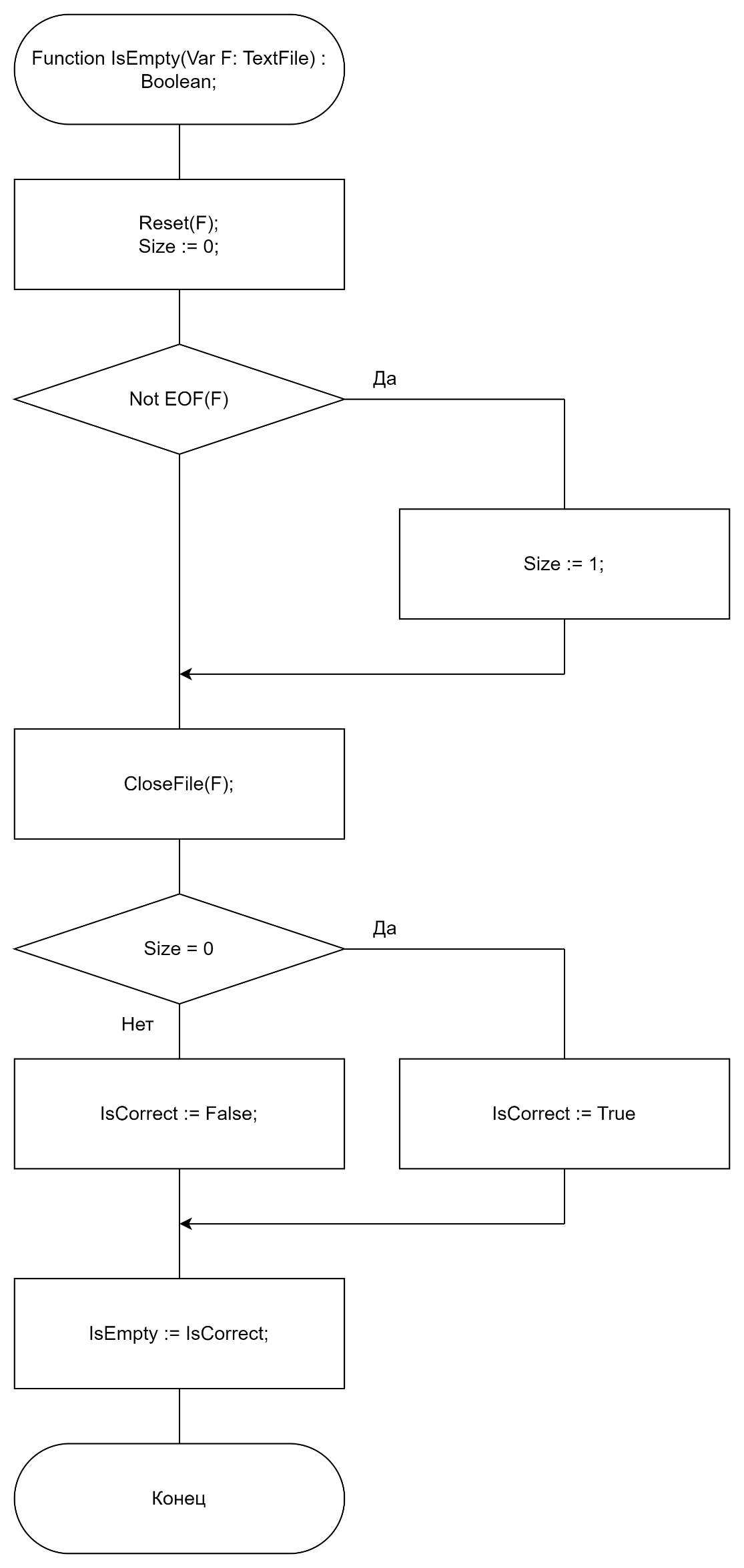


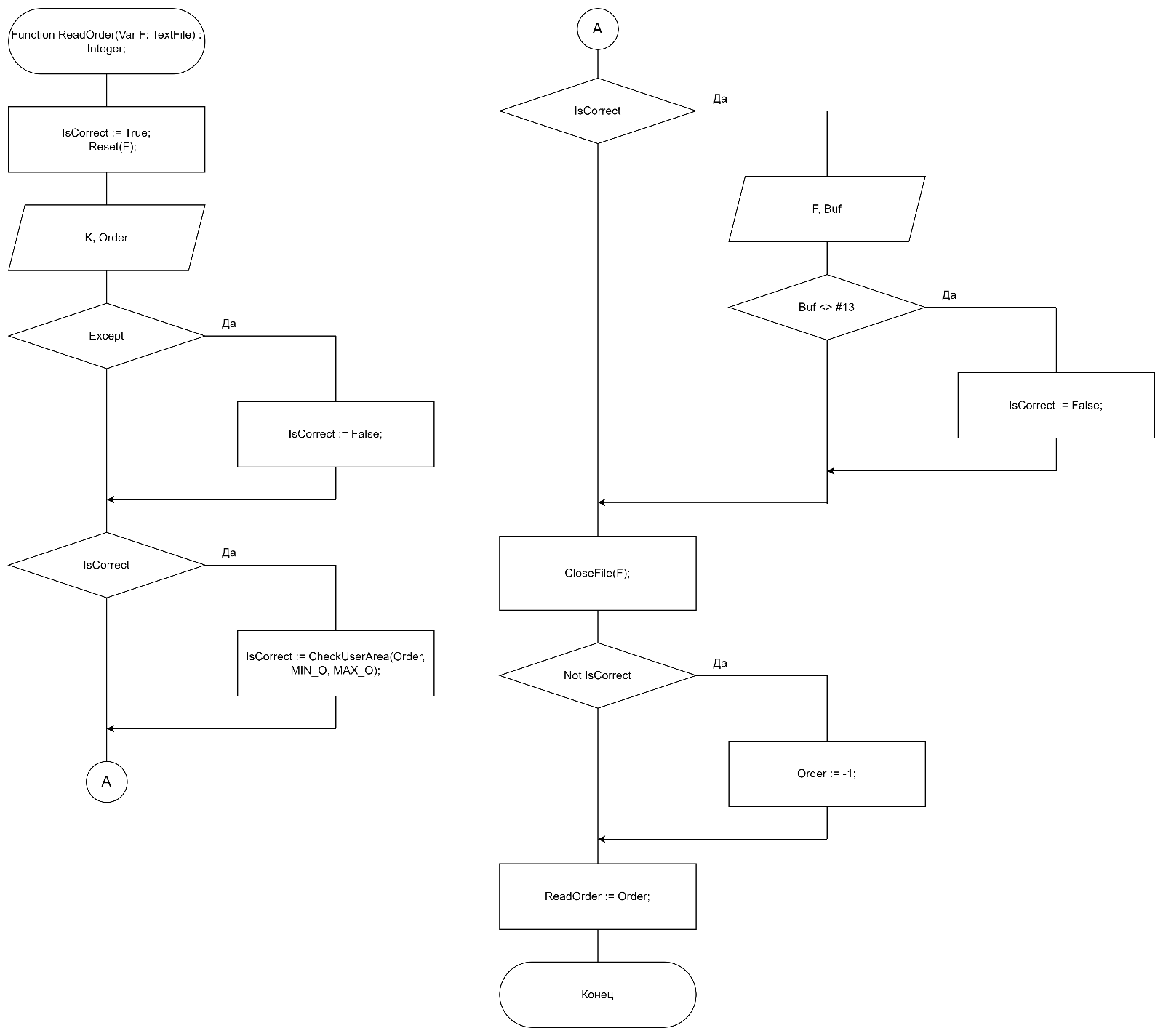






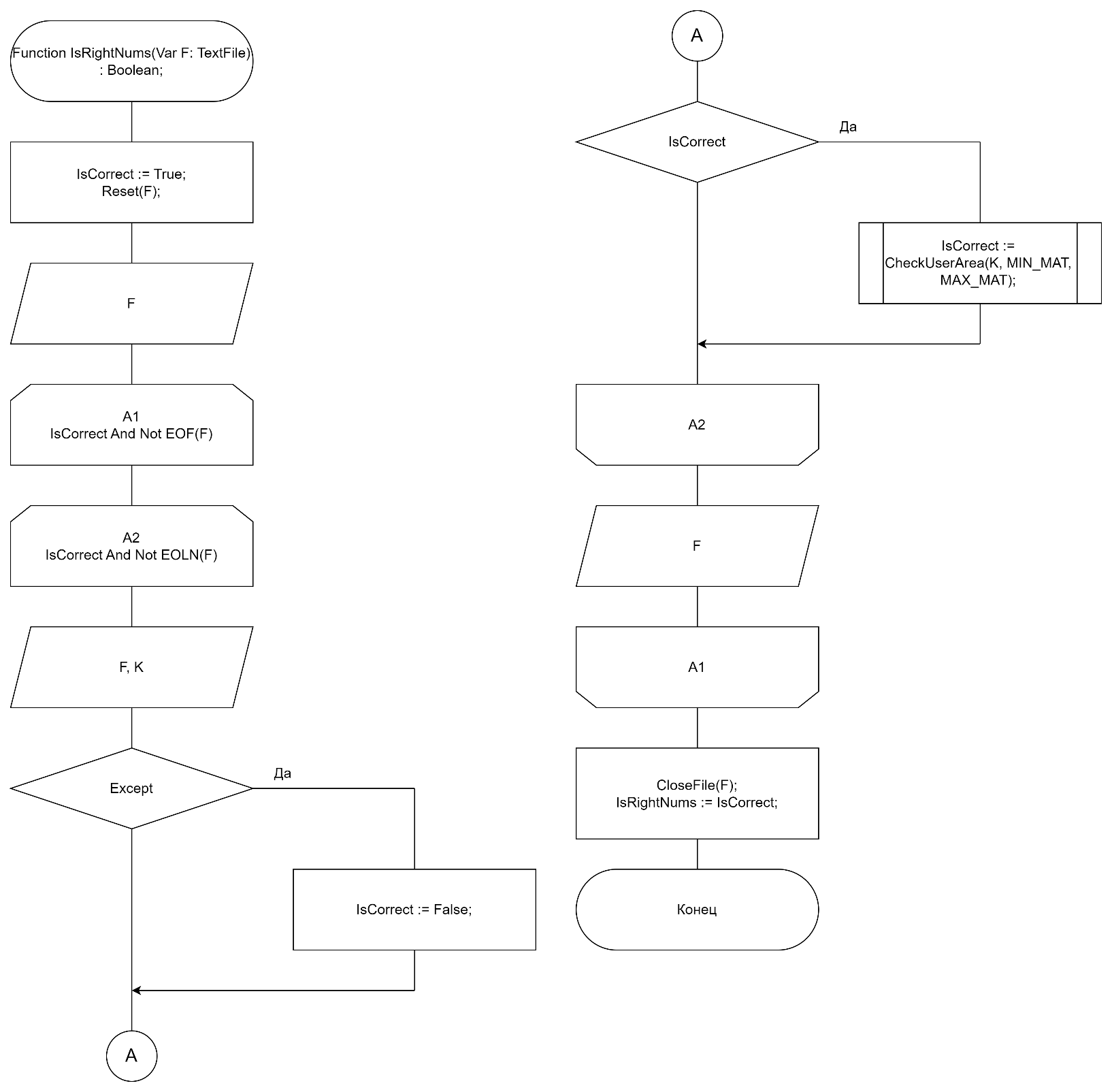
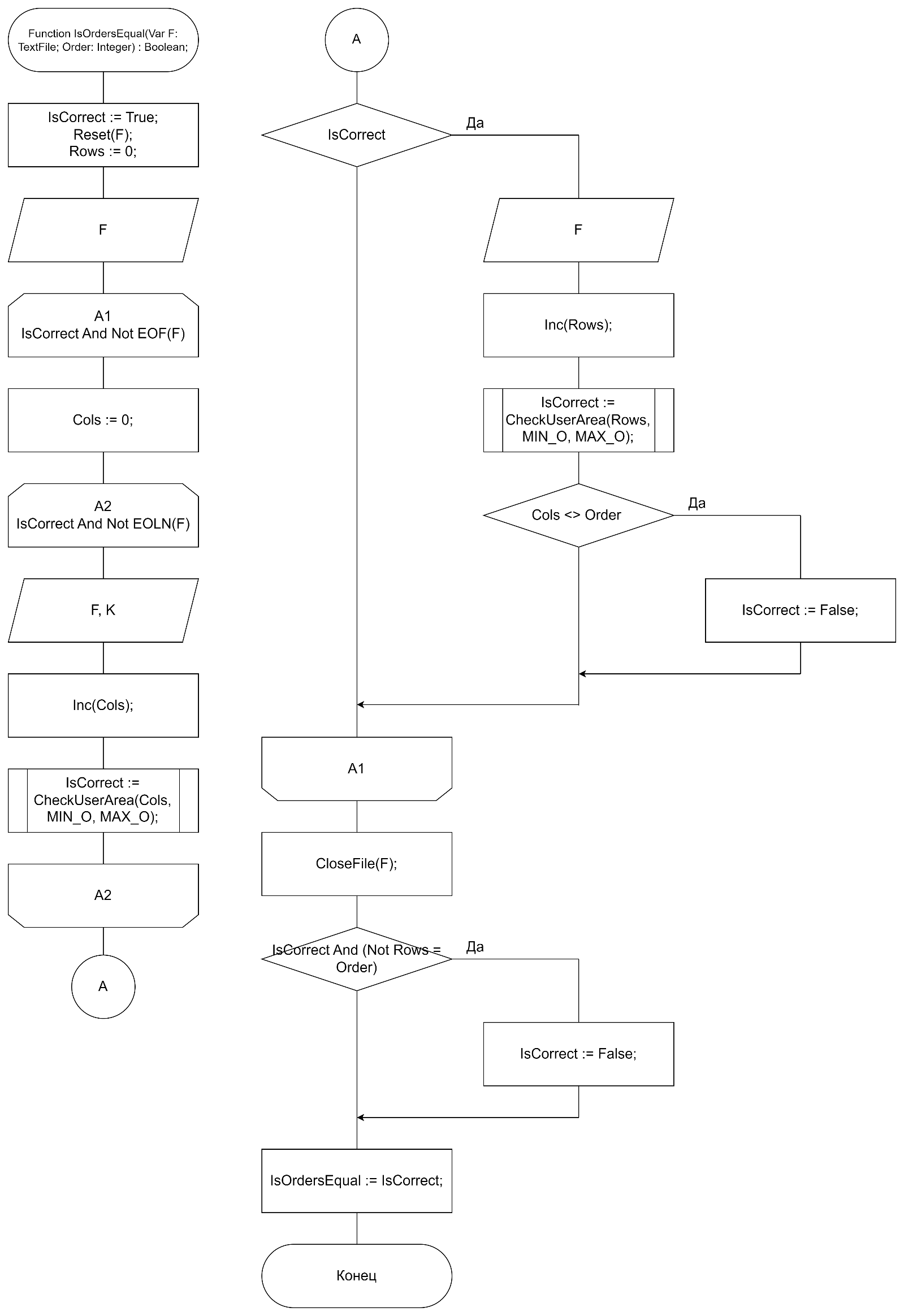






B

B

C

C

D

D

