УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2.3

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.

Дано натуральное n-значное число P. Проверить, является ли данное число палиндромом (перевертышем).

Код программы на **Delphi**:

Program labt3;

{$APPTYPE CONSOLE}

{$R \*.res}

Uses

System.SysUtils;

Const

CONS\_NUM: Integer = 1;

FILE\_NUM: Integer = 2;

PALIN\_OUTPUT\_CONTROL: Integer = -1;

Procedure PrintStatement();

Begin

Writeln('The program determines whether', #13#10#9,

'the entered natural number is a palindrome.', #13#10);

End;

Function PathCondition(Var IsCorrect: Boolean): Integer;

Var

Num: Integer;

Begin

Try

Readln(Num);

Except

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

End;

If (Num <> CONS\_NUM) And (Num <> FILE\_NUM) Then

Writeln('Choose only ', CONS\_NUM, ' or ', FILE\_NUM, '. Try again.')

Else

IsCorrect := True;

PathCondition := Num;

End;

Function ChoosingAPath(): Integer;

Var

Res: Integer;

IsCorrect: Boolean;

Begin

Writeln('Where will we work through: ', #13#10#9, 'Console: ',

CONS\_NUM, #9, 'File: ', FILE\_NUM, #13#10);

IsCorrect := False;

Repeat

Write('Your choice: ');

Res := PathCondition(IsCorrect);

Until IsCorrect;

ChoosingAPath := Res;

End;

Procedure PalinCondition(Var IsCorrect: Boolean; Var Palindrome: Integer);

Begin

Try

Readln(Palindrome);

Except

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

End;

If Palindrome < 1 Then

Writeln('Number should be natural.')

Else

IsCorrect := True;

End;

Function InputPalin(): Integer;

Var

Palindrome: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := False;

Repeat

Write('Write your number: ');

PalinCondition(IsCorrect, Palindrome);

Until IsCorrect;

InputPalin := Palindrome;

End;

Function LengthOfPalin(Palindrome: Integer): Integer;

Var

PalinLen: Integer;

Begin

PalinLen := 0;

While (Palindrome > 0) Do

Begin

Inc(PalinLen);

Palindrome := Palindrome Div 10;

End;

LengthOfPalin := PalinLen;

End;

Procedure PutInMassive(Var ArrPalin: Array Of Integer; Palindrome: Integer);

Var

I: Integer;

Begin

I := 0;

While Palindrome > 0 Do

Begin

ArrPalin[I] := Palindrome Mod 10;

Inc(I);

Palindrome := Palindrome Div 10;

End;

End;

Function PalinIsPalin(Var ArrPalin: Array Of Integer; PalinLen: Integer; Palindrome: Integer): Boolean;

Var

IsCorrect: Boolean;

I: Integer;

Begin

IsCorrect := True;

For I := 0 To PalinLen Div 2 Do

If (ArrPalin[I] <> ArrPalin[PalinLen - I - 1]) Then

IsCorrect := False;

If Palindrome < 0 Then

IsCorrect := False;

PalinIsPalin := IsCorrect;

End;

Function PalinCheack(Palindrome: Integer): Boolean;

Var

PalinLen: Integer;

Res: Boolean;

ArrPalin: Array Of Integer;

Begin

PalinLen := LengthOfPalin(Abs(Palindrome));

SetLength(ArrPalin, PalinLen);

PutInMassive(ArrPalin, Abs(Palindrome));

Res := PalinIsPalin(ArrPalin, PalinLen, Palindrome);

ArrPalin := Nil;

PalinCheack := Res;

End;

Function ViaConsole(): Integer;

Var

Palindrome, Res: Integer;

Begin

Palindrome := InputPalin();

If PalinCheack(Palindrome) And (Palindrome > PALIN\_OUTPUT\_CONTROL) Then

Res := 1

Else

Res := 0;

ViaConsole := Res;

End;

Function InputPalinFile(Var MyFile: TextFile): Integer;

Var

Palindrome: Integer;

Begin

Reset(MyFile);

Try

Readln(MyFile, Palindrome);

Except

Write('Bad number in file.');

Palindrome := -1;

End;

If Palindrome < 0 Then

Begin

Write('Number should be > 0.');

Palindrome := -1;

End

Else

If Not SeekEof(MyFile) Then

Begin

Write('Should be only one num.');

Palindrome := -1;

End;

InputPalinFile := Palindrome;

End;

Function OutputPalin(Palindrome: Integer): Integer;

Var

Res: Integer;

Begin

If Palindrome = PALIN\_OUTPUT\_CONTROL Then

Res := -1

Else

If (PalinCheack(Palindrome)) Then

Res := 1

Else

If (Palindrome <> PALIN\_OUTPUT\_CONTROL) Then

Res := 0;

OutputPalin := Res;

End;

Function WorkWithFile(Var MyFile: TextFile): Integer;

Var

Palindrome, Res: Integer;

Begin

Palindrome := InputPalinFile(MyFile);

Res := OutputPalin(Palindrome);

WorkWithFile := Res;

End;

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

Var

Bufstr: String;

Begin

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

Writeln('Write .txt file.')

Else

IsCorrect := True;

End;

Function InputWay(): String;

Var

Way: String;

IsCorrect: Boolean;

Begin

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

Repeat

IsCorrect := False;

Read(Way);

WayCondition(Way, IsCorrect);

Readln;

Until IsCorrect;

InputWay := Way;

End;

Function ViaFile(): Integer;

Var

FileWay: String;

MyFIle: TextFile;

Res: Integer;

Begin

FileWay := InputWay();

AssignFile(MyFile, FileWay);

Try

Reset(MyFile);

Res := WorkWithFile(MyFile);

Except

Begin

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

Res := -1;

End;

End;

ViaFile := Res;

End;

Procedure OutputViaConsole(Result: Integer);

Begin

If Result = 1 Then

Writeln('Palindrome.')

Else

Writeln('Not a palindrome.');

End;

Function FileCorrectOutput(Res: Integer): String;

Var

Resstr: String;

Begin

If Res = 1 Then

Resstr := 'Palindrome.'

Else

Resstr := 'Not a palindrome';

FileCorrectOutput := Resstr;

End;

Procedure OutputViaFile(Result: Integer);

Var

FileWay: String;

MyFile: TextFile;

Begin

FileWay := InputWay();

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;

Resoult: String;

Begin

PrintStatement();

Option := ChoosingAPath();

If Option = FILE\_NUM Then

Result := ViaFile()

Else

Result := ViaConsole();

Output(Option, Result);

End.

Код программы на **C++**:

#include <iostream>

#include <fstream>

using namespace std;

const int CONS\_NUM = 1;

const int FILE\_NUM = 2;

const int PALIN\_OUTPUT\_CONTROL = -1;

void printStatement() {

cout << "The program determines whether\n\t"

<< "the entered natural number is a palindrome.\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

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

}

else

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

}

string inputWay() {

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

string way;

bool isIncorrect = true;

do {

cin >> way;

wayCondition(way, isIncorrect);

} while (isIncorrect);

return way;

}

int pathCondition(bool& isIncorrect) {

int num = 0;

cin >> num;

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

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

cin.clear();

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

}

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

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

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

else

isIncorrect = false;

return num;

}

int choosingAPath() {

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

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

int result;

bool isIncorrect = true;

do {

cout << "Your choice: ";

result = pathCondition(isIncorrect);

} while (isIncorrect);

return result;

}

void palinCondition(bool& isIncorrect, int& palindrome) {

cin >> palindrome;

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

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

cin.clear();

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

}

else if (palindrome < 1)

cout << "Number should be natural.\n";

else

isIncorrect = false;

}

int inputPalin() {

bool isIncorrect = true;

int palindrome;

do {

cout << "Write your number: ";

palinCondition(isIncorrect, palindrome);

} while (isIncorrect);

return palindrome;

}

int inputPalinFile(fstream& file) {

int palindrome = 0;

file >> palindrome;

if (file.fail()) {

cout << "Bad number in file.";

palindrome = -1;

file.clear();

}

char sim;

if (palindrome < 0) {

cout << "Number should be > 0.";

palindrome = -1;

}

else if (file.get(sim) && sim != ' ' && sim != '\n') {

palindrome = -1;

cout << "Should be only one num.";

}

return palindrome;

}

int lengthOfPalin(int palindrome) {

int palinLen = 0;

while (palindrome) {

palinLen++;

palindrome /= 10;

}

return palinLen;

}

void putInMassive(int\*& arrPalin, int palindrome) {

int i = 0;

while (palindrome) {

arrPalin[i] = palindrome % 10;

i++;

palindrome = palindrome / 10;

}

}

bool palinIsPalin(int\*& arrPalin, int palinLen, int palindrome) {

bool isCorrect = true;

for (int i = 0; i < palinLen / 2; i++) {

if (arrPalin[i] != arrPalin[palinLen - i - 1])

isCorrect = false;

}

if (palindrome < 0)

isCorrect = false;

return isCorrect;

}

bool palinCheack(int palindrome) {

int palinLen = lengthOfPalin(palindrome);

int\* arrPalin = new int[palinLen];

putInMassive(arrPalin, palindrome);

bool result = palinIsPalin(arrPalin, palinLen, palindrome);

delete[] arrPalin;

arrPalin = nullptr;

return result;

}

int viaConsole() {

int palindrome = inputPalin();

if (palinCheack(palindrome) && palindrome > PALIN\_OUTPUT\_CONTROL)

return 1;

else

return 0;

}

int outputPalin(int palindrome) {

if (palindrome == PALIN\_OUTPUT\_CONTROL){

cout << "ERROR.";

return -1;

}

else if (palinCheack(palindrome))

return 1;

else if (palindrome != PALIN\_OUTPUT\_CONTROL)

return 0;

}

int workWithFile(fstream& file) {

if (file.is\_open()) {

int palindrome = inputPalinFile(file);

return outputPalin(palindrome);

}

else {

cout << "Bad File.";

return -1;

}

}

int viaFile() {

string fileWay;

fileWay = inputWay();

fstream file;

file.open(fileWay);

int result = workWithFile(file);

file.clear();

file.close();

return result;

}

void outputViaConsole(int result) {

if (result)

cout << "Palindrome.";

else

cout << "Not a palidrome.";

}

string fileCorrectOutput(int result) {

if (result)

return "\nPalindrome.";

else

return "\nNot a palindrome.";

}

void outputViaFile(int result) {

string fileWay = inputWay();

ofstream file(fileWay, ios::app);

if (file.is\_open()) {

file << fileCorrectOutput(result);

cout << "Check your file.";

}

else

cout << "\nBad output file.";

file.clear();

file.close();

}

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);

}

}

int main() {

printStatement();

int option = choosingAPath();

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

output(option, result);

return 0;

}

Код программы на **Java**:

import java.io.\*;  
import java.util.Scanner;  
import java.util.concurrent.atomic.AtomicBoolean;  
import java.util.concurrent.atomic.AtomicInteger;  
  
class ImportDate{  
 private boolean isIncorrect;  
 private int palindrome;  
  
 public ImportDate(){  
  
 }  
  
 public ImportDate(boolean isIncorrect, int palindrome){  
 this.palindrome = palindrome;  
 this.isIncorrect = isIncorrect;  
 }  
  
 public boolean isIncorrect() {  
 return isIncorrect;  
 }  
  
 public void setIncorrect(boolean incorrect) {  
 isIncorrect = incorrect;  
 }  
  
 public int getPalindrome() {  
 return palindrome;  
 }  
  
 public void setPalindrome(int palindrome) {  
 this.palindrome = palindrome;  
 }  
}  
  
public class lab3 {  
 static final int *CONS\_NUM* = 1;  
 static final int *FILE\_NUM* = 2;  
 static final int *PALIN\_OUTPUT\_CONTROL* = -1;  
  
  
 static void printStatement(){  
 System.*out*.print("The program determines whether\n\t

the entered natural number is a palindrome.\n\n");  
 }  
  
  
 static void wayCondition(String way, ImportDate Date) {  
 if (way.length() > 4) {  
 String bufstr = way.substring(way.length() - 4);  
 if (bufstr.equals(".txt"))  
 Date.setIncorrect(false);  
 else  
 System.*out*.print("Write .txt file.\n");  
 }  
 else  
 System.*out*.print("The path is too short.\n");  
 }  
  
  
 static String inputWay(Scanner in) {  
 System.*out*.print("Write way to your file: ");  
 String way;  
 ImportDate Date = new ImportDate(true, 0);  
 do {  
 way = in.nextLine();  
 *wayCondition*(way, Date);  
 } while (Date.isIncorrect());  
  
 return way;  
 }  
  
 static int pathCondition(ImportDate Date, Scanner in) {  
 int num = 0;  
 try {  
 num = Integer.*parseInt*(in.nextLine());  
 } catch (NumberFormatException error) {  
 System.*out*.print("Invalid numeric input. Try again.\n");  
 }  
 if (num != *CONS\_NUM* && num != *FILE\_NUM*)  
 System.*out*.printf("Choose only %d or %d. Try again.\n", *CONS\_NUM*,

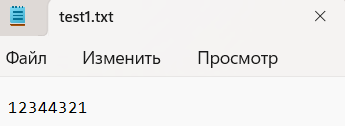
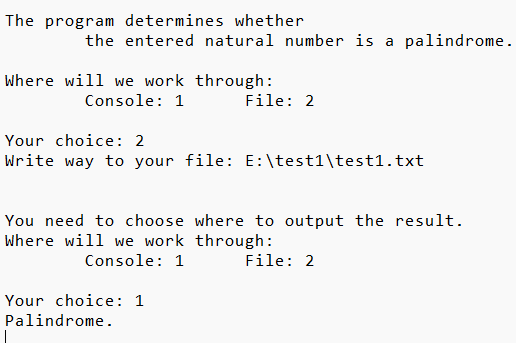
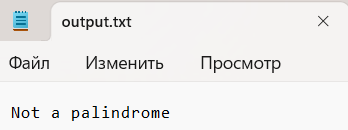
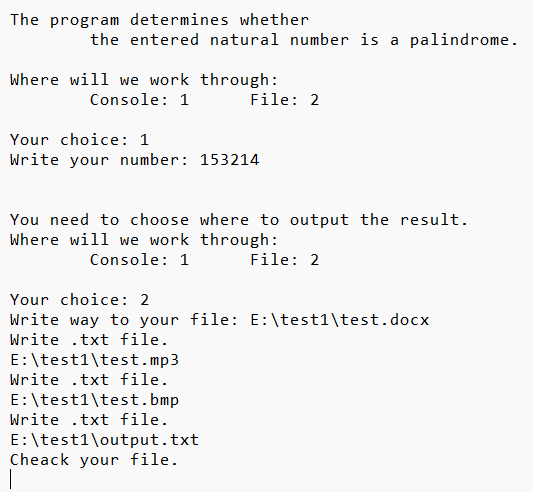
*FILE\_NUM*);  
 else Date.setIncorrect(false);  
 return num;  
 }  
  
 static int choosingAPath(Scanner in) {  
 System.*out*.printf("Where will we work through: \n\tConsole: %d \tFile:

%d\n\n", *CONS\_NUM*, *FILE\_NUM*);  
 ImportDate Date = new ImportDate(true, 0);  
 int result;  
 do {  
 System.*out*.print("Your choice: ");  
 result = *pathCondition*(Date, in);  
 } while (Date.isIncorrect());  
  
 return result;  
 }  
  
 static void palinCondition(ImportDate Date, Scanner in) {  
 try{  
 Date.setPalindrome(Integer.*parseInt*(in.nextLine()));  
 } catch (NumberFormatException error) {  
 System.*out*.print("Invalid numeric input.Try again.\n");  
 }  
 if (Date.getPalindrome() < 1)  
 System.*out*.print("Number should be natural.\n");  
 else  
 Date.setIncorrect(false);  
 }  
  
  
 static int inputPalin(Scanner in) {  
 ImportDate Date = new ImportDate(true, 0);  
 do {  
 System.*out*.print("Write your number: ");  
 *palinCondition*(Date, in);  
 } while (Date.isIncorrect());  
  
 return Date.getPalindrome();  
 }  
  
  
 static int lengthOfPalin(int palindrome) {  
 int palinLen = 0;  
 while (palindrome > 0) {  
 palinLen++;  
 palindrome /= 10;  
 }  
  
 return palinLen;  
 }  
  
  
 static void putInMassive(int[] arrPalin, int palindrome) {  
 int i = 0;  
 while (palindrome > 0) {  
 arrPalin[i] = palindrome % 10;  
 i++;  
 palindrome = palindrome / 10;  
 }  
 }  
  
  
 static boolean palinIsPalin(int[] arrPalin, int palinLen, int palindrome) {  
 boolean isCorrect = true;  
 for (int i = 0; i < palinLen / 2; i++) {  
 if (arrPalin[i] != arrPalin[palinLen - i - 1])  
 isCorrect = false;  
 }  
 if (palindrome < 0)  
 isCorrect = false;  
  
 return isCorrect;  
 }  
  
  
 static boolean palinCheack(int palindrome) {  
 int palinLen = *lengthOfPalin*(palindrome);  
 int[] arrPalin = new int[palinLen];  
 *putInMassive*(arrPalin, palindrome);  
  
 return *palinIsPalin*(arrPalin, palinLen, palindrome);  
 }  
  
  
 static int viaConsole(Scanner in) {  
 int palindrome = *inputPalin*(in);  
 if (*palinCheack*(palindrome) && palindrome > *PALIN\_OUTPUT\_CONTROL*)  
 return 1;  
 else  
 return 0;  
 }  
  
  
  
 static int inputPalinFile(BufferedReader bufferedReader) throws IOException {  
 int palindrome = 0;  
 String line = bufferedReader.readLine();  
 if (line != null)  
 palindrome = Integer.*parseInt*(line);  
  
 if (palindrome < 0) {  
 System.*out*.print("Number should be > 0.");  
 palindrome = -1;  
 }  
 return palindrome;  
 }  
  
 static int outputPalin(int palindrome){  
 if (palindrome == *PALIN\_OUTPUT\_CONTROL*) {  
 System.*out*.print("ERROR.");  
 return -1;  
 }  
 else if (*palinCheack*(palindrome))  
 return 1;  
 else return 0;  
 }  
  
 static int workWithFile(String fileWay, Scanner in){  
 assert fileWay != null;  
 File file = new File(fileWay);  
 try {  
 FileReader fileReader = new FileReader(file);  
 BufferedReader bufferedReader = new BufferedReader(fileReader);  
 int palindrome = *inputPalinFile*(bufferedReader);  
 bufferedReader.close();  
 return *outputPalin*(palindrome);  
 } catch (IOException error) {  
 System.*err*.println("Bad File.");  
 return -1;  
 }  
 }  
  
  
 static int viaFile(Scanner in) {  
 String fileWay;  
 fileWay = *inputWay*(in);  
 return *workWithFile*(fileWay, in);  
 }  
  
  
 static void outputViaConsole(int result) {  
 if (result == 1)  
 System.*out*.print("Palindrome.");  
 else  
 System.*out*.print("Not a palidrome.");  
 }  
  
  
 static String fileCorrectOutput(int result) {  
 if (result == 1)  
 return "\nPalindrome.";  
 else  
 return "\nNot a palindrome.";  
 }  
  
  
 static void outputViaFile(int result, Scanner in) throws IOException {  
 String fileWay = *inputWay*(in);  
 File file = new File(fileWay);  
 try {  
 FileWriter writer = new FileWriter(fileWay, true);  
 writer.write(*fileCorrectOutput*(result));  
 writer.close();  
 System.*out*.print("Cheack your file.");  
 } catch (IOException error) {  
 System.*err*.println("\nBad output file.");  
 }  
 }  
  
 private static void output(int option, int result, Scanner in)

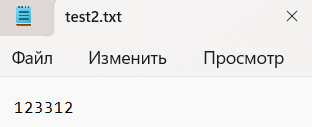
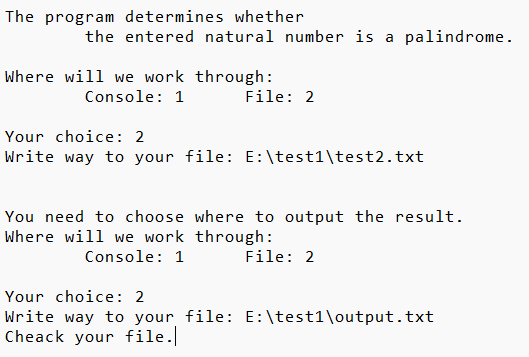
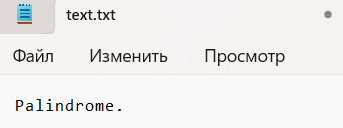
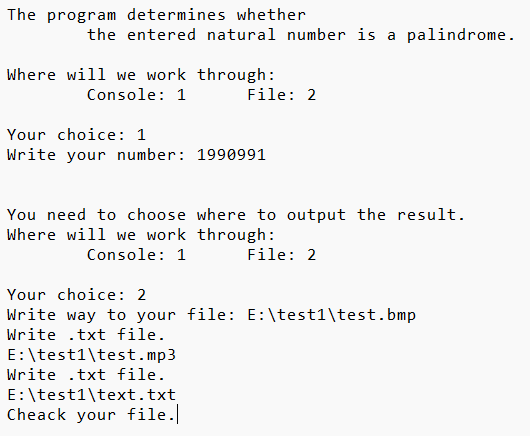
throws IOException {  
 if (result != -1) {  
 System.*out*.print("\n\nYou need to choose where to output the

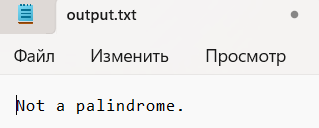
result.\n");  
 option = *choosingAPath*(in);  
  
 if (option == *FILE\_NUM*) {  
 *outputViaFile*(result, in);  
 } else {  
 *outputViaConsole*(result);  
 }  
 }  
 }  
  
  
 public static void main(String[] args) throws IOException {  
 Scanner in = new Scanner(System.*in*);  
 *printStatement*();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

Результат на **Delphi**:

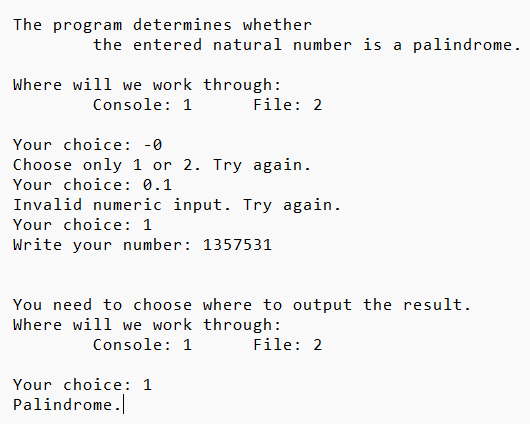


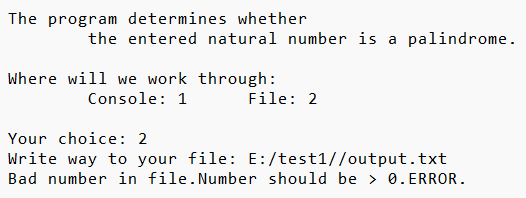
Результат на **Java**:

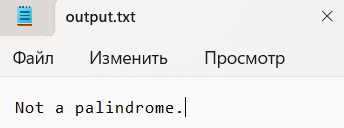




Результат на **C++**:







Блок-схема:

