УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.4

по предмету «Основы алгоритмизации и программирования»

Вариант 5

Выполнил:

Бражалович А. И.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Проверить является ли данная числовая последовательность а1, a2,..., an невозрастающей.

**Код программы Delphi:**

Program Lab2\_4;

Uses

System.SysUtils;

Type

TArr = Array Of Integer;

Const

MIN\_LENGTH = 2;

MAX\_LENGTH = 1000;

MIN\_ELEMENT = -10000;

MAX\_ELEMENT = 10000;

Procedure PrintTask();

Begin

WriteLn('Данная программа проверяет невозрастающая ли последовательность:' + #13#10);

End;

Function ChooseFileInput() : Boolean;

Var

IsFileInput: Integer;

IsCorrect, Choose: Boolean;

Begin

IsFileInput := 0;

IsCorrect := False;

Choose := False;

Repeat

WriteLn('Вы хотите вводить последовательность через файл? (Да - ', 1, ' / Нет - ', 0, ')');

Try

ReadLn(IsFileInput);

IsCorrect := True;

Except

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

If IsCorrect Then

Begin

If IsFileInput = 1 Then

Choose := True

Else If IsFileInput = 0 Then

Choose := False

Else

Begin

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

End;

Until IsCorrect;

ChooseFileInput := Choose;

End;

Function CheckArea(Num: Integer; Const MIN, MAX: Integer) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

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

Begin

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

IsCorrect := False;

End;

CheckArea := IsCorrect;

End;

Function ReadPathToFile() : String;

Var

PathToFile: String;

IsCorrect: Boolean;

Begin

PathToFile := '';

IsCorrect := False;

Repeat

Write('Введите путь к файлу с расширением .txt с последовательностью, у которой количество членов должно быть не меньше ', MIN\_LENGTH, ' и не больше ', MAX\_LENGTH, ' а элементы должны быть в диапазоне [', MIN\_ELEMENT, ':', MAX\_ELEMENT,']: ');

ReadLn(PathToFile);

If ExtractFileExt(PathToFile) = '.txt' Then

IsCorrect := True

Else

Begin

WriteLn('Расширение файла не .txt!');

IsCorrect := False;

End;

Until IsCorrect;

ReadPathToFile := PathToFile;

End;

Function IsExists(PathToFile: String) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := False;

If FileExists(PathToFile) Then

IsCorrect := True;

IsExists := IsCorrect;

End;

Function IsAbleToReading(Var F: TextFile) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

Try

Reset(F);

CloseFile(F);

Except

IsCorrect := False;

End;

IsAbleToReading := IsCorrect;

End;

Function IsAbleToWriting(PathToFile: String) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

If FileIsReadOnly(PathToFile) Then

IsCorrect := False;

IsAbleToWriting := IsCorrect;

End;

Function IsEmpty(Var F: TextFile) : Boolean;

Var

Size: Integer;

IsCorrect: Boolean;

Begin

Size := 0;

IsCorrect := False;

Reset(F);

If Not EOF(F) Then

Size := 1;

CloseFile(F);

If Size = 0 Then

IsCorrect := True;

IsEmpty := IsCorrect;

End;

Function IsRightFileNums(Var F: TextFile) : Boolean;

Var

Buf: Char;

NumOfElements, Element: Integer;

IsCorrect: Boolean;

Begin

Buf := ' ';

NumOfElements := 0;

Element := 0;

IsCorrect := True;

Reset(F);

Try

Read(F, NumOfElements);

Except

IsCorrect := False;

End;

ReadLn(F, Buf);

If Buf <> #13 Then

IsCorrect := False;

If IsCorrect Then

IsCorrect := CheckArea(NumOfElements, MIN\_LENGTH, MAX\_LENGTH);

While IsCorrect And Not EOF(F) Do

Begin

While IsCorrect And Not EOLN(F) Do

Begin

Try

Read(F, Element);

Except

IsCorrect := False;

End;

If IsCorrect Then

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

End;

ReadLn(F);

End;

CloseFile(F);

IsRightFileNums := IsCorrect;

End;

Function IsOrdersEqual(Var F: TextFile) : Boolean;

Var

Length, Element, NumOfElements: Integer;

IsCorrect: Boolean;

Begin

NumOfElements := 0;

Length := 0;

Element := 0;

IsCorrect := True;

Reset(F);

Readln(F, NumOfElements);

While IsCorrect And Not EOF(F) Do

Begin

Read(F, Element);

Inc(Length);

IsCorrect := CheckArea(Length, 0 , MAX\_LENGTH);

If Length <> NumOfElements Then

IsCorrect := True

End;

CloseFile(F);

If IsCorrect And (NumOfElements <> Length) 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('Длины 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 ReadFileLengthOfArr(Var F: TextFile) : Integer;

Var

NumOfElements: Integer;

Begin

NumOfElements := 0;

Reset(F);

Read(F, NumOfElements);

CloseFile(F);

ReadFileLengthOfArr := NumOfElements;

End;

Function ReadFileArr(Var F: TextFile; NumOfElements: Integer) : TArr;

Var

Arr: TArr;

Num: Integer;

Begin

Num := 0;

SetLength(Arr, NumOfElements);

Reset(F);

Readln(F);

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

Read(F, Arr[Num]);

CloseFile(F);

ReadFileArr := Arr;

End;

Function ReadConsoleLengthOfArr() : Integer;

Var

NumOfElements: Integer;

IsCorrect: Boolean;

Begin

NumOfElements := 0;

IsCorrect := False;

Repeat

Write('Введите длину числовой последовательности[', MIN\_LENGTH, ' : ', MAX\_LENGTH, ']: ');

Try

Readln(NumOfElements);

IsCorrect := True;

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckArea(NumOfElements, MIN\_LENGTH, MAX\_LENGTH);

Until IsCorrect;

ReadConsoleLengthOfArr := NumOfElements;

End;

Function ReadConsoleArr(NumOfElements: Integer) : TArr;

Var

Arr: TArr;

Num, Row, Col: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := False;

SetLength(Arr, NumOfElements);

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

Repeat

Write('Введите ', (Num + 1), ' член последовательности [', MIN\_ELEMENT, ' : ', MAX\_ELEMENT, ']: ');

Try

Readln(Arr[Num]);

IsCorrect := True;

Except

Writeln('Проверьте корректность ввода данных!');

IsCorrect := False;

End;

If IsCorrect Then

IsCorrect := CheckArea(Arr[Num], MIN\_ELEMENT, MAX\_ELEMENT);

Until IsCorrect;

ReadConsoleArr := Arr;

End;

Procedure ReadArr(Var Arr: TArr; Var NumOfElements: Integer);

Var

RF: TextFile;

Begin

If ChooseFileInput() Then

Begin

GetFileNormalReading(RF);

NumOfElements := ReadFileLengthOfArr(RF);

Arr := ReadFileArr(RF, NumofElements);

End

Else

Begin

NumOfElements := ReadConsoleLengthOfArr();

Arr := ReadConsoleArr(NumOfElements);

End;

End;

Function IsSequenceNonGrowing(Var Arr: TArr): Boolean;

Var

NonGrowing: Boolean;

Ind: Integer;

Begin

NonGrowing := True;

For Ind := Low(Arr) To (High(Arr) - 1) Do

Begin

If Arr[Ind] < Arr[Ind + 1] Then

NonGrowing := False

End;

IsSequenceNonGrowing := NonGrowing;

End;

Procedure FreeMemory(Var Arr: TArr);

Begin

Arr := Nil;

End;

Function ChooseFileOutput() : Boolean;

Var

IsFileOutput: Integer;

IsCorrect, Choose: Boolean;

Begin

IsFileOutput := 0;

IsCorrect := False;

Choose := False;

WriteLn;

Repeat

WriteLn('Вы хотите выводить результат через файл? (Да - ', 1, ' / Нет - ', 0, ')');

Try

ReadLn(IsFileOutput);

IsCorrect := True;

Except

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

If IsCorrect Then

Begin

If IsFileOutput = 1 Then

Choose := True

Else If IsFileOutput = 0 Then

Choose := False

Else

Begin

WriteLn('Некорректный выбор!');

IsCorrect := False;

End;

End;

Until IsCorrect;

ChooseFileOutput := Choose;

End;

Procedure PrintConsoleResult( Var NonGrowing: Boolean);

Begin

If NonGrowing Then

Writeln('Последовательность является невозрастающей')

Else

Writeln('Последовательность не соответствует условию');

End;

Procedure PrintFileResult(Var F: TextFile; Var NonGrowing: Boolean);

Begin

Append(F);

WriteLn(F);

If NonGrowing Then

Writeln('Последовательность является невозрастающей')

Else

Writeln('Последовательность не соответствует условию');

CloseFile(F);

End;

Procedure PrintResult(NonGrowing: Boolean);

Var

WF: TextFile;

Begin

If ChooseFileOutput() Then

Begin

GetFileNormalWriting(WF);

PrintFileResult(WF, NonGrowing);

End

Else

PrintConsoleResult(NonGrowing);

End;

Var

NumOfElements: Integer;

Arr: TArr;

NonGrowing: Boolean;

Begin

NumOfElements:= 0;

NonGrowing := True;

PrintTask();

ReadArr(Arr, NumOfElements);

NonGrowing := IsSequenceNonGrowing(Arr);

PrintResult(NonGrowing);

FreeMemory(Arr);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <string.h>

#include <fstream>

using namespace std;

const int MIN\_LENGTH = 2;

const int MAX\_LENGTH = 1000;

const int MIN\_ELEMENT = -10000;

const int MAX\_ELEMENT = 10000;

void printTask()

{

cout << "Данная программа проверяет невозрастающая ли последовательность:\n\n";

}

void setLengthArr(int\*& arr, int length)

{

arr = new int [length];

}

bool chooseFileInput()

{

int isFileInput;

bool isCorrect, choose;

isFileInput = 0;

choose = false;

do {

isCorrect = true;

cout << "Вы хотите вводить последовательность через файл? (Да - " << 1 << " / Нет - " << 0 << ")\n";

cin >> isFileInput;

if (cin.fail())

{

isCorrect = false;

cout << "Некорректный выбор!\n";

cin.clear();

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

}

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

{

isCorrect = false;

cout << "Некорректный выбор!\n";

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

}

if (isCorrect)

{

if (isFileInput == 1)

choose = true;

else if (isFileInput == 0)

choose = false;

else

{

isCorrect = false;

cout << "Некорректный выбор!\n";

}

}

} while (!isCorrect);

return choose;

}

bool checkArea(int num, const int MIN, const int MAX)

{

bool isCorrect;

isCorrect = true;

if (num < MIN || num > MAX)

{

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

isCorrect = false;

}

return isCorrect;

}

string readPathFile()

{

string pathToFile;

bool isCorrect;

int len;

pathToFile = "";

isCorrect = false;

len = 0;

do

{

cout << "Введите путь к файлу с расширением .txt с последовательностью, у которой количество членов должно быть не меньше " << MIN\_LENGTH << " и не больше " << MAX\_LENGTH << " а элементы должны быть в диапазоне [" << MIN\_ELEMENT << " : " << MAX\_ELEMENT << "]:";

cin >> pathToFile;

len = pathToFile.length();

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

isCorrect = true;

else

{

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

isCorrect = false;

}

} while (!isCorrect);

return pathToFile;

}

bool isExists(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ifstream file(pathToFile);

if (file.good())

isCorrect = true;

file.close();

return isCorrect;

}

bool isAbleToReading(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ifstream file(pathToFile);

if (file.is\_open())

isCorrect = true;

file.close();

return isCorrect;

}

bool isAbleToWriting(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ofstream file(pathToFile, ios::app);

if (file.is\_open())

isCorrect = true;

file.close();

return isCorrect;

}

bool isEmpty(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ifstream file(pathToFile);

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

isCorrect = true;

file.close();

return isCorrect;

}

bool isRightFileNums(string pathToFile)

{

int element;

int numOfElements;

bool isCorrect;

numOfElements = 0;

element = 0;

isCorrect = true;

ifstream file(pathToFile);

file >> numOfElements;

if (file.fail())

{

isCorrect = false;

file.clear();

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

}

if (isCorrect)

{

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

{

isCorrect = false;

cout << "Длины не равны\n";

}

}

if (isCorrect)

isCorrect = checkArea(numOfElements, MIN\_LENGTH, MAX\_LENGTH);

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

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

{

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

{

file >> element;

if (file.fail())

{

isCorrect = false;

file.clear();

}

if (isCorrect)

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

}

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

}

file.close();

return isCorrect;

}

bool isOrdersEqual(string pathToFile)

{

int length;

int element;

int numOfElements;

bool isCorrect;

numOfElements = 0;

length = 0;

element = 0;

isCorrect = true;

ifstream file(pathToFile);

file >> numOfElements;

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

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

{

file >> element;

length++;

isCorrect = checkArea(length, 0, MAX\_LENGTH);

if (length != numOfElements)

isCorrect = true;

}

file.close();

if (isCorrect && (numOfElements != length))

isCorrect = false;

return isCorrect;

}

void getFileNormalReading(string& pathToFile)

{

bool isCorrect;

do

{

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile))

{

isCorrect = false;

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

}

if (isCorrect && !isAbleToReading(pathToFile))

{

isCorrect = false;

cout << "Файл закрыт для чтения!\n";

}

if (isCorrect && !isAbleToWriting(pathToFile))

{

isCorrect = false;

cout << "Файл закрыт для записи!\n";

}

if (isCorrect && isEmpty(pathToFile))

{

isCorrect = false;

cout << "Файл пуст!\n";

}

if (isCorrect && !isRightFileNums(pathToFile))

{

isCorrect = false;

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

}

if (isCorrect && !isOrdersEqual(pathToFile))

{

isCorrect = false;

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

}

} while (!isCorrect);

}

void getFileNormalWriting(string& pathToFile)

{

bool isCorrect;

do

{

isCorrect = true;

pathToFile = readPathFile();

if (!isExists(pathToFile))

{

isCorrect = false;

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

}

if (isCorrect && !isAbleToWriting(pathToFile))

{

isCorrect = false;

cout << "Файл закрыт для записи!\n";

}

} while (!isCorrect);

}

int readFileLengthOfArr(string pathToFile)

{

int numOfElements;

numOfElements = 0;

ifstream file(pathToFile);

file >> numOfElements;

file.close();

return numOfElements;

}

int\* readFileArr(string pathToFile, int numOfElements)

{

int\* arr;

int num;

setLengthArr(arr, numOfElements);

ifstream file(pathToFile);

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

for (num = 0; num < numOfElements; num++)

{

file >> arr[num];

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

}

file.close();

return arr;

}

int readConsoleLengthOfArr()

{

int numOfElements;

bool isCorrect;

numOfElements = 0;

do

{

isCorrect = true;

cout << "Введите длинну числовой последовательности[" << MIN\_LENGTH << " : " << MAX\_LENGTH << "]: ";

cin >> numOfElements;

if (cin.fail())

{

isCorrect = false;

cin.clear();

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

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

}

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

{

isCorrect = false;

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

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

}

if (isCorrect)

isCorrect = checkArea(numOfElements, MIN\_LENGTH, MAX\_LENGTH);

} while (!isCorrect);

return numOfElements;

}

int\* readConsoleArr(int numOfElements)

{

int\* arr;

bool isCorrect;

int num;

num = 0;

setLengthArr(arr, numOfElements);

for (int num = 0; num < numOfElements; num++)

do

{

isCorrect = true;

cout << "Введите в " << (num + 1) << " член последовательности [" << MAX\_ELEMENT << " : " << MIN\_ELEMENT << "]: ";

cin >> arr[num];

if (cin.fail())

{

isCorrect = false;

cin.clear();

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

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

}

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

{

isCorrect = false;

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

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

}

if (isCorrect)

isCorrect = checkArea(arr[num], MIN\_ELEMENT, MAX\_ELEMENT);

} while (!isCorrect);

return arr;

}

void readArr(int\*& arr, int& numOfElements)

{

string pathToFile;

pathToFile = "";

if (chooseFileInput())

{

getFileNormalReading(pathToFile);

numOfElements = readFileLengthOfArr(pathToFile);

arr = readFileArr(pathToFile, numOfElements);

}

else

{

numOfElements = readConsoleLengthOfArr();

arr = readConsoleArr(numOfElements);

}

}

bool isSequenceNonGrowing(int\*& arr, int numOfElements)

{

bool nonGrowing;

int ind;

nonGrowing = true;

ind = 0;

for (ind; ind < numOfElements - 1; ind++)

{

if (arr[ind] < arr[ind + 1])

nonGrowing = false;

}

return nonGrowing;

}

void freeMemory(int\*& arr)

{

delete[] arr;

}

bool chooseFileOutput()

{

int isFileOutput;

bool isCorrect, choose;

isFileOutput = 0;

choose = false;

cout << "\n";

do {

isCorrect = true;

cout << "Вы хотите выводить результат через файл? (Да - " << 1 << " / Нет - " << 0 << ")\n";

cin >> isFileOutput;

if (cin.fail())

{

isCorrect = false;

cout << "Некорректный выбор!\n";

cin.clear();

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

}

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

{

isCorrect = false;

cout << "Некорректный выбор!\n";

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

}

if (isCorrect)

{

if (isFileOutput == 1)

choose = true;

else if (isFileOutput == 0)

choose = false;

else

{

isCorrect = false;

cout << "Некорректный выбор!\n";

}

}

} while (!isCorrect);

return choose;

}

void printConsoleResult(bool& nonGrowing)

{

if (nonGrowing)

cout << "Последовательность является невозрастающей";

else

cout << "Последовательность не соответствует условию";

}

void printFileResult(string pathToFile, bool& nonGrowing)

{

fstream file(pathToFile, ios::app);

file << "\n";

if (nonGrowing)

file << "Последовательность является невозрастающей";

else

file << "Последовательность не соответствует условию";

file.close();

}

void printResult(bool nonGrowing)

{

string pathToFile;

pathToFile = "";

if (chooseFileOutput())

{

getFileNormalWriting(pathToFile);

printFileResult(pathToFile, nonGrowing);

}

else

printConsoleResult(nonGrowing);

}

int main()

{

bool nonGrowing;

int\* arr;

int numOfElements;

nonGrowing = true;

numOfElements = 0;

setlocale(LC\_ALL, "RU");

printTask();

readArr(arr, numOfElements);

nonGrowing = isSequenceNonGrowing(arr, numOfElements);

printResult(nonGrowing);

freeMemory(arr);

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\_LENGTH = 2;

public static final int MAX\_LENGTH = 1000;

public static final int MIN\_ELEMENT = -10000;

public static final int MAX\_ELEMENT = 10000;

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

public static Scanner scanFile;

public static File file;

public static void printTask() {

System.out.println("Данная программа проверяет невозрастающая ли последовательность:\n");

}

public static boolean chooseFileInput() {

int isFileInput;

boolean isCorrect, choose;

isFileInput = 0;

choose = true;

do {

System.out.println("Вы хотите вводить последовательность через файл? (Да - " + 1 + " / Нет - " + 0 + ")");

isCorrect = true;

try {

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

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect) {

if (isFileInput == 1)

choose = true;

else if (isFileInput == 0)

choose = false;

else {

isCorrect = false;

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

}

}

} while (!isCorrect);

return choose;

}

public static boolean checkArea(int num, final int MIN, final int MAX) {

boolean isCorrect;

isCorrect = true;

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

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

isCorrect = false;

}

return isCorrect;

}

public static String readPathFile() {

boolean isCorrect;

int len;

String pathToFile;

isCorrect = false;

len = 0;

pathToFile = "";

do {

System.out.println("Введите путь к файлу с расширением .txt с последовательностью, у которой количество членов должно быть не меньше " + MAX\_LENGTH + " и не больше " + MIN\_LENGTH + " а элементы должны быть в диапазоне [" + MAX\_ELEMENT + " : " + MAX\_ELEMENT + "]:" );

pathToFile = scanConsole.nextLine();

len = pathToFile.length();

if (len > 4 && pathToFile.substring(len - 4).equals(".txt"))

isCorrect = true;

else

{

isCorrect = false;

System.out.println("Расширение файла не .txt!");

}

} while (!isCorrect);

return pathToFile;

}

public static boolean isExists(String pathToFile) {

boolean isCorrect;

isCorrect = false;

file = new File(pathToFile);

if (file.exists())

isCorrect = true;

return isCorrect;

}

public static boolean isAbleToReading() {

boolean isCorrect;

isCorrect = false;

if (file.canRead())

isCorrect = true;

return isCorrect;

}

public static boolean isAbleToWriting() {

boolean isCorrect;

isCorrect = false;

if (file.canWrite())

isCorrect = true;

return isCorrect;

}

public static boolean isEmpty() {

boolean isCorrect;

isCorrect = false;

if (file.length() == 0)

isCorrect = true;

return isCorrect;

}

public static boolean isRightFileNums() {

int numOfElements;

int element;

boolean isCorrect;

isCorrect = true;

element = 0;

numOfElements = 0;

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

try {

numOfElements = 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("Длины не равны");

}

}

if (isCorrect)

isCorrect = checkArea(numOfElements, MIN\_LENGTH, MAX\_LENGTH);

scanFile.nextLine();

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

try {

element = scanFile.nextInt();

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

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

}

scanFile.close();

return isCorrect;

}

public static boolean isOrdersEqual() {

int length;

int element;

int numOfElements;

boolean isCorrect;

numOfElements = 0;

length = 0;

element = 0;

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

numOfElements = scanFile.nextInt();

isCorrect = true;

scanFile.nextLine();

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

element = scanFile.nextInt();

length++;

isCorrect = checkArea(length, 0, MAX\_LENGTH);

if (length > numOfElements)

isCorrect = false;

}

scanFile.close();

if (isCorrect && (numOfElements != length))

isCorrect = false;

return isCorrect;

}

public static void getFileNormalReading() {

boolean isCorrect;

String pathToFile;

int order;

order = 0;

pathToFile = "";

do {

isCorrect = true;

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 readFileLengthOfArr() {

int numOfElements;

numOfElements = 0;

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

numOfElements = scanFile.nextInt();

scanFile.close();

return numOfElements;

}

public static int[] readFileArr(int numOfElements) {

int[] arr;

arr = new int[numOfElements];

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

scanFile.nextLine();

for (int num = 0; num < numOfElements; num++)

arr[num] = scanFile.nextInt();

scanFile.close();

return arr;

}

public static int readConsoleOrder() {

int numOfElements;

boolean isCorrect;

numOfElements = 0;

do {

System.out.print("Введите длинну числовой последовательности[" + MIN\_LENGTH + " : " + MAX\_LENGTH + "]: ");

isCorrect = true;

try {

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

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

isCorrect = checkArea(numOfElements, MIN\_LENGTH, MAX\_LENGTH);

} while (!isCorrect);

return numOfElements;

}

public static int[] readConsoleMatrix(int numOfElemnts) {

int[] arr;

boolean isCorrect;

arr = new int[numOfElemnts];

for (int num = 0; num < numOfElemnts; num++)

do {

System.out.print("Введите в " + (num + 1) + " член последовательности [" + MAX\_ELEMENT + " : " + MAX\_ELEMENT + "]: ");

isCorrect = true;

try {

arr[num] = Integer.parseInt(scanConsole.nextLine());

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect)

isCorrect = checkArea(arr[num], MIN\_ELEMENT, MAX\_ELEMENT);

} while (!isCorrect);

return arr;

}

public static int[] readArr() {

int[] arr;

int numOfElements;

if (chooseFileInput()) {

getFileNormalReading();

numOfElements = readFileLengthOfArr();

arr = readFileArr(numOfElements);

}

else {

numOfElements = readConsoleOrder();

arr = readConsoleMatrix(numOfElements);

}

return arr;

}

public static boolean isSequenceNonGrowing(int[] arr, int numOfElements) {

boolean nonGrowing;

int ind;

nonGrowing = true;

ind = 0;

for (ind = 0; ind < numOfElements - 1; ind++) {

if (arr[ind] < arr[ind + 1])

nonGrowing = false;

}

return nonGrowing;

}

public static boolean chooseFileOutput() {

int isFileOutput;

boolean isCorrect, choose;

isFileOutput = 0;

choose = true;

do {

System.out.println("Вы хотите выводить матрицу через файл? (Да - " + 1 + " / Нет - " + 0 + ")");

isCorrect = true;

try {

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

} catch (NumberFormatException e) {

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

isCorrect = false;

}

if (isCorrect) {

if (isFileOutput == 1)

choose = true;

else if (isFileOutput == 0)

choose = false;

else {

isCorrect = false;

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

}

}

} while (!isCorrect);

return choose;

}

public static void printConsoleResult(boolean nonGrowing) {

if (nonGrowing)

System.out.print("Полседовательность является невозрастающей");

else

System.out.print("Последовательность не соотвествует условию");

}

public static void printFileResult(boolean nonGrowing) {

try {

FileWriter writer = new FileWriter(file, true);

if (nonGrowing)

writer.write("Полседовательность является невозрастающей");

else

writer.write("Последовательность не соотвествует условию");

writer.close();

} catch (IOException e) {}

}

public static void printResult(boolean nonGrowing) {

if (chooseFileOutput()) {

getFileNormalWriting();

printFileResult(nonGrowing);

}

else

printConsoleResult(nonGrowing);

}

public static void main(String[] args) {

boolean nonGrowing;

int numOfElements;

int[] arr;

nonGrowing = true;

numOfElements = 0;

printTask();

arr = readArr();

numOfElements = arr.length;

nonGrowing = isSequenceNonGrowing(arr, numOfElements);

printResult(nonGrowing);

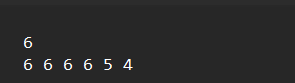
scanConsole.close();

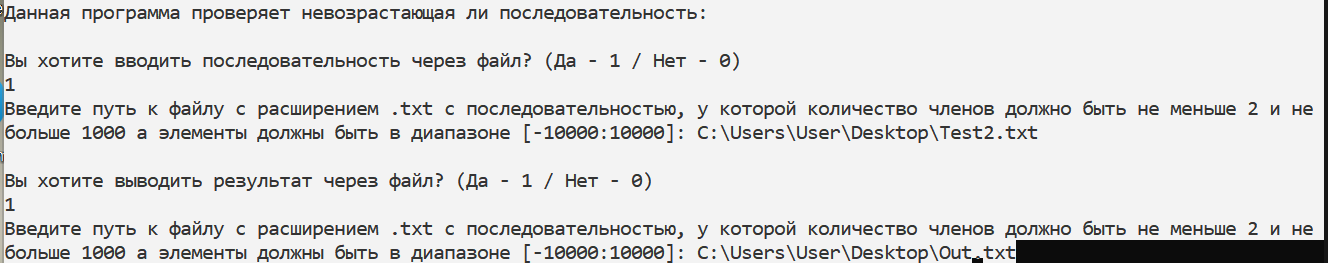
}

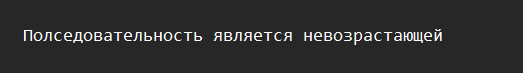
}

**Скриншоты:**

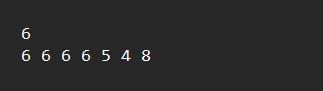
**Delphi:**

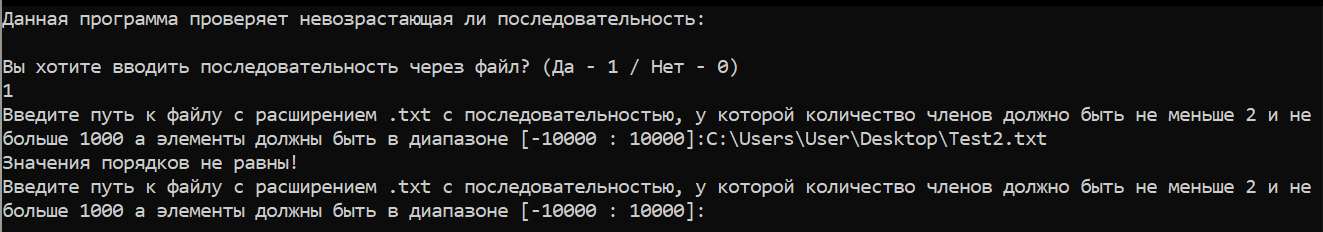
****

****

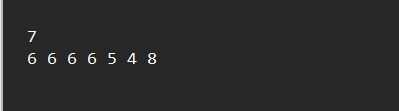
****

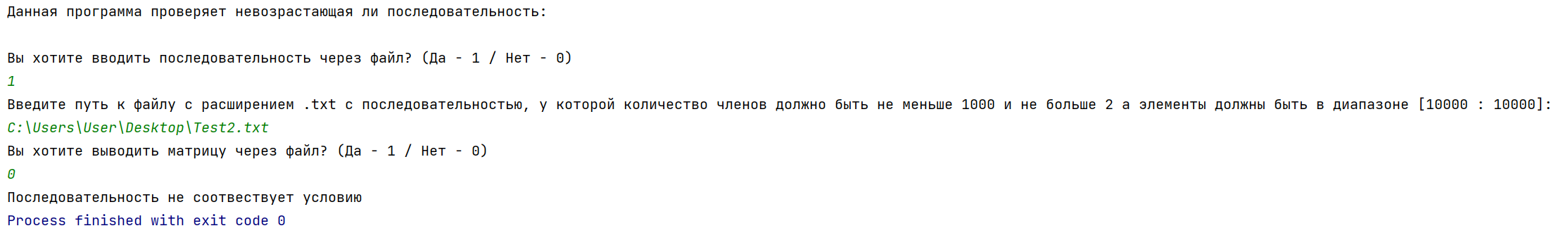
**C++:**

****

****

**Java:**

****

****

**Блок-схема:**









