УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.4

по предмету

Основы алгоритмизации и программирования

Вариант 3

Выполнил:

Бетеня К.С.

Проверила:

Данилова Г.В.

Группа 351005

Минск 2023

Задание:

Разработать алгоритмы и программы, реализующие эти алгоритмы. Основные функции программ оформить в виде процедур и функций. Исходные данные вводить из текстового файла. Результаты работы программы также поместить в текстовый файл и вывести на экран. Проверить является ли данная числовая последовательность а1, a2,..., an невозрастающей.

Код программы на **Delphi**:

Program labt4;

{$APPTYPE CONSOLE}

{$R \*.res}

Uses

System.SysUtils;

Const

CONS\_NUM: Integer = 1;

FILE\_NUM: Integer = 2;

MIN\_ARR\_SIZE: Integer = 2;

Procedure PrintStatement();

Begin

Writeln('The program calculates whether the entered', #13#10#9, 'natural

number sequence is increasing.', #13#10);

End;

//work with way to the file

Function PathCheck(Var IsIncorrect: Boolean): Integer;

Var

Path: Integer;

IsCorrect: boolean;

Begin

IsCorrect := false;

Try

Read(Path);

IsCorrect := true;

Except

Write('Invalid numeric input. Try again.', #13#10);

End;

If (Path <> CONS\_NUM) And (Path <> FILE\_NUM) And IsCorrect Then

Write('Choose only ', CONS\_NUM, ' or ', FILE\_NUM, ' Try again.', #13#10)

Else if IsCorrect then

IsIncorrect := True;

PathCheck := Path;

End;

Function ChoosingAPath(): Integer;

Var

Path: Integer;

IsIncorrect: Boolean;

Begin

Writeln('Where will we work throught: ', #13#10#9, 'Console: ', CONS\_NUM, #9,

'File: ', FILE\_NUM, #13#10);

IsIncorrect := False;

Repeat

Write('Your choise: ');

Path := PathCheck(IsIncorrect);

Until IsIncorrect;

ChoosingAPath := Path;

End;

//block of condition cheack

Function IsArrIncreasing(Var ArrOfNumb: Array Of Real): Boolean;

Var

I: Integer;

IsConditionYes: Boolean;

Begin

IsConditionYes := True;

For I := 1 To High(ArrOfNumb) Do

If ArrOfNumb[I] > ArrOfNumb[I - 1] Then

Else

IsConditionYes := False;

IsArrIncreasing := IsConditionYes;

End;

Function ResultOfArrChecking(IsIncreas: Boolean): Integer;

Begin

If IsIncreas Then

ResultOfArrChecking := 1

Else

ResultOfArrChecking := 0;

End;

///work in console

Function InputArrSize(): Integer;

Var

IsCorrect: Boolean;

ArrSize: Integer;

Begin

IsCorrect := False;

Repeat

Write('How much number in massive: ');

Try

Read(ArrSize);

Except

Writeln('Invalid numeric input. Try again.');

End;

If (ArrSize < MIN\_ARR\_SIZE) Then

Writeln('Min num is ', MIN\_ARR\_SIZE, '. Try again.')

Else

IsCorrect := True;

Until IsCorrect;

InputArrSize := ArrSize;

End;

Function EnteringTheCurrentNumber(I: Integer): Real;

Var

CurrentNum: Real;

IsCorrect: Boolean;

Begin

IsCorrect := False;

Repeat

Write('Write your ', I + 1, ' number: ');

Try

Read(CurrentNum);

IsCorrect := True;

Except

Writeln('Invalid numeric input. Try again.');

End;

Until IsCorrect;

EnteringTheCurrentNumber := CurrentNum;

End;

Procedure InputArr(Var ArrOfNumb: Array Of Real);

Var

I: Integer;

Begin

For I := 0 To High(ArrOfNumb) Do

ArrOfNumb[I] := EnteringTheCurrentNumber(I);

End;

Function ViaConsole(): Integer;

Var

ArrSize, Res: Integer;

ArrOfNumb: Array Of Real;

IsIncreasing: Boolean;

Begin

ArrSize := InputArrSize;

SetLength(ArrOfNumb, ArrSize);

InputArr(ArrOfNumb);

IsIncreasing := IsArrIncreasing(ArrOfNumb);

Res := ResultOfArrChecking(IsIncreasing);

ArrOfNumb := Nil;

ViaConsole := Res;

End;

///work with file

Procedure FileRestriction();

Begin

Writeln(#13#10, 'Rules for storing information in a file: ', #13#10#9, '1.

The first line contains an integer: ', #13#10#9#9,

'the number of array elements;', #13#10#9, '2. The second line is

real number', #13#10#9#9, 'entered separated by spaces.', #13#10);

End;

Procedure WayCondition(Way: String; Var IsCorrect: Boolean);

Begin

If ExtractFileExt(Way) <> '.txt' Then

Writeln('Write .txt file.')

Else

IsCorrect := True;

End;

Function InputWayToTheFile(): String;

Var

Way: String;

IsCorrect: Boolean;

Begin

IsCorrect := False;

Write('Write way to the file: ');

Readln;

Repeat

Read(Way);

WayCondition(Way, IsCorrect);

Readln;

Until IsCorrect;

InputWayToTheFile := Way;

End;

Function IsFileIntegrity(FileWay: String): Boolean;

Var

MyFile: TextFile;

IsIntegrity: Boolean;

Begin

IsIntegrity := False;

If FileExists(FileWay) Then

IsIntegrity := True;

IsFileIntegrity := IsIntegrity;

End;

Function WorkWithArr(ArrSize: Integer; Var ArrOfNumb: Array Of Real): Integer;

Var

IsIncreasin: Boolean;

Begin

IsIncreasin := IsArrIncreasing(ArrOfNumb);

WorkWithArr := ResultOfArrChecking(IsIncreasin);

End;

Function ResultOfReading(IsCorrect: Boolean; ArrSize: Integer;

Var ArrOfNumb: Array Of Real): Integer;

Var

Res: Integer;

Begin

If IsCorrect Then

Res := WorkWithArr(ArrSize, ArrOfNumb)

Else

Begin

Write('ERROR in file.');

Res := -1;

End;

ResultOfReading := Res;

End;

Function IsCorrectInputFromFile(Var MyFile: TextFile; Var ArrSize: Integer): Integer;

Var

IsCorrect: Boolean;

ArrOfNumb: Array Of Real;

I: Integer;

Begin

IsCorrect := True;

Try

Read(MyFile, ArrSize);

Except

IsCorrect := False;

End;

If (ArrSize < MIN\_ARR\_SIZE) And IsCorrect Then

IsCorrect := False;

If IsCorrect Then

Begin

SetLength(ArrOfNumb, ArrSize);

For I := 0 To High(ArrOfNumb) Do

Try

Read(MyFile, ArrOfNumb[I]);

Except

IsCorrect := False;

End;

End;

If SeekEof(MyFile) <> True Then

IsCorrect := False;

IsCorrectInputFromFile := ResultOfReading(IsCorrect, ArrSize, ArrOfNumb);

End;

Function IsReadingCorrect(FileWay: String; ArrSize: Integer): Integer;

Var

Res: Integer;

MyFile: TextFile;

Begin

AssignFile(MyFile, FileWay);

Try

Reset(MyFile);

Res := IsCorrectInputFromFile(MyFile, ArrSize);

Except

Begin

Write('Bad File.', #13#10);

Res := -1;

End;

End;

IsReadingCorrect := Res;

End;

Function WorkWithFile(FileWay: String): Integer;

Var

ArrSize, Res: Integer;

Begin

Res := IsReadingCorrect(FileWay, ArrSize);

WorkWithFile := Res;

End;

Function WorkWithIntergrityResoult(IsIntergrity: Boolean; FileWay: String): Integer;

Begin

If IsIntergrity Then

WorkWithIntergrityResoult := WorkWithFile(FileWay)

Else

Begin

Write('Bad File.');

WorkWithIntergrityResoult := -1;

End;

End;

Function ViaFile(): Integer;

Var

FileWay: String;

IsIntegrity: Boolean;

Begin

FileRestriction();

FileWay := InputWayToTheFile();

IsIntegrity := IsFileIntegrity(FileWay);

ViaFile := WorkWithIntergrityResoult(IsIntegrity, FileWay);

End;

///output console

Procedure OutputViaConsole(Result: Integer);

Begin

If Result = 1 Then

Write('Increase.')

Else

Write('Uncreased.');

End;

///output file

Function FileCorrectOutput(Res: Integer): String;

Begin

If (Res = 1) Then

FileCorrectOutput := 'Increase.'

Else

FileCorrectOutput := 'Uncreased.';

End;

Procedure OutputViaFile(Result: Integer);

Var

FileWay: String;

MyFile: TextFile;

Begin

FileWay := InputWayToTheFile();

AssignFile(MyFile, FileWay);

Try

Try

Reset(MyFile);

Append(MyFile);

Write(MyFile, FileCorrectOutput(Result));

Write('Cheack your file.');

Finally

CloseFile(MyFile);

End;

Except

Write(#13#10, 'Bad output file.');

End;

End;

Procedure Output(Option, Result: Integer);

Begin

If (Result <> -1) Then

Begin

Writeln(#13#10#10, 'You need to choose where to output the result.');

Option := ChoosingAPath();

If Option = FILE\_NUM Then

OutputViaFile(Result)

Else

OutputViaConsole(Result);

End;

End;

Var

Option, Result: Integer;

Begin

PrintStatement();

Option := ChoosingAPath();

If Option = FILE\_NUM Then

Result := ViaFile()

Else

Result := ViaConsole();

Output(Option, Result);

Readln;

Readln;

End.

Код программы на **C++**:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

const int CONS\_NUM = 1;

const int FILE\_NUM = 2;

const int MIN\_ARR\_SIZE = 2;

void printStatement()

{

cout << "The program calculates whether the entered\n\t"

"natural number sequence is increasing.\n\n";

}

// work with way to the file

int pathСheck(bool& isIncorrect)

{

int path;

cin >> path;

if (cin.fail() || cin.get() != '\n')

{

cerr << "Invalid numeric input. Try again.\n";

cin.clear();

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

}

else if (path != CONS\_NUM && path != FILE\_NUM)

cerr << "Choose only " << CONS\_NUM << " or "

<< FILE\_NUM << ". Try again.\n";

else

isIncorrect = false;

return path;

}

int choosingAPath()

{

cout << "Where will we work through: \n\tConsole: "

<< CONS\_NUM << "\tFile: " << FILE\_NUM << "\n\n";

int path;

bool isIncorrect = true;

do

{

cout << "Your choice: ";

path = pathСheck(isIncorrect);

} while (isIncorrect);

return path;

}

// block of conditon check

bool isArrIncreasing(double\*& arrOfNumb, int arrSize)

{

bool isConditionYes = true;

for (int i = 1; i < arrSize; i++)

if (arrOfNumb[i] > arrOfNumb[i - 1]);

else

isConditionYes = false;

return isConditionYes;

}

int resultOfArrChecking(bool isIncreas)

{

if (isIncreas)

return 1;

else

return 0;

}

/// work in console

int inputArrSize()

{

bool isIncorrect = true;

int arrSize;

do

{

cout << "How much number in massive: ";

cin >> arrSize;

if (cin.fail() || cin.get() != '\n')

{

cerr << "Invalid numeric input. Try again.\n";

cin.clear();

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

}

else if (arrSize < MIN\_ARR\_SIZE)

cerr << "Min num is "<< MIN\_ARR\_SIZE << ". Try again.\n";

else

isIncorrect = false;

} while (isIncorrect);

return arrSize;

}

double enteringTheCurrentNumber(int i)

{

bool isIncorrect = true;

double currentNum;

do

{

cout << "Write your " << i + 1 << " number: ";

cin >> currentNum;

if (cin.fail() || cin.get() != '\n')

{

cerr << "Invalid numeric input. Try again.\n";

cin.clear();

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

}

else

isIncorrect = false;

} while (isIncorrect);

return currentNum;

}

void inputArr(double\*& arrOfNumb, int arrSize)

{

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

arrOfNumb[i] = enteringTheCurrentNumber(i);

}

int viaConsole()

{

int arrSize = inputArrSize();

double\* arrOfNumb = new double[arrSize];

inputArr(arrOfNumb, arrSize);

bool isIncreasing = isArrIncreasing(arrOfNumb, arrSize);

int result = resultOfArrChecking(isIncreasing);

delete[] arrOfNumb;

arrOfNumb = nullptr;

return result;

}

/// work with file

void fileRestriction()

{

cout << "\nRules for storing information in a file:\n\t"

"1. The first line contains an integer: \n\t\t"

"the number of array elements;\n\t"

"2. The second line is real number\n\t\t"

"entered separated by spaces.\n\n";

}

void wayCondition(string way, bool& isIncorrect)

{

if (way.size() > 4)

{

string bufstr = way.substr(way.size() - 4);

if (bufstr == ".txt")

isIncorrect = false;

else

cerr << "Write .txt file.\n";

}

else

cerr << "The path is too short.\n";

}

string inputWayToTheFile()

{

string way;

bool isIncorrect = true;

cout << "Write way to your file: ";

do

{

cin >> way;

wayCondition(way, isIncorrect);

} while (isIncorrect);

return way;

}

bool isFileIntegrity(string fileWay)

{

bool isIntegrity = false;

fstream file;

file.open(fileWay);

if (file.is\_open())

isIntegrity = true;

file.clear();

file.close();

return isIntegrity;

}

int workWithArr(int arrSize, double\*& arrOfNumb)

{

bool isIncreasin = isArrIncreasing(arrOfNumb, arrSize);

return resultOfArrChecking(isIncreasin);

}

int resultOfReading(bool isCorrect, int arrSize, double\*& arrOfNumb)

{

if (isCorrect){

int res = workWithArr(arrSize, arrOfNumb);

delete[] arrOfNumb;

arrOfNumb = nullptr;

return res;

}

else {

cerr << "ERROR in file.";

return -1;

}

}

int isCorrectInputFromFile(ifstream& file, int& arrSize, double\*& arrOfNumb)

{

bool isCorrect = true;

file >> arrSize;

if (file.fail() || file.get() != '\n')

isCorrect = false;

if (arrSize < MIN\_ARR\_SIZE && isCorrect)

isCorrect = false;

if (isCorrect) {

arrOfNumb = new double[arrSize];

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

{

file >> arrOfNumb[i];

if (file.fail())

isCorrect = false;

}

}

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

isCorrect = false;

return resultOfReading(isCorrect, arrSize, arrOfNumb);

}

int isReadingCorrect(string fileWay, int arrSize, double\*& arrOfNumb)

{

ifstream file(fileWay, ios::in);

int result = isCorrectInputFromFile(file, arrSize, arrOfNumb);

file.clear();

file.close();

return result;

}

int workWithFile(string fileWay)

{

int arrSize = 0;

double\* arrOfNumb;

int result = isReadingCorrect(fileWay, arrSize, arrOfNumb);

return result;

}

int workWithIntergrityResoult(bool isIntegrity, string fileWay)

{

if (isIntegrity)

return workWithFile(fileWay);

else

{

cerr << "Bad File.";

return -1;

}

}

int viaFile()

{

fileRestriction();

string fileWay = inputWayToTheFile();

bool isIntegrity = isFileIntegrity(fileWay);

return workWithIntergrityResoult(isIntegrity, fileWay);

}

/// output console

void outputViaConsole(int result)

{

if (result)

cout << "Increase.";

else

cout << "Uncreased.";

}

/// output file

string fileCorrectOutput(int result)

{

if (result)

return "\nIncrease.";

else

return "\nUncreased.";

}

void outputViaFile(int result)

{

string fileWay = inputWayToTheFile();

ofstream file(fileWay, ios::app);

if (file.is\_open())

{

file << fileCorrectOutput(result);

cout << "Check your file.";

}

else

cerr << "\nBad output file.";

file.clear();

file.close();

}

/// output

void output(int option, int result)

{

if (result != -1)

{

cout << "\n\nYou need to choose where to output the result.\n";

option = choosingAPath();

option == FILE\_NUM ? outputViaFile(result) :

outputViaConsole(result);

}

}

/// main block

int main()

{

printStatement();

int option = choosingAPath();

int result = option == FILE\_NUM ? viaFile() : viaConsole();

output(option, result);

return 0;

}

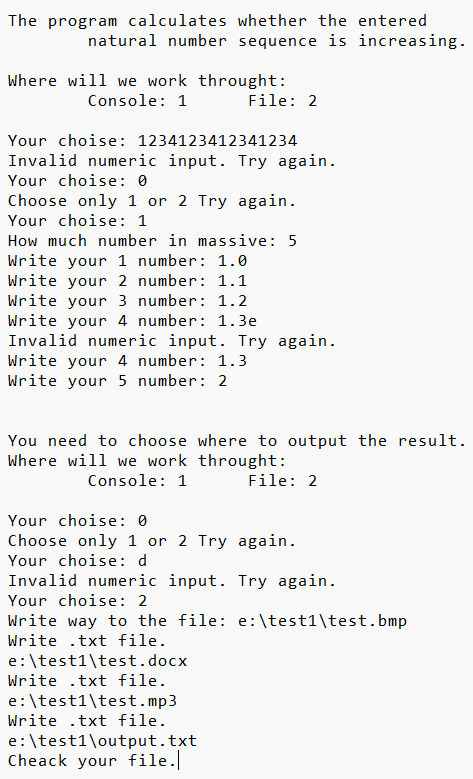
Код программы на **Java**:

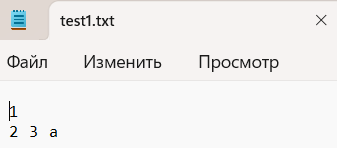
package lab2;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.util.Scanner;  
  
class SaveData{  
 private boolean isCorrect;  
 private int arrSize;  
  
 public SaveData(boolean isCorrect, int arrSize){  
 this.isCorrect = isCorrect;  
 this.arrSize = arrSize;  
 }  
  
 public boolean isCorrect() {  
 return isCorrect;  
 }  
  
 public void setCorrect(boolean correct) {  
 isCorrect = correct;  
 }  
  
 public int getArrSize() {  
 return arrSize;  
 }  
  
 public void setArrSize(int arrSize) {  
 this.arrSize = arrSize;  
 }  
}  
  
public class lab4 {  
 static final int *CONS\_NUM* = 1;  
 static final int *FILE\_NUM* = 2;  
 static final int *MIN\_ARR\_SIZE* = 2;  
  
 static void printStatement(){  
 System.*out*.print("""  
 The program calculates whether the entered  
 natural number sequence is increasing.  
 """);  
 }  
  
  
 // work with way to the file  
 static int pathCheck(SaveData SD, Scanner in){  
 int path = 0;  
 try{  
 path = Integer.*parseInt*(in.nextLine());  
 } catch(Exception error){  
 System.*err*.print("Invalid numeric input. Try again.\n");  
 }  
 if (path != *CONS\_NUM* && path != *FILE\_NUM*)  
 System.*out*.printf("Choose only %d or %d. Try again.\n", *CONS\_NUM*,

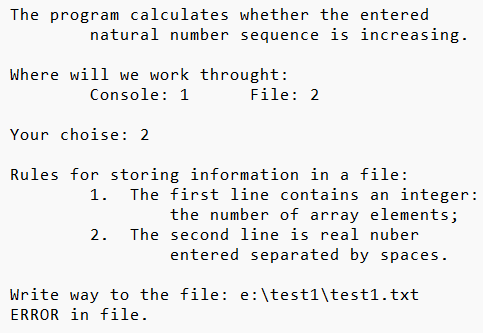
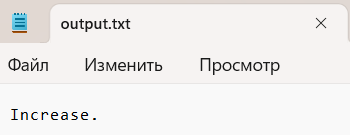
*FILE\_NUM*);  
 else  
 SD.setCorrect(false);  
  
 return path;  
 }  
  
  
 static int choosingAPath(Scanner in){  
 System.*out*.printf("Where will we work through: \n\t\tConsole: " +  
 "%d\t\tFile: %d\n\n", *CONS\_NUM*, *FILE\_NUM*);  
  
 int path = 0;  
 SaveData SD = new SaveData(true, 0);  
 do{  
 System.*out*.print("Your choice: ");  
 path = *pathCheck*(SD, in);  
 } while (SD.isCorrect());  
  
 return path;  
 }  
  
  
 // block of condition check  
 static boolean isArrIncreasing(double[] arrOfNumb, SaveData SD){  
 boolean isConditionYes = true;  
 for (int i = 1; i < SD.getArrSize(); i++) {  
 if (arrOfNumb[i] > arrOfNumb[i - 1]);  
 else  
 isConditionYes = false;  
 }  
  
 return isConditionYes;  
 }  
  
  
 static int resultOfArrChecking(boolean isIncreas){  
 if (isIncreas)  
 return 1;  
 else  
 return 0;  
 }  
  
  
 /// work in console  
 static int inputArrSize(Scanner in){  
 boolean isIncorrect = true;  
 int arrSize = 0;  
  
 do {  
 System.*out*.print("How much number in massive: ");  
 try {  
 arrSize = Integer.*parseInt*(in.nextLine());  
 } catch(Exception error){  
 System.*err*.print("Invalid numeric input. Try again.\n");  
 }  
 if (arrSize < *MIN\_ARR\_SIZE*)  
 System.*err*.printf("Min num is %d. Try again.\n", *MIN\_ARR\_SIZE*);  
 else  
 isIncorrect = false;  
 } while (isIncorrect);  
  
 return arrSize;  
 }  
  
  
 static double enteringTheCurrentNumber(int i, Scanner in){  
 boolean isIncorret = true;  
 double currentNum = 0.0;  
  
 do{  
 System.*out*.printf("Write your %d number: ", i + 1);  
 try{  
 currentNum = Double.*parseDouble*(in.nextLine());  
 isIncorret = false;  
 } catch(Exception error){  
 System.*err*.print("Invalid numeric input. Try again.\n");  
 }  
 } while (isIncorret);  
  
 return currentNum;  
 }  
  
  
 static void inputArr(double[] arrOfNumb, int arrSize, Scanner in){  
 for (int i = 0; i < arrSize; i++){  
 arrOfNumb[i] = *enteringTheCurrentNumber*(i, in);  
 }  
 }  
  
  
 static int viaConsole(Scanner in){  
 SaveData SD = new SaveData(true, 0);  
 SD.setArrSize(*inputArrSize*(in));  
 double[] arrOfNumb = new double[SD.getArrSize()];  
  
 *inputArr*(arrOfNumb, SD.getArrSize(), in);  
 boolean isIncreasing = *isArrIncreasing*(arrOfNumb, SD);  
  
 return *resultOfArrChecking*(isIncreasing);  
 }  
  
  
 /// work with file  
 static void fileRestriction(){  
 System.*out*.print("""  
 \nRules for storing information in a file:  
 \t1. The first line contains an integer:  
 \t\tthe number of array elements;  
 \t2. The second line is real number  
 \t\tentered separated by spaces.  
   
 """);  
 }  
  
  
 static void wayCondition(String way, SaveData SD){  
 if (way.length() > 4) {  
 String bufstr = way.substring(way.length() - 4);  
 if (bufstr.equals(".txt"))  
 SD.setCorrect(false);  
 else  
 System.*err*.print("Write .txt file.\n");  
 }  
 else  
 System.*err*.print("The path is too short.\n");  
 }  
  
  
 static String inputWayToTheFile(Scanner in){  
 String way = "";  
 SaveData SD = new SaveData(true, 0);  
  
 System.*out*.print("Write way to your file: ");  
 do{  
 way = in.nextLine();  
  
 *wayCondition*(way, SD);  
 } while (SD.isCorrect());  
  
 return way;  
 }  
  
  
 static boolean isFileIntegrity(String fileWay)  
 {  
 boolean isIntegrity = false;  
  
 File file = new File(fileWay);  
 if (file.canRead())  
 isIntegrity = true;  
  
 return isIntegrity;  
 }  
  
 static int workWithArr(SaveData SD, double[] arrOfNumb) {  
 boolean isIncreasin = *isArrIncreasing*(arrOfNumb, SD);  
 return *resultOfArrChecking*(isIncreasin);  
 }  
  
 static int resultOfReading(boolean isCorrect, SaveData SD, double[] arrOfNumb)  
 {  
 if (isCorrect)  
 return *workWithArr*(SD, arrOfNumb);  
 else {  
 System.*err*.print("ERROR in file.");  
  
 return -1;  
 }  
 }  
  
  
 static int isCorrectInputFromFile(Scanner in,SaveData SD){  
 boolean isCorrect = true;  
 double[] arrOfNumb = new double[0];  
 if (!in.hasNextInt())  
 isCorrect = false;  
 else {  
 int arrSize = in.nextInt();  
 SD.setArrSize(arrSize);  
 if (arrSize < *MIN\_ARR\_SIZE*)  
 isCorrect = false;  
  
 arrOfNumb = new double[arrSize];  
 for (int i = 0; i < arrSize; i++) {  
 if (!in.hasNextDouble())  
 isCorrect = false;  
  
 arrOfNumb[i] = in.nextDouble();  
 }  
  
 if (in.hasNext())  
 isCorrect = false;  
 }  
 return *resultOfReading* (isCorrect, SD, arrOfNumb);  
 }  
  
  
 static int isReadingCorrect(String fileWay, SaveData SD) {  
 try {  
 Scanner fileScanner = new Scanner(new File(fileWay));  
 return *isCorrectInputFromFile*(fileScanner, SD);  
 } catch(FileNotFoundException error) {  
 System.*out*.print("File not found.\n");  
 return -1;  
 }  
 }  
  
 static int workWithFile(String fileWay) {  
 SaveData SD = new SaveData(true, 0);  
  
 return *isReadingCorrect*(fileWay, SD);  
 }  
  
  
 static int workWithIntergrityResoult(boolean isIntegrity, String fileWay) {  
 if (isIntegrity)  
 return *workWithFile*(fileWay);  
 else  
 {  
 System.*out*.print("Bad File.");  
  
 return -1;  
 }  
 }  
  
  
 static int viaFile(Scanner in){  
 *fileRestriction*();  
  
 String fileWay = *inputWayToTheFile*(in);  
 boolean isIntergrity = *isFileIntegrity*(fileWay);  
  
 return *workWithIntergrityResoult*(isIntergrity, fileWay);  
 }  
  
  
 /// output console  
 static void outputViaConsole(int result){  
 if (result == 1)  
 System.*out*.print("Increasing.");  
 else  
 System.*out*.print("Uncreased.");  
 }  
  
  
 // output file  
 static String fileCorrectOutput(int result){  
 if (result == 1)  
 return "\nIncrease.";  
 else  
 return "\nUncreased.";  
 }  
  
  
 static void outputViaFile(int result, Scanner in)  
 {  
 String fileWay = *inputWayToTheFile*(in);  
 File file = new File(fileWay);  
 StringBuilder builder;  
 if (file.canWrite())  
 {  
 try{  
 FileWriter writer = new FileWriter(fileWay);  
 builder = new StringBuilder();  
 builder.append(*fileCorrectOutput*(result));  
 writer.write(builder.toString());  
 writer.close();  
 System.*out*.print("Check your file.");  
 } catch (Exception error){  
 System.*err*.print("Can't write in this file.");  
 }  
 }  
 else  
 System.*err*.print("\nBad output file.");  
 }  
  
  
 /// output  
 static void output(int option, int result, Scanner in)  
 {  
 if (result != -1)  
 {  
 System.*out*.print("\n\nYou need to choose where to output the

result.\n");  
 option = *choosingAPath*(in);  
  
 if (option == *FILE\_NUM*) {  
 *outputViaFile*(result, in);  
 } else {  
 *outputViaConsole*(result);  
 }  
 }  
 }  
  
  
 public static void main(String[] args){  
 Scanner in = new Scanner(System.*in*);  
  
 *printStatement*();  
  
 int option = *choosingAPath*(in);  
 int result = option == *FILE\_NUM* ? *viaFile*(in) : *viaConsole*(in);  
  
 *output*(option, result, in);  
 }  
}

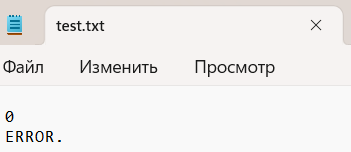
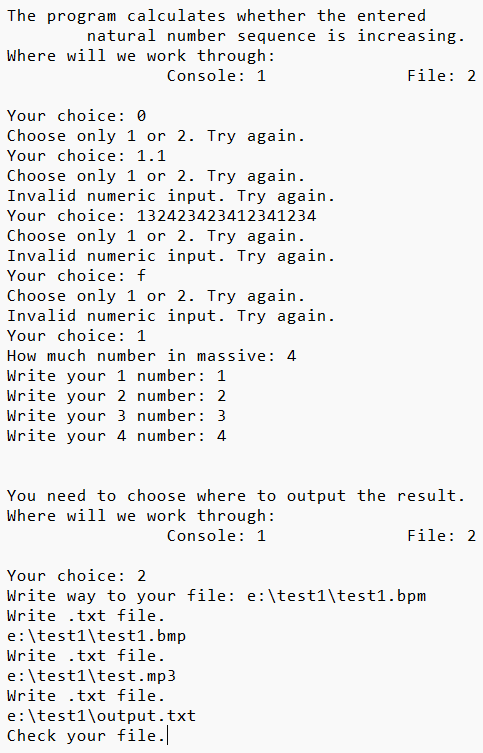
Результат на **Delphi**:



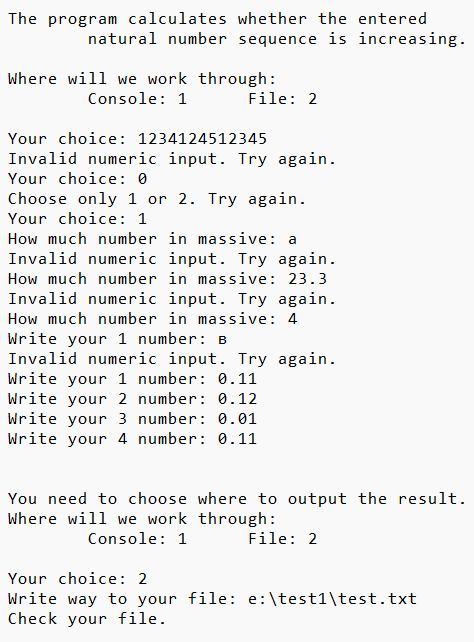


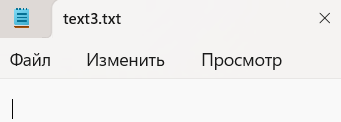


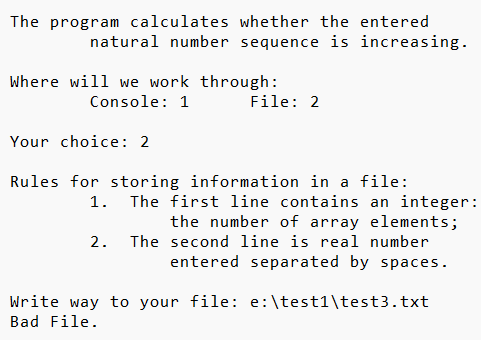
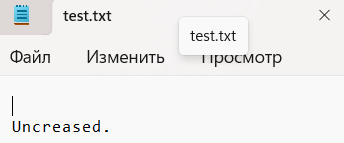
Результат на **Java**:



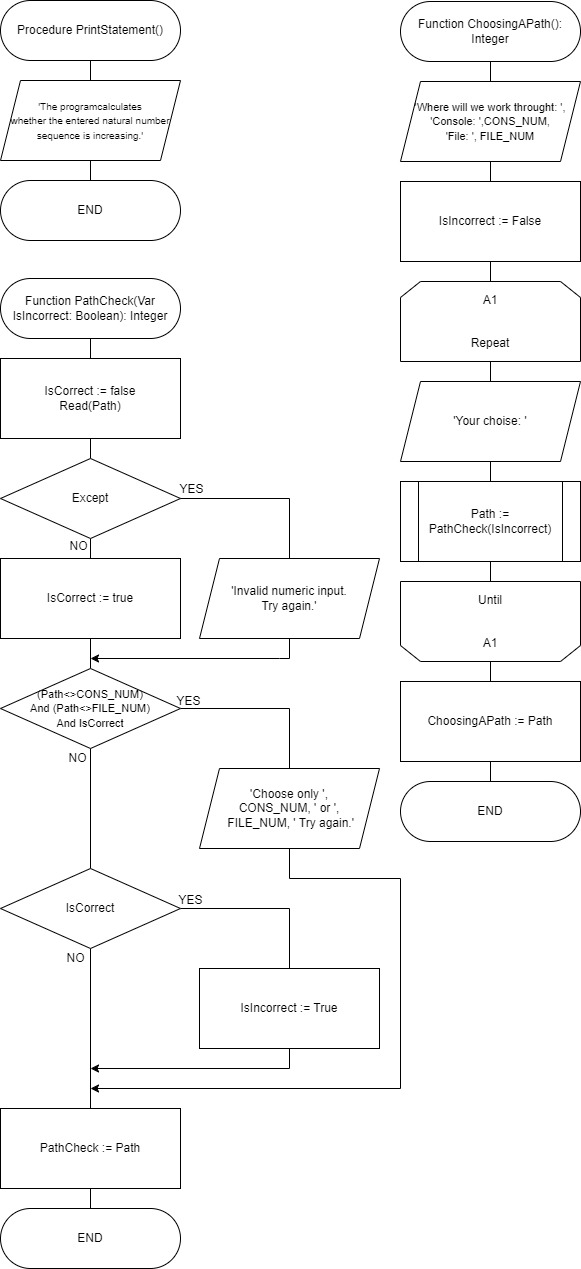
Результат на **C++**:

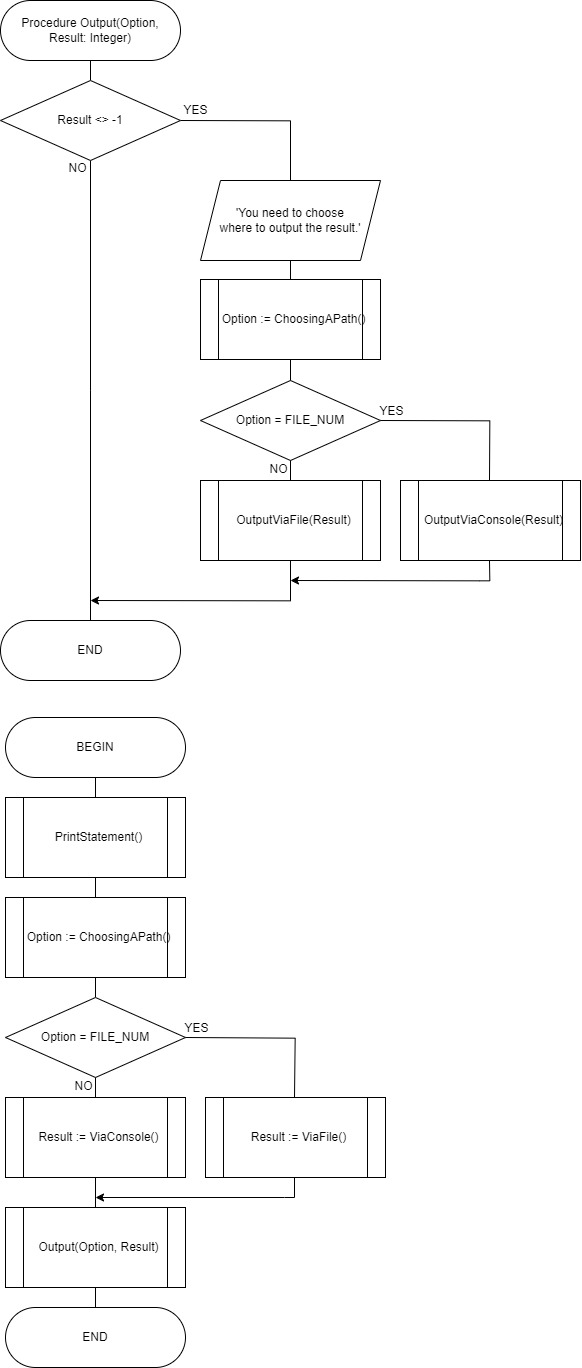
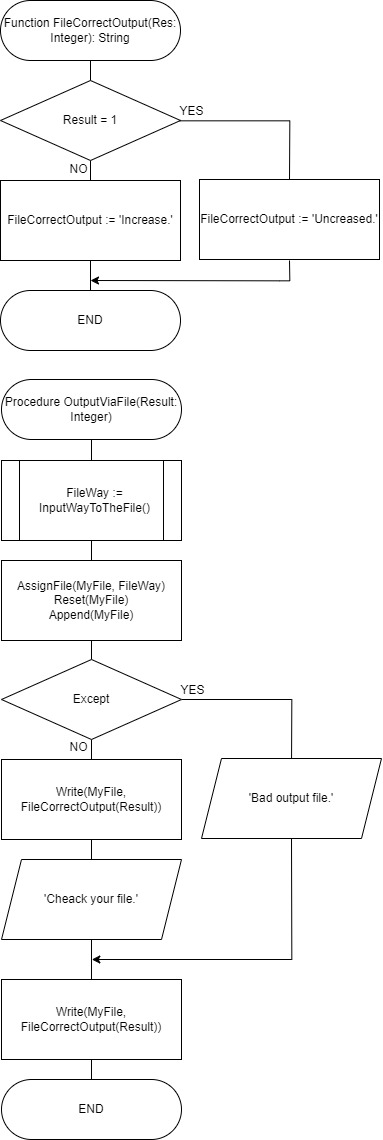
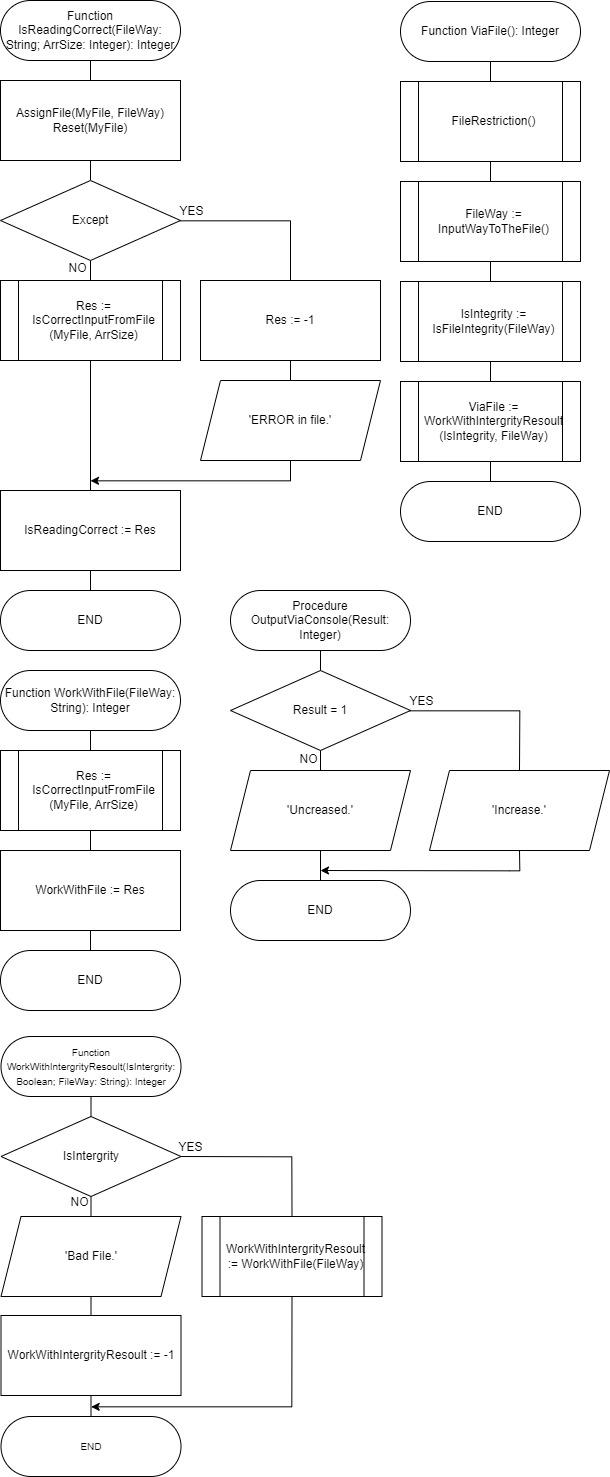
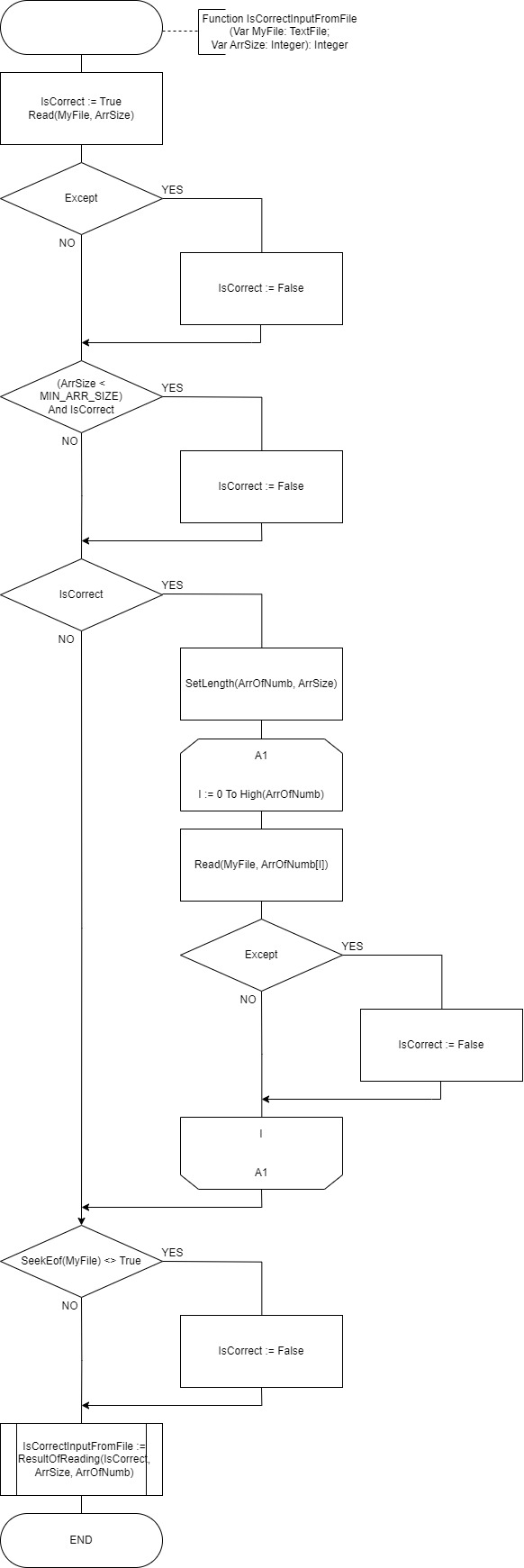
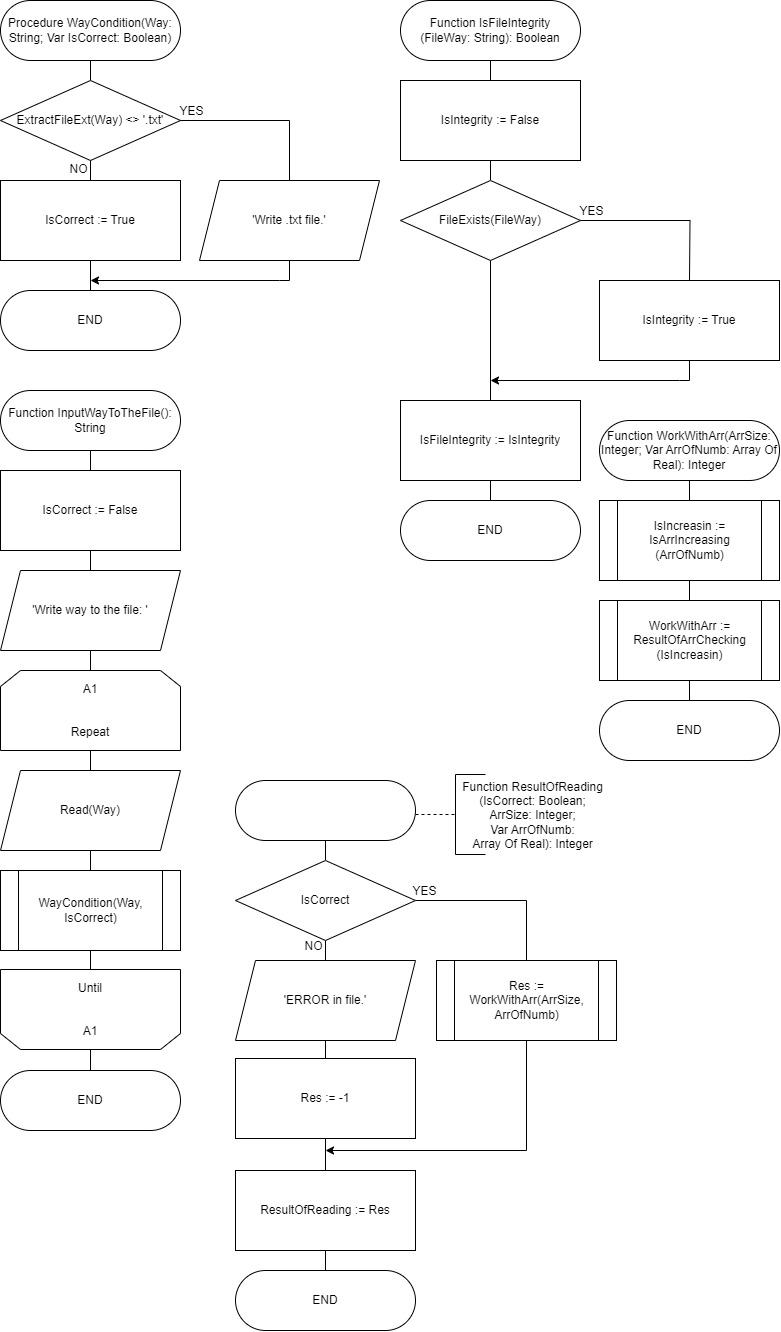
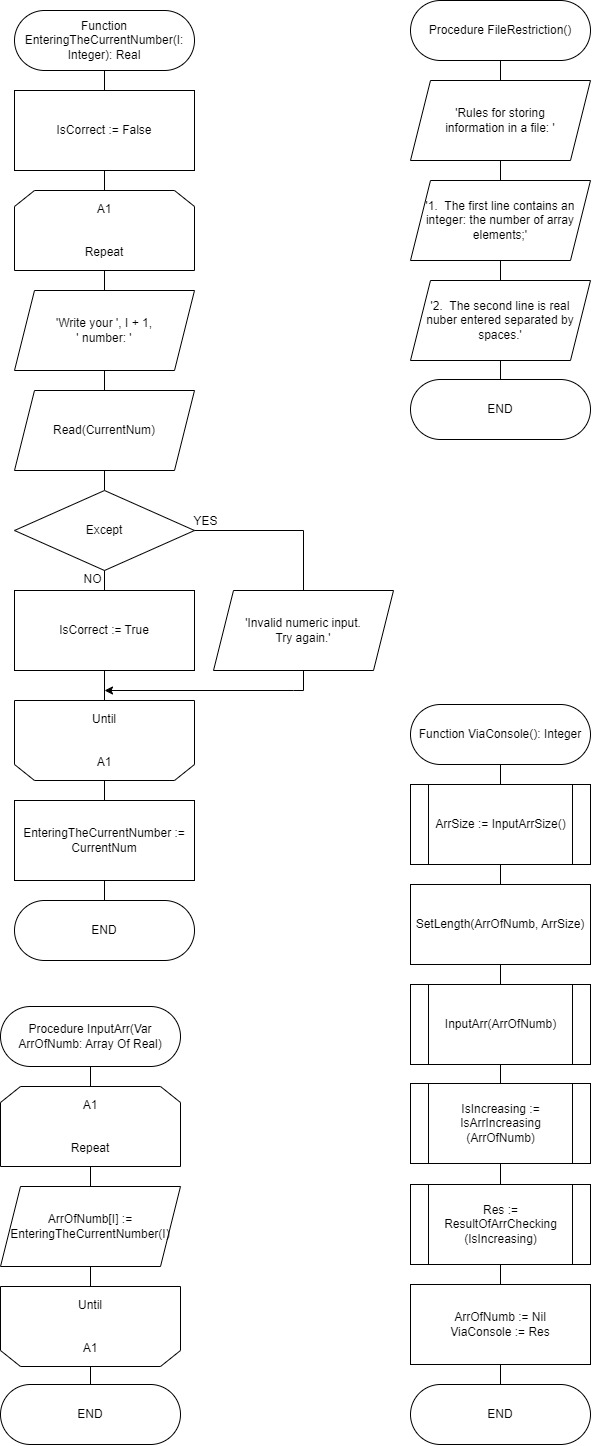
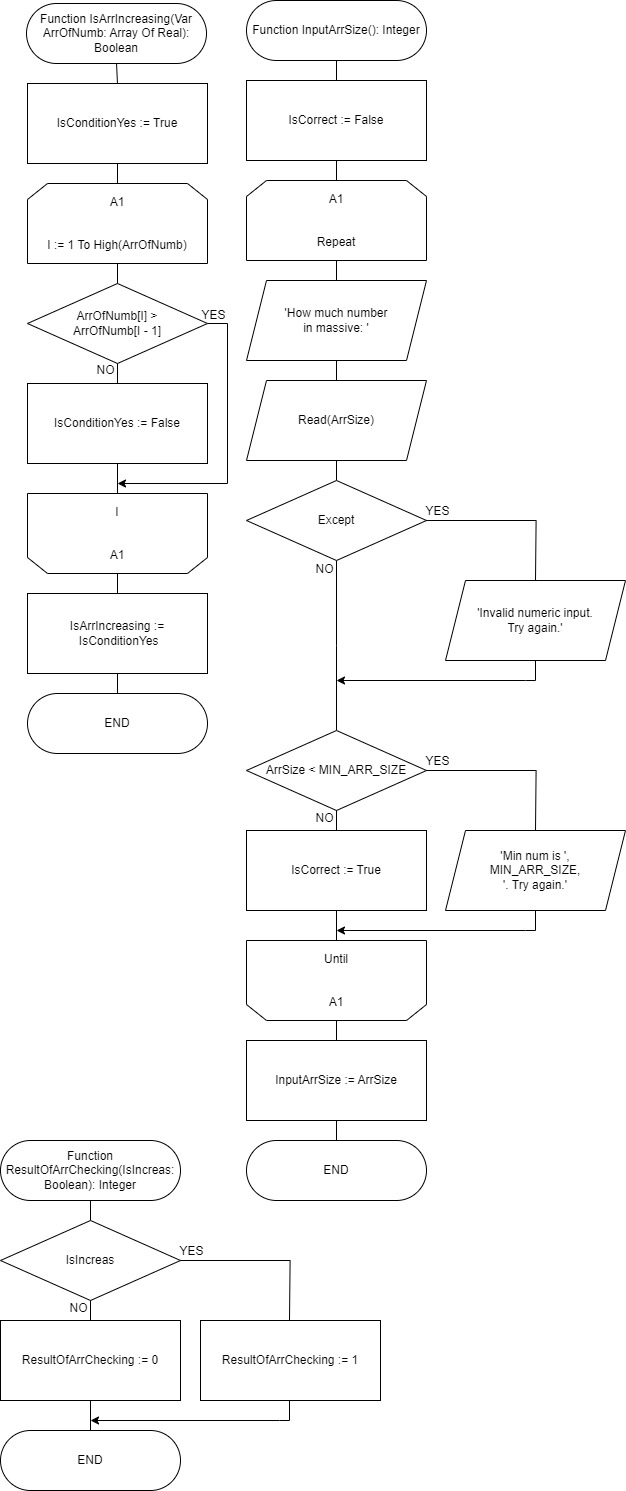






Блок-схема:





**Юнит Тесты (Unit Tests)**

**Код Unit Tests**

#include "pch.h"

#include "CppUnitTest.h"

#include "..\..\lab4.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(TestMethod1)

{

int arrSize = 3;

double\* arrOfNumb = new double[0.0, 0.0, 0.0];

Assert::AreEqual(isArrIncreasing(arrOfNumb, arrSize), false);

}

TEST\_METHOD(TestMethod2)

{

int arrSize = 3;

double\* arrOfNumb = new double[0.000010, 0.000011, 0.000009];

Assert::AreEqual(isArrIncreasing(arrOfNumb, arrSize), false);

}

TEST\_METHOD(TestMethod3)

{

int arrSize = 3;

double\* arrOfNumb = new double[1E+1 + 1,1E+1 + 2,1E+1 - 1];

Assert::AreEqual(isArrIncreasing(arrOfNumb, arrSize), false);

}

};

}

**Скриншоты:**

