УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.3

по предмету «Основы алгоритмизации и программирования»

Вариант 18

Выполнил:

Егоров А.С.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

В данной матрице А порядка n найти наибольшую сумму, образованную двумя соседними элементами. Соседями считаются элементы справа, слева, сверху и снизу.

**Код программы Delphi:**

Program Exercise3;

Uses

System.SysUtils;

Const

MIN\_MAX\_ORDER = 2;

Type

MatrixType = Array Of Array Of Real;

Procedure InitializationMatrix(Var Matrix: MatrixType; Const N: Integer);

Var

I: Integer;

Begin

SetLength(Matrix, N + 1, N + 1);

End;

Procedure Free(Var Matrix: MatrixType);

Begin

Matrix := Nil;

End;

Function InputMethod(): Boolean;

Var

IsCorrect: Boolean;

Choice: String;

Begin

IsCorrect := False;

Choice := '';

Writeln('The program works with console input or files.');

Repeat

Writeln('To use console enter ', #39, 'console', #39, '.', #13#10,

'To use a file enter ', #39, 'file', #39, '.'#13#10,

'Enter what type you want to use: ');

Readln(Choice);

If Choice = 'console' Then

Begin

IsCorrect := True;

Result := True;

End

Else If Choice = 'file' Then

Begin

IsCorrect := True;

Result := False;

End

Else

Writeln('The word ', Choice, ' don', #39,

't match any of method to input the data.');

Until IsCorrect;

End;

Function InputSizeOfMatrixFromConsole(): Integer;

Var

IsCorrect: Boolean;

N: Integer;

Begin

N := 0;

// asking for Size of matrix

Repeat

IsCorrect := False;

Try

Writeln('Enter the matrix order:');

Readln(N);

IsCorrect := True;

Except

Writeln('Invalid type. Try again.');

End;

If (N < MIN\_MAX\_ORDER) And IsCorrect Then

Begin

Writeln('Matrix order cannot be less than ', MIN\_MAX\_ORDER, '.');

IsCorrect := False;

End;

Until IsCorrect;

N := N - 1;

Result := N;

End;

Function InputElementsOfMatrixFromConsole(Const N: Integer): MatrixType;

Var

IsCorrect: Boolean;

I: Integer;

J: Integer;

Matrix: MatrixType;

Begin

InitializationMatrix(Matrix, N);

IsCorrect := False;

For I := 0 To N Do

Begin

For J := 0 To N Do

Begin

IsCorrect := False;

Repeat

Try

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

Readln(Matrix[I][J]);

IsCorrect := True;

Except

Writeln('Invalid type! Try again.');

End;

Until IsCorrect;

End;

End;

Result := Matrix;

End;

Function IsFileTxt(Var FileName: String): Boolean;

Var

FileType: String;

Begin

FileType := FileName.Substring(FileName.Length - 4);

If FileType = '.txt' Then

Result := True

Else

Result := False;

End;

Function IsFileReadable(Var InFile: TextFile): Boolean;

Begin

Try

Reset(InFile);

Result := True;

Except

Result := False;

End;

CloseFile(InFile);

End;

Function IsFileWritable(Var OutFile: TextFile): Boolean;

Begin

Try

Rewrite(OutFile);

Result := True;

Except

Result := False;

End;

CloseFile(OutFile);

End;

Function InputReadFileName(): String;

Var

FileName: String;

IsCorrect: Boolean;

InFile: TextFile;

Begin

FileName := '';

IsCorrect := False;

Repeat

IsCorrect := False;

Assign(InFile, FileName);

// Inputting name of file or path to the file including file

Writeln('Enter the name of file in this directory or path to this file

including name of file:');

Readln(FileName);

If (Not FileExists(FileName)) Then

Writeln('This file or the path to the file is specified incorrectly or does

not exist! Try again.')

Else If (Not IsFileTxt(FileName)) Then

Writeln('This file or path to the file isn', #39,

't .txt! Try again.')

Else If (Not IsFileReadable(InFile)) Then

Writeln('The program can', #39, 't read this file!s Try again.')

Else

IsCorrect := True;

Until IsCorrect;

Result := FileName;

End;

Function InputWriteFileName(): String;

Var

FileName: String;

IsCorrect: Boolean;

OutFile: TextFile;

Begin

FileName := '';

IsCorrect := False;

Repeat

Assign(OutFile, FileName);

IsCorrect := False;

// Inputting name of file or path to the file including file

Writeln('Enter the name of file in this directory or path to this file

including name of file:');

Readln(FileName);

If (Not FileExists(FileName)) Then

Writeln('This file or the path to the file is specified incorrectly or does

not exist! Try again.')

Else If (Not IsFileTxt(FileName)) Then

Writeln('This file or path to the file isn', #39,

't .txt! Try again.')

Else If (Not IsFileWritable(OutFile)) Then

Writeln('The program can', #39,

't write into this file! Try again.')

Else

IsCorrect := True;

Until IsCorrect;

Result := FileName;

End;

Function InputSizeOfMatrixFromFile(Var InFile: TextFile): Integer;

Var

IsCorrect: Boolean;

N: Integer;

Begin

N := 0;

IsCorrect := False;

Try

Read(InFile, N);

IsCorrect := True;

Except

Writeln('Invalid type. Check data in the file.');

End;

If IsCorrect And (N < MIN\_MAX\_ORDER) Then

Begin

Writeln('Matrix order cannot be less than ', MIN\_MAX\_ORDER, '.');

N := 0;

End;

N := N - 1;

Result := N;

End;

Function InputElementsOfMatrixFromFile(Var InFile: TextFile; Var N: Integer)

: MatrixType;

Var

I: Integer;

J: Integer;

Matrix: MatrixType;

Begin

InitializationMatrix(Matrix, N);

For I := 0 To N Do

Begin

For J := 0 To N Do

Begin

If N <> -1 Then

Begin

Try

Read(InFile, Matrix[I][J]);

Except

Writeln('Invalid type! Try again.');

N := -1;

End;

End;

End;

End;

Result := Matrix;

End;

Function IsAnotherFile(): Boolean;

Var

IsCorrect: Boolean;

Choice: String;

Begin

IsCorrect := False;

Choice := '';

Writeln('Read error was detected in your file.', #13#10, 'This program can',

#39, 't continue to read this file.');

Repeat

Writeln('Do you want to change file?(yes/no)');

Readln(Choice);

If Choice = 'yes' Then

Begin

IsCorrect := True;

Result := True;

End

Else If Choice = 'no' Then

Begin

IsCorrect := True;

Result := False;

End

Else

Writeln('You input incorrect word! Try again.');

Until IsCorrect;

End;

Procedure Input(Var Matrix: MatrixType; Var N: Integer);

Var

FileName: String;

InFile: TextFile;

IsExit: Boolean;

Begin

IsExit := False;

FileName := '';

If InputMethod() Then

Begin

// input size of matrix

N := InputSizeOfMatrixFromConsole();

// input elements of matrix

Matrix := InputElementsOfMatrixFromConsole(N);

End

Else

Begin

Repeat

FileName := InputReadFileName();

AssignFile(InFile, FileName);

Reset(InFile);

N := InputSizeOfMatrixFromFile(InFile);

Matrix := InputElementsOfMatrixFromFile(InFile, N);

If N <> -1 Then

IsExit := True

Else If Not IsAnotherFile() Then

IsExit := True;

CloseFile(InFile);

Until IsExit;

End;

End;

Function FindBestSum(Var Matrix: MatrixType; Const N: Integer): Real;

Var

BestSum: Real;

I: Integer;

J: Integer;

Begin

BestSum := 0.0;

For I := 0 To N Do

Begin

For J := 0 To N Do

Begin

If (J < N) And ((Matrix[I][J] + Matrix[I][J + 1]) > BestSum) Then

BestSum := Matrix[I][J] + Matrix[I][J + 1];

If (I < N) And ((Matrix[I][J] + Matrix[I + 1][J]) > BestSum) Then

BestSum := Matrix[I][J] + Matrix[I + 1][J];

End;

End;

Result := BestSum;

End;

Function OutputMethod(): Boolean;

Var

IsCorrect: Boolean;

Choice: String;

Begin

IsCorrect := False;

Choice := '';

Writeln('The program works with console output or files.');

Repeat

Writeln('To use console enter ', #39, 'console', #39, '.', #13#10,

'To use a file enter ', #39, 'file', #39, '.'#13#10,

'Enter what type you want to use: ');

Readln(Choice);

If Choice = 'console' Then

Begin

IsCorrect := True;

Result := True;

End

Else If Choice = 'file' Then

Begin

IsCorrect := True;

Result := False;

End

Else

Writeln('The word ', Choice, ' don', #39,

't match any of method to output the data.');

Until IsCorrect;

End;

Procedure OutputConsole(Const Answer: Real);

Begin

Writeln('The answer is ', Answer:6:3, '.');

End;

Procedure OutputFile(Const Answer: Real);

Var

FileName: String;

OutFile: TextFile;

Begin

FileName := InputWriteFileName();

Assign(OutFile, FileName);

Rewrite(OutFile);

Writeln(OutFile, 'The answer is ', Answer:6:3, '.');

CloseFile(OutFile);

Writeln('Answer has been written down successful.');

End;

Procedure Output(Const Answer: Real);

Begin

If OutputMethod() Then

OutputConsole(Answer)

Else

OutputFile(Answer);

End;

Procedure Main();

Var

N: Integer;

Answer: Real;

Matrix: MatrixType;

Begin

N := 0;

Answer := 0.0;

Input(Matrix, N);

If N <> -1 Then

Begin

Answer := FindBestSum(Matrix, N);

Output(Answer);

End;

Free(Matrix);

End;

Begin

Main();

// freeze console

Writeln('Press enter to exit...');

Readln;

End.

**Код программы С++:**

#include<iostream>

#include<fstream>

#include<string>

#include<iomanip> // for std::setprecision()

const int MIN\_MATRIX\_ORDER = 2;

void initialization(double\*\*& matrix,const int n)

{

matrix = new double\* [n];

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

{

matrix[i] = new double[n];

}

}

void free(double\*\*& matrix,const int n)

{

if (matrix != nullptr)

{

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

{

delete[] matrix[i];

}

delete[] matrix;

matrix = nullptr;

}

}

bool inputMethod()

{

bool isIncorrect = true;

std::string choice = "\0";

// asking what the type user want to use

std::cout << "The program works with console input or files.\n";

do

{

std::cout << "To use console enter 'console'.\n"

<< "To use a file enter 'file'.\n"

<< "Enter what type you want to use: \n";

std::cin >> choice;

if (choice == "console")

{

return true;

}

else if (choice == "file")

{

isIncorrect = false;

}

else // wrong input

std::cerr << "The word '" << choice

<< "' don't match any of method to input the data.\n";

} while (isIncorrect);

return isIncorrect;

}

int inputSizeOfMatrixFromConsole()

{

bool isIncorrect = true;

int n = 0;

// asking for Size of matrix

do

{

std::cout << "Enter the matrix order:\n";

std::cin >> n;

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

{

std::cin.clear();

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

std::cerr << "Invalid type. Try again.\n";

}

else if (n < MIN\_MATRIX\_ORDER)

std::cerr << "Matrix order cannot be less than " << MIN\_MATRIX\_ORDER

<< ".\n";

else

isIncorrect = false;

} while (isIncorrect);

return n;

}

double\*\* inputElementsOfMatrixFromConsole(int& n)

{

bool isIncorrect = true;

double\*\* matrix = nullptr;

// initialization of matrix

initialization(matrix, n);

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

{

for (int j = 0; j < n; j++)

{

isIncorrect = true;

do

{

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

std::cin >> matrix[i][j];

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

{

std::cin.clear();

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

std::cerr << "Invalid type. Try again.\n";

}

else

isIncorrect = false;

} while (isIncorrect);

}

}

return matrix;

}

inline bool isFileExist(const std::string nameFile)

{

std::fstream inOut(nameFile);

if (inOut.is\_open())

{

inOut.close();

return true;

}

return false;

}

inline bool isFileReadable(const std::string nameFile)

{

std::ifstream in(nameFile);

if (in.good()) // if we can read from this file

{

in.close();

return true;

}

return false;

}

inline bool isFileWritable(const std::string nameFile)

{

std::ofstream out(nameFile);

if (out.good()) // if we can write in this file

{

out.close();

return true;

}

return false;

}

inline bool isTextFile(const std::string nameFile)

{

std::string type = nameFile.substr(nameFile.length() - (size\_t)(4));

return (type == ".txt") ? true : false;

}

bool isAnotherFile()

{

bool isIncorrect = true;

std::string userAnswer = "\0";

std::cout << "Read error was detected in your file.\n"

<< "This program can't continue to read this file.\n";

do

{

std::cout << "Do you want to change file?(yes/no)\n";

std::cin >> userAnswer;

if (userAnswer == "yes")

return true;

else if (userAnswer == "no")

isIncorrect = false;

else

std::cout << "You input incorrect word! Try again.\n";

} while (isIncorrect);

return false;

}

std::string inputReadFileName()

{

std::string fileName = "\0";

bool isIncorrect = true;

do

{

// Inputting name of file or path to the file including file

std::cout << "Enter the name of file in this directory or path to this file

including name of file:\n";

std::cin >> fileName;

if (!isFileExist(fileName)) // if file doesn't exist

{

std::cerr << "This file or the path to the file is specified incorrectly or

does not exist! Try again.\n";

}

else if (!isTextFile(fileName)) // if file isn't txt

{

std::cerr << "This file or path to the file isn't .txt! Try again.\n";

}

else if (!isFileReadable(fileName))

{

std::cerr << "The program can't read this file! Try again.\n";

}

else

isIncorrect = false; // to exit this loop

} while (isIncorrect);

return fileName;

}

std::string inputWriteFileName()

{

std::string fileName = "\0";

bool isIncorrect = true;

do

{

// Inputting name of file or path to the file including file

std::cout << "Enter the name of file in this directory or path to this file

including name of file:\n";

std::cin >> fileName;

if (!isFileExist(fileName)) // if file doesn't exist

{

std::cerr << "This file or the path to the file is specified incorrectly or

does not exist! Try again.\n";

}

else if (!isTextFile(fileName)) // if file isn't txt

{

std::cerr << "This file or path to the file isn't .txt! Try again.\n";

}

else if (!isFileWritable(fileName))

{

std::cerr << "The program can't write into this file! Try again.\n";

}

else

isIncorrect = false; // to exit this loop

} while (isIncorrect);

return fileName;

}

int inputSizeOfMatrixFromFile(std::ifstream& in)

{

int n;

in >> n;

if (in.fail())

{

in.clear();

std::cerr << "Invalid type. Check data in the file.\n";

n = 0;

}

else if (n < MIN\_MATRIX\_ORDER)

{

std::cerr << "Matrix order cannot be less than " << MIN\_MATRIX\_ORDER << ".\n";

n = 0;

}

return n;

}

double\*\* inputElementsOfMatrixFromFile(std::ifstream& in, int& n)

{

double\*\* matrix;

initialization(matrix,n);

bool isCorrect = true;

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

{

for (int j = 0; j < n; j++)

{

in >> matrix[i][j];

if(in.fail())

{

in.clear();

std::cerr << "Invalid type. Check data in the file.\n";

n = 0;

}

}

}

return matrix;

}

void input(double\*\*& matrix, int& n)

{

if (inputMethod())

{

// input size of matrix

n = inputSizeOfMatrixFromConsole();

// input elements of matrix

matrix = inputElementsOfMatrixFromConsole(n);

}

else

{

bool isNotExit = true;

std::string fileName = "\0";

std::ifstream in;

do

{

fileName = inputReadFileName();

in.open(fileName);

// input size of matrix

n = inputSizeOfMatrixFromFile(in);

// input elements of matrix

matrix = inputElementsOfMatrixFromFile(in, n);

if (n != 0)

{

isNotExit = false;

}

else if (!isAnotherFile())

{

isNotExit = false;

}

in.close();

} while (isNotExit);

}

}

bool outputMethod()

{

bool isIncorrect = true;

std::string choice = "\0";

// asking what the type user want to use

std::cout << "The program is ready to show answer.\n";

do

{

std::cout << "To output in console enter 'console'.\n"

<< "To output in a file enter 'file'.\n"

<< "Enter what type you want to use:\n";

std::cin >> choice;

if (choice == "console")

{

return true;

}

else if (choice == "file")

{

isIncorrect = false;

}

else // wrong input

std::cerr << "The word '" << choice

<< "' don't match any of method to output the data.\n";

} while (isIncorrect);

return false;

}

inline void outputConsole(const double answer)

{

std::cout << "The answer is " << answer << std::setprecision(3) << ".\n";

}

inline void outputFile(const double answer)

{

std::string fileName = inputWriteFileName();

std::ofstream out;

out.open(fileName);

out << "The answer is " << answer << std::setprecision(3) << ".\n";

out.close();

std::cout << "Answer has been written down successful.\n";

}

void output(const double answer)

{

outputMethod() ? outputConsole(answer) : outputFile(answer);

}

double findBestSum(double\*\*& matrix,const int n)

{

double bestSum = 0;

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

{

for (int j = 0; j < n; j++)

{

if (j + 1 < n && matrix[i][j] + matrix[i][j + 1] > bestSum)

bestSum = matrix[i][j] + matrix[i][j + 1];

if (i + 1 < n && matrix[i][j] + matrix[i + 1][j] > bestSum)

bestSum = matrix[i][j] + matrix[i + 1][j];

}

}

return bestSum;

}

int main()

{

int n = 0;

double answer = 0.0;

double\*\* matrix = nullptr;

input(matrix, n);

if (n != 0)

{

answer = findBestSum(matrix, n);

output(answer);

}

free(matrix, n);

return 0;

}

**Код программы Java:**

import java.io.\*;  
import java.util.Scanner;  
  
public class Main {  
 final static int minMatrixOrder = 2;  
  
 final static Scanner in = new Scanner(System.in);  
  
 public static double[][] initialization(int n)  
 {  
 return new double[n][n];  
 }  
  
 public static boolean inputMethod()  
 {  
 boolean isIncorrect = true;  
 String choice = "\0";  
 System.out.println("The program works with console input or files.");  
 do  
 {  
 System.out.print("""  
 To use console enter 'console'.  
 To use a file enter 'file'.  
 Enter what type you want to use:  
 """);  
 choice = in.nextLine();  
 if (choice.equals("console"))  
 {  
 return true;  
 }  
 else if (choice.equals("file"))  
 {  
 isIncorrect = false;  
 }  
 else // wrong input  
 System.err.println("The word " + choice + " don't match any of method

to input the data.");  
 } while(isIncorrect);  
 return isIncorrect;  
 }  
  
 public static int inputSizeOfMatrixFromConsole()  
 {  
 boolean isIncorrect = true;  
 int n = 0;  
 // asking for Size of matrix  
 do  
 {  
 try  
 {  
 System.out.println("Enter the matrix order: ");  
 n = Integer.parseInt(in.nextLine());  
 }  
 catch (NumberFormatException exception)  
 {  
 System.err.println("Invalid type! Try again.");  
 }  
 if (n < minMatrixOrder)  
 {  
 System.err.printf("Matrix order cannot be less than %d! Try again.\n",

minMatrixOrder);  
 }  
 else  
 isIncorrect = false;  
 } while (isIncorrect);  
 return n;  
 }  
  
 public static double[][] inputElementsOfMatrixFromConsole(int n)  
 {  
 double[][] matrix;  
 boolean isIncorrect = true;  
 // initialization of matrix  
 matrix = initialization(n);  
 for (int i = 0; i < matrix.length; i++)  
 {  
 for (int j = 0; j < matrix.length; j++)  
 {  
 isIncorrect = true;  
 do  
 {  
 try  
 {  
 System.out.printf("Enter a%d%d:\n",i+1,j+1);  
 matrix[i][j] = Double.parseDouble(in.nextLine());  
 isIncorrect = false;  
 }  
 catch (NumberFormatException exception) {  
 System.err.println("Invalid type! Try again.");  
 }  
 } while (isIncorrect);  
 }  
 }  
 return matrix;  
 }  
  
 public static String inputReadFileName()  
 {  
 String fileName = "\0";  
 boolean isIncorrect = true;  
 File tempFile;  
 do  
 {  
 // Inputting name of file or path to the file including file  
 System.out.println("Enter the name of file in this directory or path to

this file including name of file:");  
 fileName = in.nextLine();  
 tempFile = new File(fileName);  
 if (!tempFile.exists()) // if file doesn't exist  
 {  
 System.err.println("This file or the path to the file is specified

incorrectly or does not exist! Try again.");  
 }  
 else if (!fileName.endsWith(".txt")) // if file isn't .txt  
 {  
 System.err.println("This file or path to the file isn't .txt!

Try again.");  
 }  
 else if (!tempFile.canRead())  
 {  
 System.err.println("The program can't read this file! Try again.");  
 }  
 else  
 {  
 isIncorrect = false; // to exit this loop  
 }  
 }while(isIncorrect);  
 return fileName;  
 }  
  
 public static String inputWriteFileName()  
 {  
 String fileName = "\0";  
 boolean isIncorrect = true;  
 File tempFile;  
 do  
 {  
 // Inputting name of file or path to the file including file  
 System.out.println("Enter the name of file in this directory or path to

this file including name of file:");  
 fileName = in.nextLine();  
 tempFile = new File(fileName);  
 if (!tempFile.exists()) // if file doesn't exist  
 {  
 System.err.println("This file or the path to the file is specified

incorrectly or does not exist! Try again.");  
 }  
 else if (!fileName.endsWith(".txt")) // if file isn't .txt  
 {  
 System.err.println("This file or path to the file isn't .txt!

Try again.");  
 }  
 else if (!tempFile.canWrite())  
 {  
 System.err.println("The program can't read this file! Try again.");  
 }  
 else  
 {  
 isIncorrect = false; // to exit this loop  
 }  
 }while(isIncorrect);  
 return fileName;  
 }  
  
 public static Boolean isAnotherFile()  
 {  
 boolean isIncorrect = true;  
 String userAnswer = "\0";  
 System.out.println("Read error was detected in your file.\n" +  
 "This program can't continue to read this file.");  
 do  
 {  
 System.out.println("Do you want to change file?(yes/no)");  
 userAnswer = in.nextLine();  
 if (userAnswer.equals("yes"))  
 {  
 return true;  
 }  
 else if (userAnswer.equals("no"))  
 {  
 isIncorrect = false;  
 }  
 else  
 {  
 System.out.println("You input incorrect word! Try again.");  
 }  
 } while (isIncorrect);  
 return false;  
 }  
  
 public static int inputSizeOfMatrixFromFile(BufferedReader reader)

throws IOException  
 {  
 int n = 0;  
 char currentChar = '\0';  
 boolean wasNumber = false;  
 boolean isCorrect = false; // for exit the loop  
 String number = ""; // string for parse Number  
 int character = 0;  
 while(!isCorrect && (character = reader.read()) != -1)  
 {  
 currentChar = (char)(character);  
 if (Character.isAlphabetic(currentChar))  
 {  
 throw new IOException();  
 }  
 else if ((Character.isDigit(currentChar)))  
 {  
 number += currentChar;  
 wasNumber = true;  
 }  
 else if (wasNumber)  
 {  
 n = Integer.parseInt(number);  
 isCorrect = true;  
 }  
 }  
 if (n < minMatrixOrder && wasNumber)  
 {  
 System.err.printf("Matrix order cannot be less than %d.\n",

minMatrixOrder);  
 n = 0;  
 }  
 return n;  
 }  
  
 public static double[][] inputElementsOfMatrixFromFile (BufferedReader reader,

int n) throws IOException  
 {  
 boolean wasNumber = false;  
 boolean isCorrect = false;  
 double[][] matrix = null;  
 char currentChar = '\0';  
 String number = "";  
 int character = 0;  
 // initialization of matrix  
 matrix = initialization(n);  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < n; j++)  
 {  
 isCorrect = false;  
 wasNumber = false;  
 number = "";  
 while(!isCorrect && (character = reader.read()) != -1)  
 {  
 currentChar = (char)(character);  
 if (Character.isAlphabetic(currentChar))  
 {  
 throw new IOException();  
 }  
 else if ((Character.isDigit(currentChar)) || currentChar == '.')  
 {  
 number += currentChar;  
 wasNumber = true;  
 }  
 else if (wasNumber)  
 {  
 matrix[i][j] = Double.parseDouble(number);  
 isCorrect = true;  
 }  
 }  
 if (!number.isEmpty())  
 {  
 matrix[i][j] = Double.parseDouble(number);  
 }  
 }  
 }  
 return matrix;  
 }  
  
 public static double[][] input()  
 {  
 int n = 0; // size  
 double[][] matrix = null;  
 if (inputMethod())  
 {  
 // input size of matrix  
 n = inputSizeOfMatrixFromConsole();  
 // input elements of matrix  
 matrix = inputElementsOfMatrixFromConsole(n);  
 }  
 else  
 {  
 boolean isNotExit = true;  
 do  
 {  
 String fileName = inputReadFileName();  
 try (BufferedReader reader = new BufferedReader

(new FileReader(fileName)))  
 {  
 // input size of matrix  
 n = inputSizeOfMatrixFromFile(reader);  
 // input elements of matrix  
 matrix = inputElementsOfMatrixFromFile(reader,n);  
 }  
 catch(IOException ex)  
 {  
 System.err.println("Invalid type. Check data in the file.");  
 n = 0;  
 }  
 if (n != 0)  
 {  
 isNotExit = false;  
 }  
 else if (!isAnotherFile())  
 {  
 isNotExit = false;  
 }  
 } while(isNotExit);  
 }  
 return matrix;  
 }  
  
 public static boolean outputMethod()  
 {  
 boolean isIncorrect = true;  
 String choice = "";  
 System.out.println("The program is ready to show answer.");  
 do {  
 System.out.print("""  
 To output in console enter 'console'.  
 To output in a file enter 'file'.  
 Enter what type you want to use:  
 """);  
 choice = in.nextLine();  
 if (choice.equals("console"))  
 {  
 return true;  
 }  
 else if (choice.equals("file"))  
 {  
 isIncorrect = false;  
 }  
 else // wrong input  
 System.err.println("The word '" + choice +

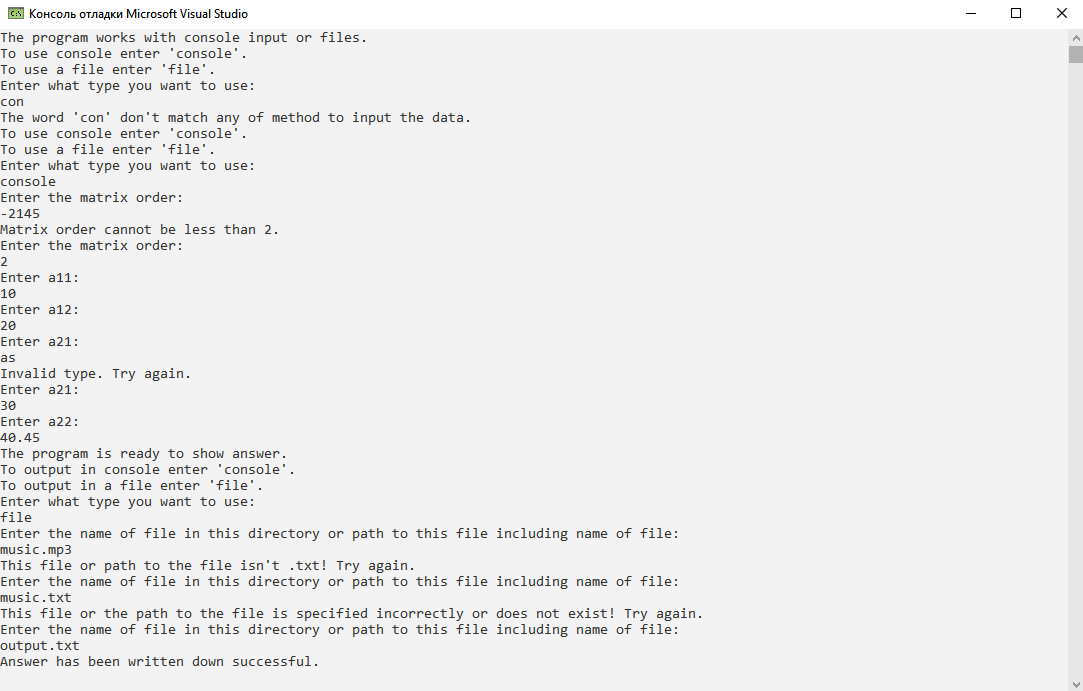
"' don't match any of method to output the data.");  
 } while(isIncorrect);  
 return false;  
 }  
  
 public static void outputConsole(final double answer)  
 {  
 System.out.printf("The answer is %6.3f.",answer);  
 }  
  
 public static void outputFile(final double answer)  
 {  
 String fileName = inputWriteFileName();  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName)))  
 {  
 writer.write("The answer is " + answer + ".\n");  
 }  
 catch (IOException ex)  
 {  
 System.err.println("Something went wrong!!!");  
 }  
 System.out.println("Answer has been written down successful.");  
 }  
  
 public static void output(final double answer)  
 {  
 if (outputMethod())  
 {  
 outputConsole(answer);  
 }  
 else  
 {  
 outputFile(answer);  
 }  
 }  
  
 public static double findBestSum(double[][] matrix)  
 {  
 double bestSum = 0;  
 for (int i = 0; i < matrix.length; i++)  
 {  
 for (int j = 0; j < matrix[0].length; j++)  
 {  
 if (j + 1 < matrix[0].length &&

matrix[i][j] + matrix[i][j + 1] > bestSum)  
 bestSum = matrix[i][j] + matrix[i][j + 1];  
 if (i + 1 < matrix[0].length &&

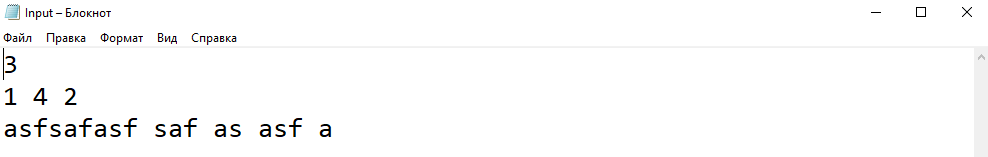
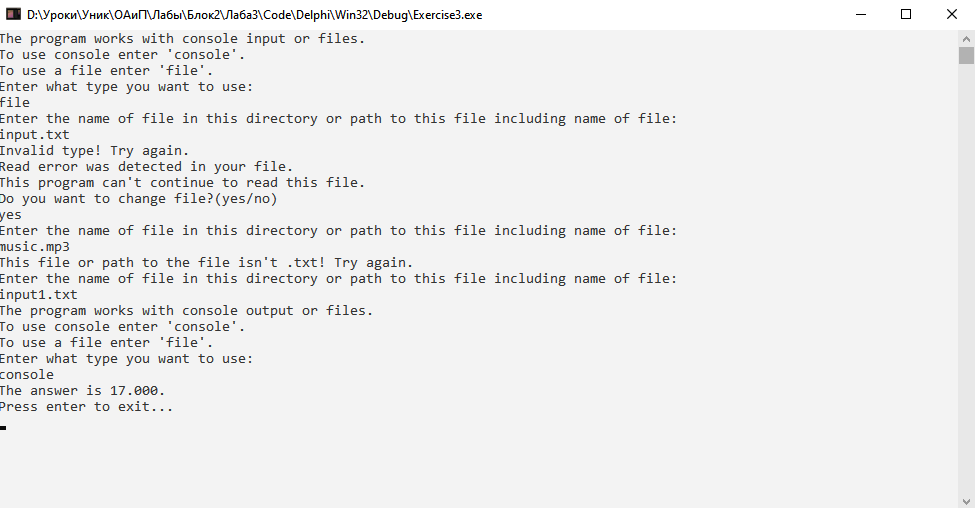
matrix[i][j] + matrix[i + 1][j] > bestSum)  
 bestSum = matrix[i][j] + matrix[i + 1][j];  
 }  
 }  
 return bestSum;  
 }  
  
 public static void main(String[] args)  
 {  
 double answer = 0.0;  
 double[][] matrix = null;  
 matrix = input();  
 if (matrix != null)  
 {  
 answer = findBestSum(matrix);  
 output(answer);  
 }  
 in.close();  
 }  
}

**Скриншоты:**

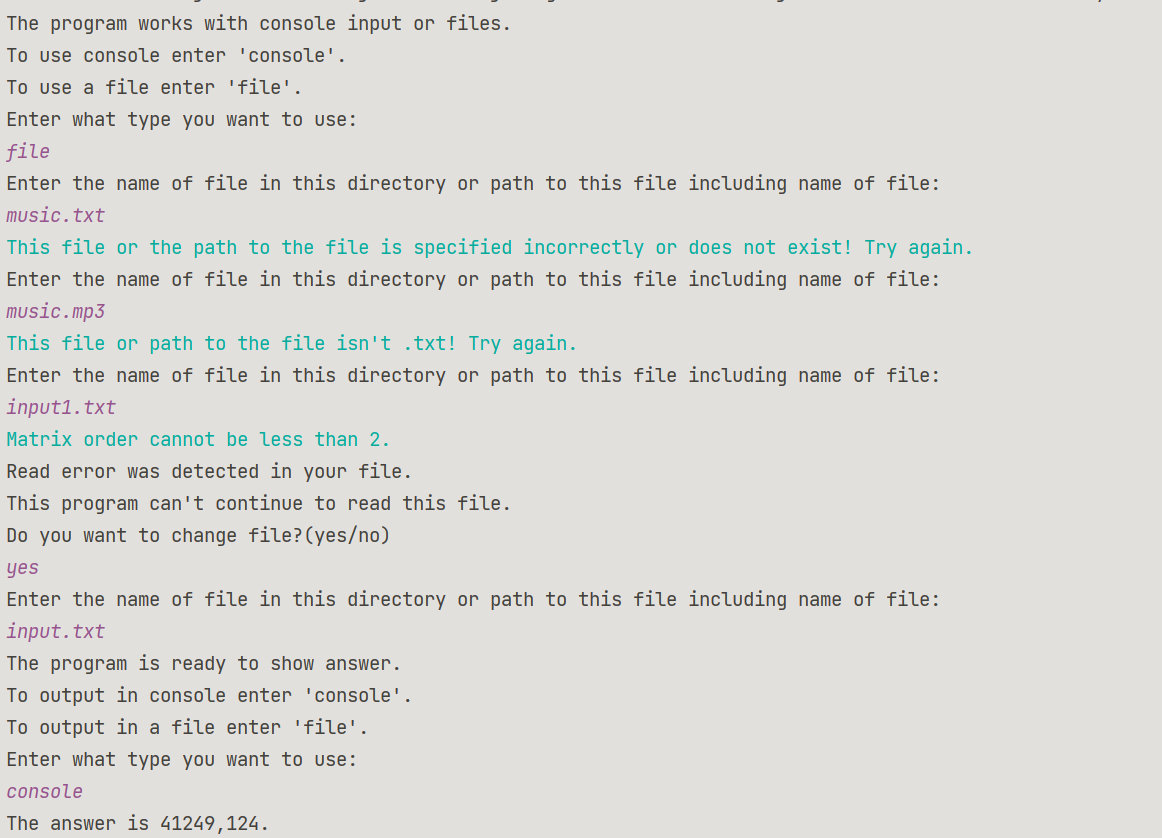
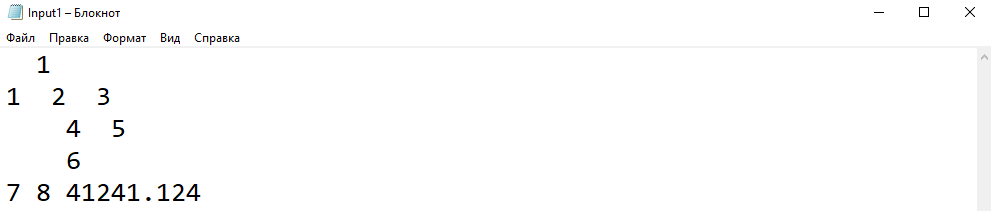
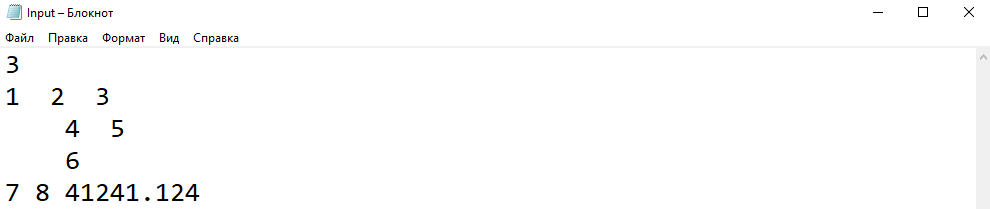
**C++:**



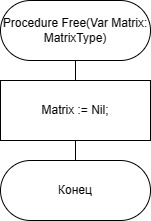
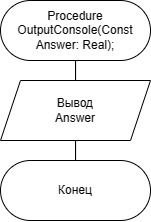
**Delphi:**

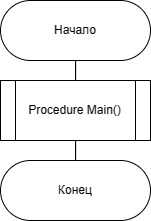
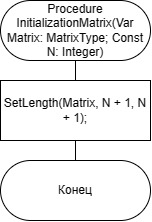


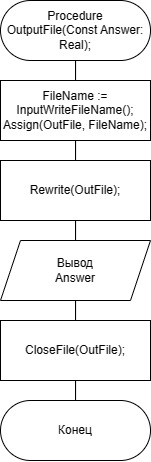
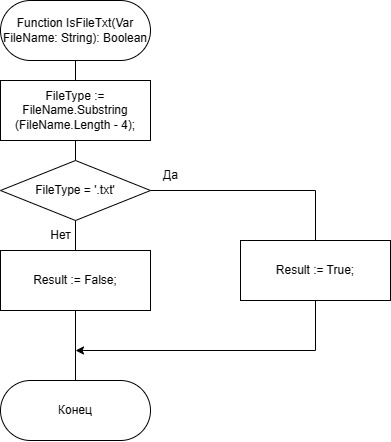
**Java:**

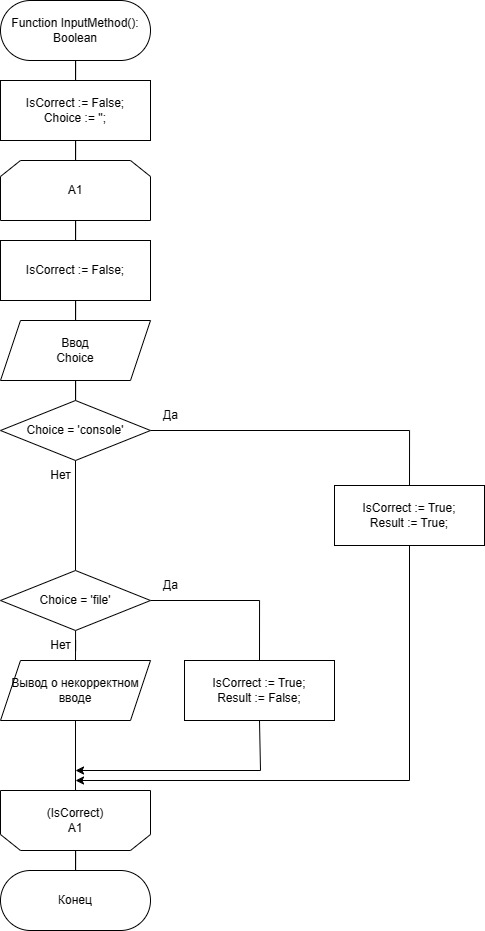
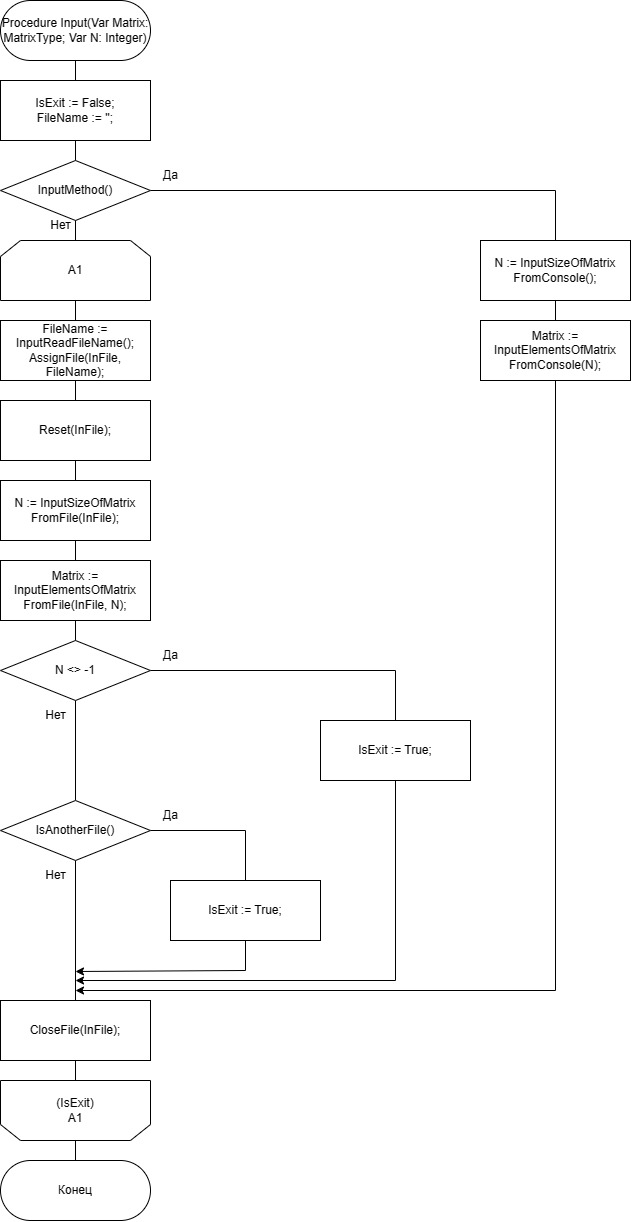
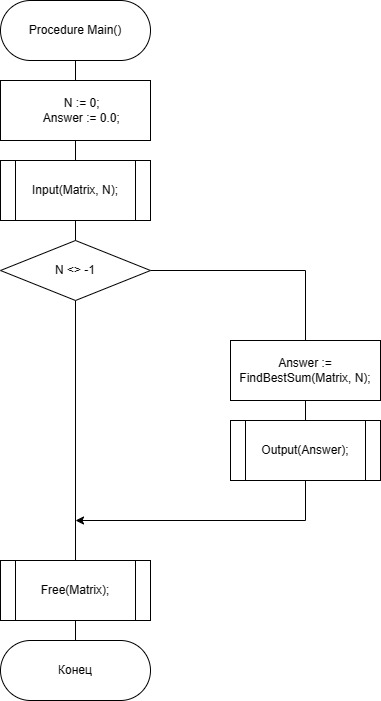


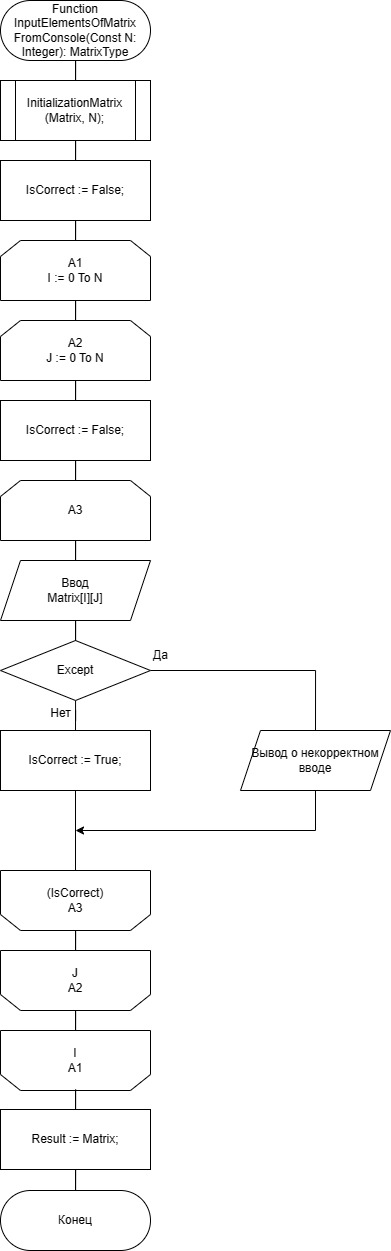
**Блок-схема:**

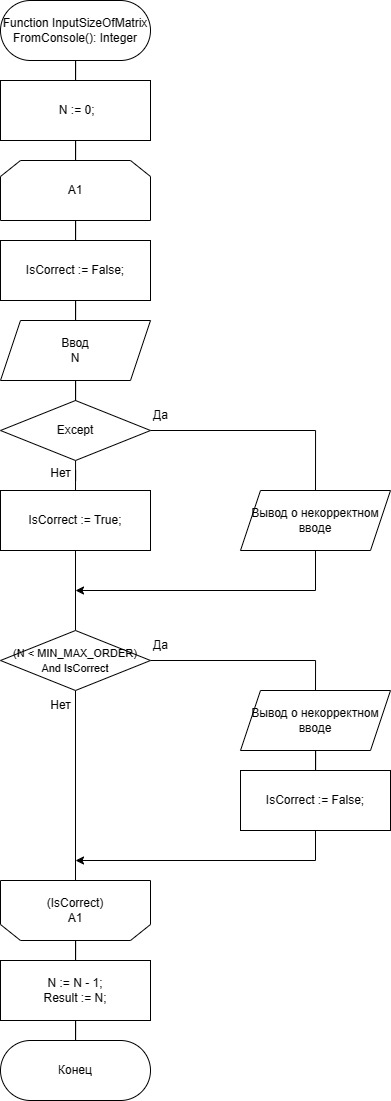
****

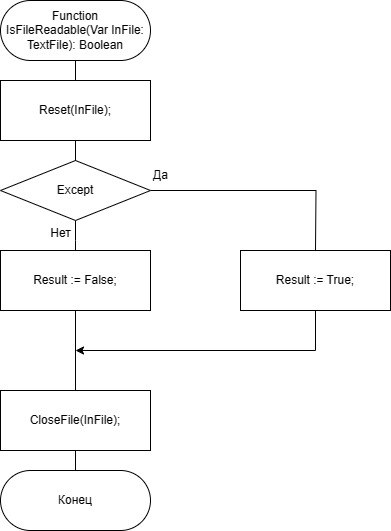


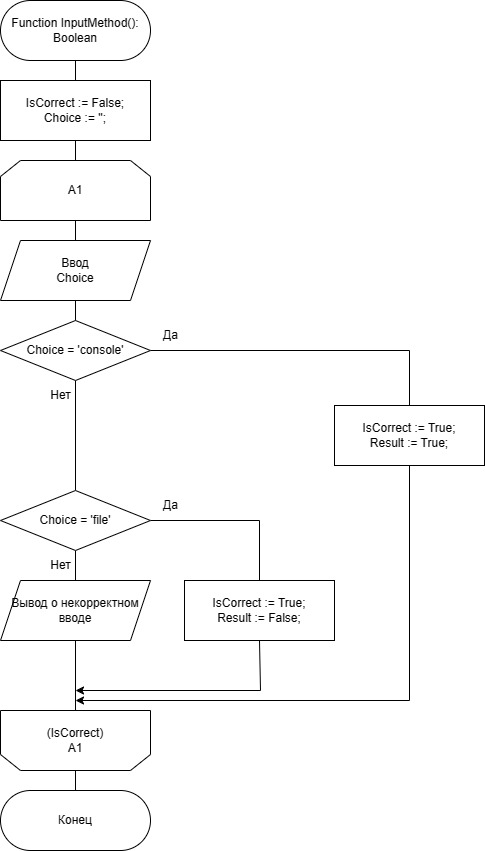
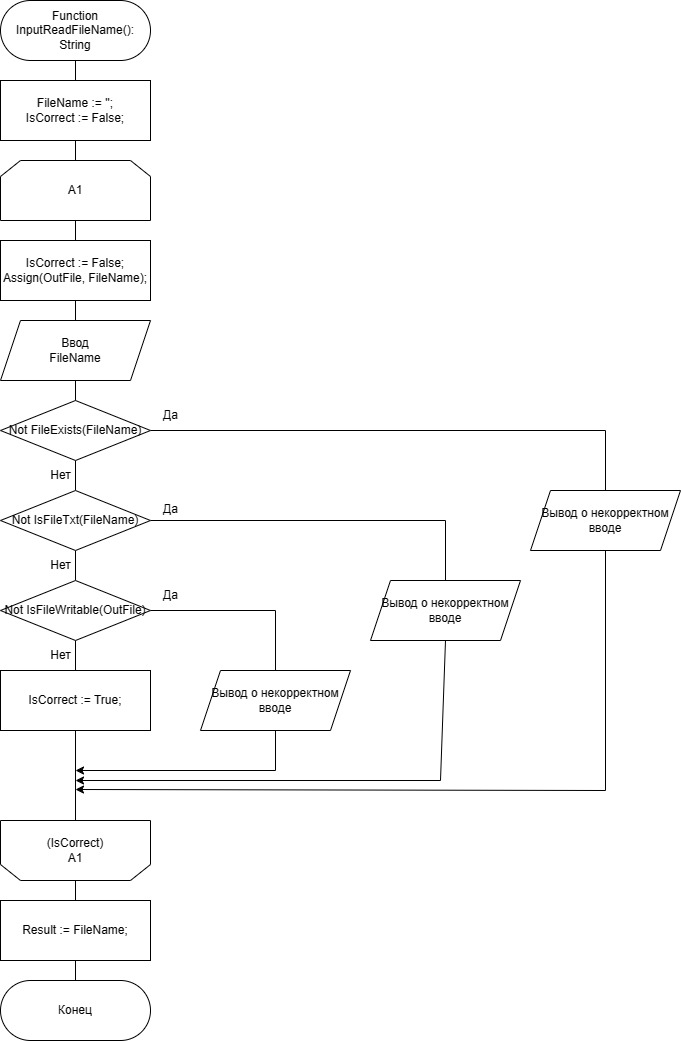
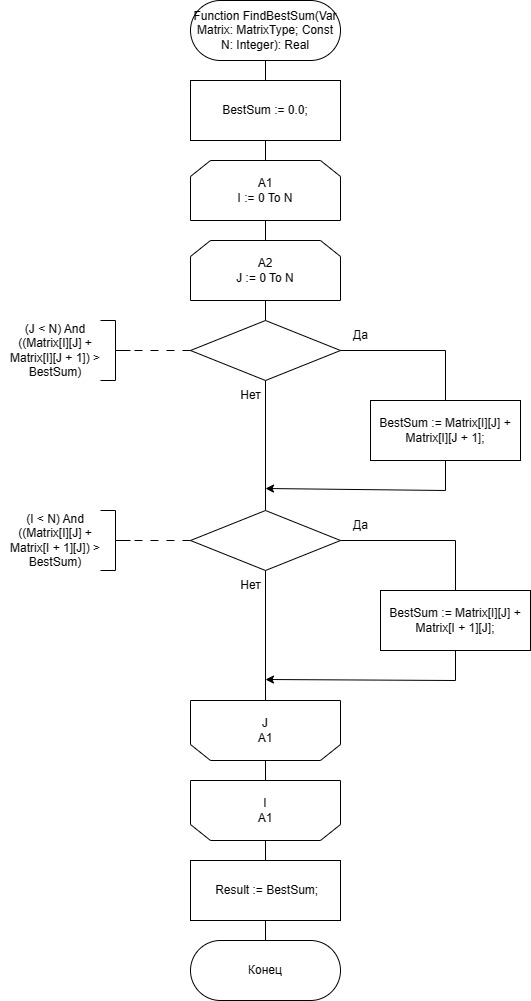
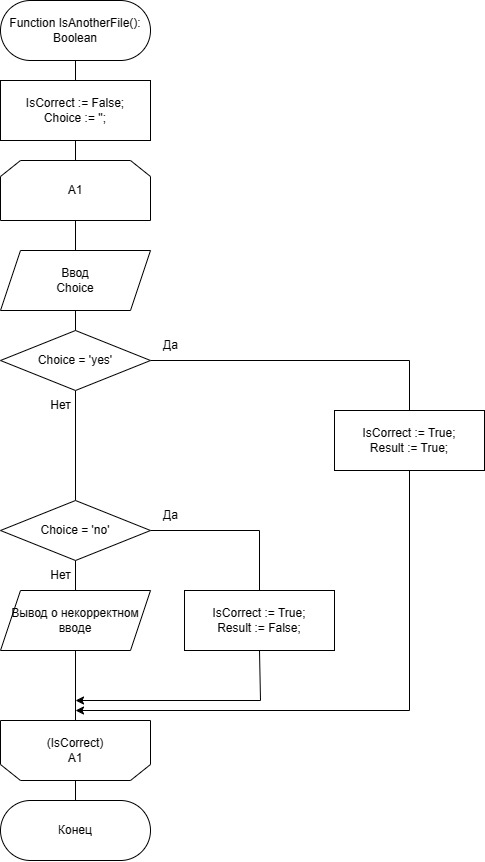
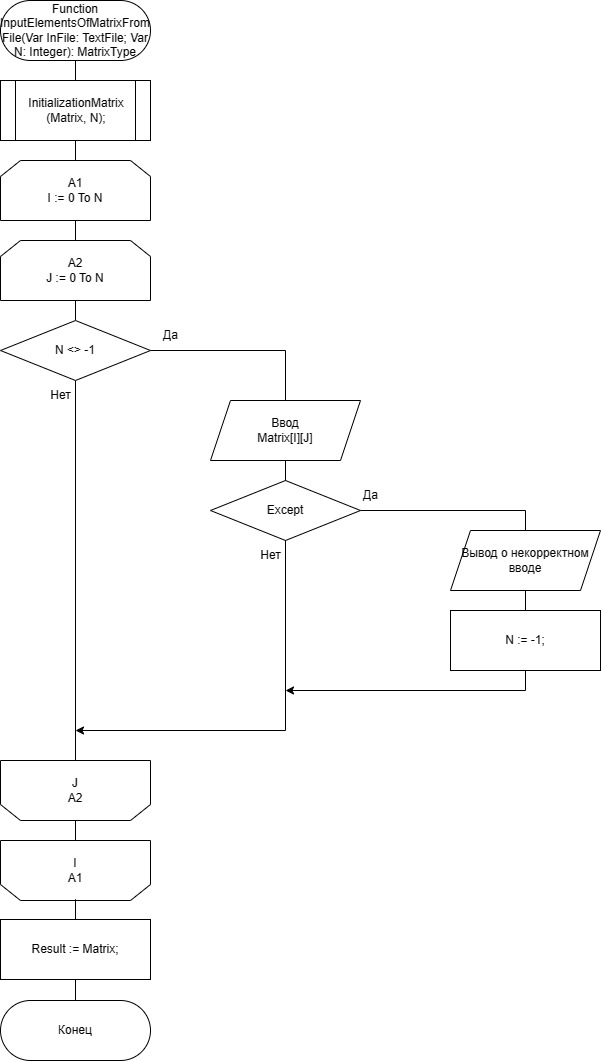
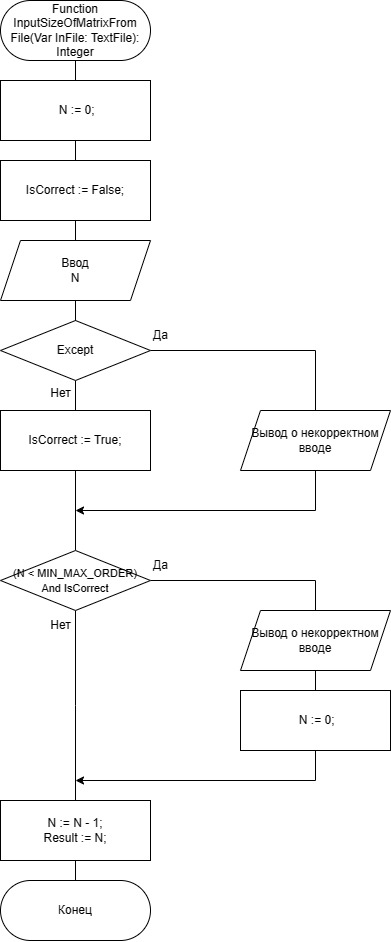
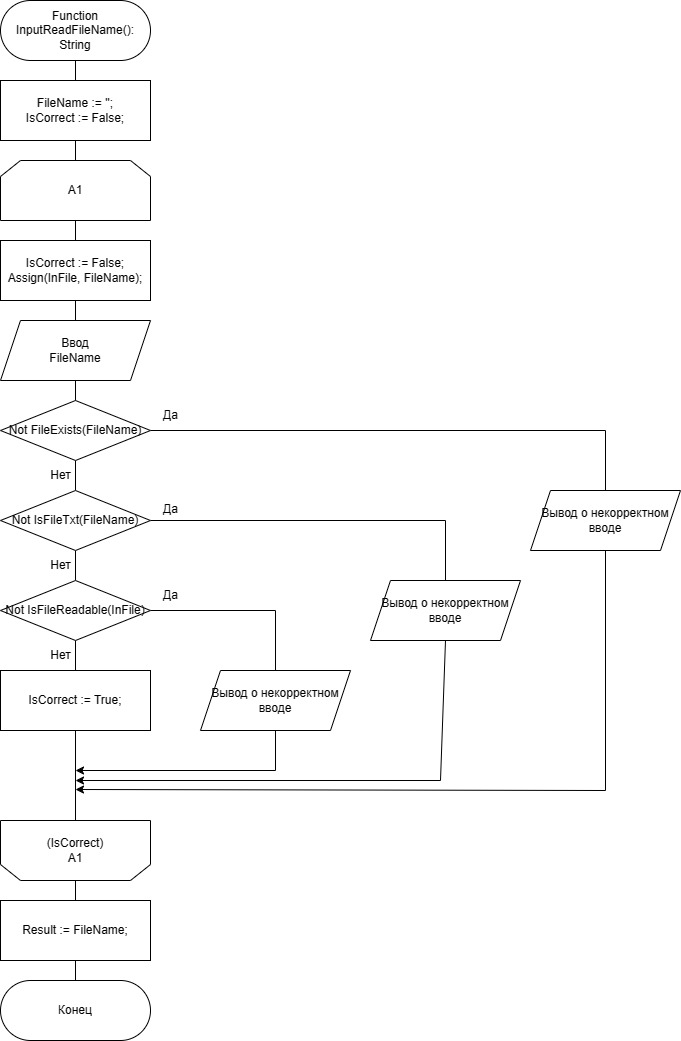
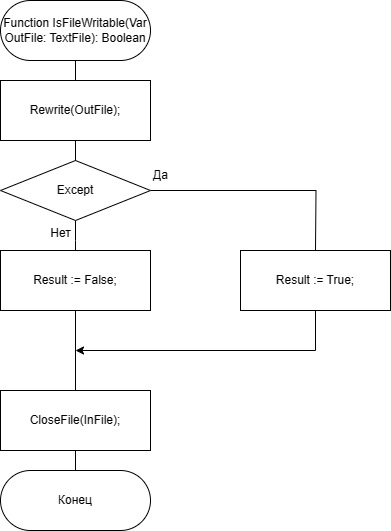
****

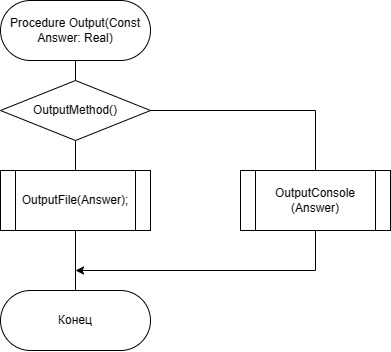


****



****



****

**Юнит Тесты (Unit Tests)**

**Код Unit Tests**

#include "pch.h"

#include "CppUnitTest.h"

#include "../Exercise3/Exercise3.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(TestMethod1)

{

const int SIZE = 3;

double\*\* matrix{ new double\* [SIZE] {

{new double[SIZE] {10.0,20.0,30.0}},

{new double[SIZE] {30.0,40.0,50.0}},

{new double[SIZE] {60.0,70.0,80.0}}

}};

Assert::AreEqual(findBestSum(matrix, SIZE), 150.0);

free(matrix, SIZE);

}

TEST\_METHOD(TestMethod2)

{

const int SIZE = 3;

double\*\* matrix{ new double\* [SIZE] {

{new double[SIZE] {50.0,40.0,50.0}},

{new double[SIZE] {90.0,120.0,30.0}},

{new double[SIZE] {30.0,200.0,15.0}}

} };

Assert::AreEqual(findBestSum(matrix, SIZE), 320.0);

free(matrix, SIZE);

}

TEST\_METHOD(TestMethod3)

{

const int SIZE = 3;

double\*\* matrix{ new double\* [SIZE] {

{new double[SIZE] {15.25, 17.35,13.46}},

{new double[SIZE] {35.00,-12.89,17.34}},

{new double[SIZE] {20.50,-232.0, 0.00}}

} };

Assert::AreEqual(findBestSum(matrix, SIZE), 55.50);

free(matrix, SIZE);

}

};

}

**Скриншоты:**

