УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.1

по предмету

Основы алгоритмизации и программирования

Вариант 3

Выполнил:

Бетеня К.С.

Проверила:

Данилова Г.В.

Группа 351005

Минск 2023

Задание:

Разработать программу решения задачи с использованием процедур и функций над строками и операций над множествами. Определить номер позиции k-го вхождения строки st1 в строку st2. Если такого нет, возвратить -1.

Код программы на **Delphi**:

Program Lab1;

{$APPTYPE CONSOLE}

{$R \*.res}

Uses

System.SysUtils;

Const

MIN\_K = 1;

ERR\_VALUE\_OF\_K = -1;

VALUE\_OF\_DEFAULT\_RESULT = -1;

STANDARD\_NUMBER\_OF\_STRINGS = 1;

MIN\_FILE\_WAY\_SIZE = 5;

FILE\_KEY = 1;

CONSOLE\_KEY = 2;

Type

TMassive = Array [0 .. STANDARD\_NUMBER\_OF\_STRINGS] Of String;

//text information output block

Procedure ConditionOutput();

Begin

Writeln('The program determines the position number K ', #13#10,

'of the occurrence of the first line in the second.', #13#10,

'If there are no matches, returns -1.', #13#10);

End;

Procedure WorkWayConditionOutput();

Begin

Writeln('Where will we work through: ', #13#10#9, 'File: ', FILE\_KEY,

' Console: ', CONSOLE\_KEY, #13#10);

End;

Procedure FileRestriction();

Begin

Write(#13#10, '\*the first number in the file is the number ', #13#10,

'of the occurrence the index of which you want to find,', #13#10,

'and after that the substring and the string\*', #13#10,

'Write way to your file: ');

End;

//choice of direction

Function ChoosingWorkWay(): Integer;

Var

Path: Integer;

IsCorrect: Boolean;

IsCorrectPath: Boolean;

Begin

WorkWayConditionOutput();

Path := 0;

IsCorrect := False;

Write('Please write were we should work: ');

Repeat

IsCorrectPath := False;

Try

Readln(Path);

IsCorrectPath := True;

Except

Write('Error. You should write a number. Try again: ');

End;

If (Path <> CONSOLE\_KEY) And (Path <> FILE\_KEY) And IsCorrectPath Then

Write('Error method. Try again: ')

Else

If (IsCorrectPath) Then

IsCorrect := True;

Until IsCorrect;

ChoosingWorkWay := Path;

End;

//input and check path to the file

Function PathCondition(Way: String): Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := False;

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

Write('Write .txt file. Try again: ')

Else

IsCorrect := True;

PathCondition := IsCorrect;

End;

Function InputPath(): String;

Var

Way: String;

IsCorrect: Boolean;

Begin

Repeat

Readln(Way);

IsCorrect := PathCondition(Way);

Until IsCorrect;

InputPath := Way;

End;

Function InputPathToTheFile(): String;

Var

FileWay: String;

IsCorrect: Boolean;

Begin

IsCorrect := False;

FileWay := '';

Repeat

FileWay := InputPath();

If (Not FileExists(FileWay)) Then

Write('Can not open a file. Try write another way: ')

Else

IsCorrect := True;

Until IsCorrect;

InputPathToTheFile := FileWay;

End;

//input from file

Function InputKFromFile(FileWay: String): Integer;

Var

K: Integer;

MyFile: TextFile;

Begin

Try

AssignFile(MyFile, FileWay);

Try

Reset(MyFile);

Readln(MyFile, K);

If K < MIN\_K Then

Begin

Write('Min position number is ', MIN\_K, '. Try again: ');

K := ERR\_VALUE\_OF\_K;

End;

Finally

Close(MyFile);

End;

Except

Write('First string is natural number. Try again: ');

K := ERR\_VALUE\_OF\_K;

End;

InputKFromFile := K;

End;

Procedure InputStringFromFile(Var MyFile: TextFile; Var Str: TMassive);

Var

Counter: Char;

I: Integer;

Begin

For I := 0 To STANDARD\_NUMBER\_OF\_STRINGS Do

Begin

Repeat

Read(MyFile, Counter);

If (Counter <> #$D) And (Counter <> #$1A) Then

Str[I] := Str[I] + Counter

Else

Read(MyFile, Counter);

Until (Counter = #$A) Or (Counter = #$1A);

End;

End;

Procedure SettingTheCursor(Var MyFile: TextFile);

Var

BufferInt: Integer;

BufferChar: Char;

Begin

Read(MyFile, BufferInt);

Read(MyFile, BufferChar);

Read(MyFile, BufferChar);

End;

Function CheckEndOfFile(Var MyFile: TextFile): Boolean;

Var

Res: Boolean;

Begin

If Not SeekEOF(MyFile) Then

Begin

Write('In file should be only 1 number and 2 strings. Try again: ');

CheckEndOfFile := False;

End

Else

CheckEndOfFile := True;

End;

Procedure SysOfInputStringsFromFile(Var MyFile: TextFile; Var Str: TMassive);

Begin

SettingTheCursor(MyFile);

InputStringFromFile(MyFile, Str);

End;

//input from console

Function CheckKCondition(K: Integer; IsCorrectInput: Boolean): Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

If Not IsCorrectInput Then

IsCorrect := False;

If (K < MIN\_K) And (IsCorrect) Then

Begin

Write('Min position number is ', MIN\_K, '. Try again: ');

IsCorrect := False;

End;

CheckKCondition := IsCorrect;

End;

Function InputKFromConsole(): Integer;

Var

K: Integer;

IsCorrect, IsCorrectInput: Boolean;

Begin

K := 0;

Write('The position numbers of which occurrence you want to find: ');

Repeat

IsCorrectInput := False;

Try

Readln(K);

IsCorrectInput := True;

Except

Write('First string is natural number. Try again: ');

End;

IsCorrect := CheckKCondition(K, IsCorrectInput);

Until IsCorrect;

InputKFromConsole := K;

End;

Function InputStringFromConsole(): String;

Var

Str: String;

Current: Char;

IsCorrect: Boolean;

Begin

Str := '';

Repeat

Read(Current);

If Current <> #13 Then

Str := Str + Current;

Until Current = #13;

Read(Current);

InputStringFromConsole := Str;

End;

Function IsCorrectInput(Str1, Str2: String; IsItEndOfFile: Boolean): Boolean;

Begin

If (Str1 = '') Or (Str2 = '') Then

Begin

Write('Bad strings input. Try again: ');

IsCorrectInput := False;

End

Else

IsCorrectInput := IsItEndOfFile;

End;

Procedure SysOfInputStringsFromConsole(Var Str: TMassive);

Begin

Write('Write your first string: ');

Str[0] := InputStringFromConsole();

Write('Write your second string: ');

Str[1] := InputStringFromConsole();

End;

//search for result

Function IsStringsEqual(Str1, Str2: String; I: Integer): Boolean;

Var

J: Integer;

IsCorrect: Boolean;

Begin

IsCorrect := True;

For J := 2 To Length(Str1) - 1 Do

If ((Str2[I + J - 1] <> Str1[J]) And IsCorrect) Then

IsCorrect := False;

IsStringsEqual := IsCorrect;

End;

Function CalculationOfTheResult(K: Integer; Str1, Str2: String): Integer;

Var

Res, I: Integer;

IsCorrect: Boolean;

Begin

Res := -1;

For I := 1 To Length(Str2) - 1 Do

If (Str2[I] = Str1[1]) Then

Begin

IsCorrect := IsStringsEqual(Str1, Str2, I);

If IsCorrect Then

K := K - 1;

If ((K = 0) And IsCorrect) Then

Res := I;

End;

CalculationOfTheResult := Res;

End;

//output systeme

Procedure OutputFromFile(Result: Integer);

Var

IsCorrect: Boolean;

FileWay: String;

MyFile: TextFile;

Begin

IsCorrect := False;

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

Repeat

FileWay := InputPathToTheFile();

AssignFile(MyFile, FileWay);

Try

Try

Append(MyFile);

ReWrite(MyFile);

Writeln(MyFile, Result);

Write('Cheack your file.');

IsCorrect := True;

Finally

Close(MyFile);

End;

Except

Write('Bad output file. Try again: ');

End;

Until IsCorrect;

End;

Procedure OutputFromConsole(Result: Integer);

Begin

Write(Result, #13#10);

End;

Procedure ResultOutput(Result: Integer);

Var

Path: Integer;

Begin

Writeln('You need to choose where to write information from.');

Path := ChoosingWorkWay();

If (Path = CONSOLE\_KEY) Then

OutputFromConsole(Result)

Else

OutputFromFile(Result);

End;

//block of distributive functions

Function InputFileWay(Path: Integer): String;

Begin

If Path = CONSOLE\_KEY Then

InputFileWay := ''

Else

InputFileWay := InputPathToTheFile();

End;

Function KInput(Path: Integer; FileWay: String): Integer;

Begin

If Path = CONSOLE\_KEY Then

KInput := InputKFromConsole()

Else

KInput := InputKFromFile(FileWay);

End;

Function IsCorrectStringsInput(Path: Integer; FileWay: String;

Var Str: TMassive; K: Integer): Boolean;

Var

IsCorrect: Boolean;

IsItEndOfFile: Boolean;

MyFile: TextFile;

Begin

IsCorrect := True;

If K = ERR\_VALUE\_OF\_K Then

IsCorrect := False;

If (Path = CONSOLE\_KEY) And IsCorrect Then

SysOfInputStringsFromConsole(Str)

Else

If IsCorrect Then

Begin

IsItEndOfFile := True;

Try

AssignFile(MyFile, FileWay);

Try

Reset(MyFile);

SysOfInputStringsFromFile(MyFile, Str);

IsItEndOfFile := CheckEndOfFile(MyFile);

Finally

Close(MyFile);

End;

Except

IsCorrect := False;

Write('Bad strings input from file. Try again: ');

End;

IsCorrect := IsCorrectInput(Str[0], Str[1], IsItEndOfFile);

End;

IsCorrectStringsInput := IsCorrect;

End;

Function InputSystem(Var Str: TMassive): Integer;

Var

Path, K, I: Integer;

IsCorrect: Boolean;

FileWay: String;

Begin

Writeln(#13#10, 'You need to choose where to read information from.');

Path := ChoosingWorkWay();

If Path = FILE\_KEY Then

FileRestriction();

Repeat

FileWay := InputFileWay(Path);

For I := 0 To STANDARD\_NUMBER\_OF\_STRINGS Do

Str[I] := '';

K := KInput(Path, FileWay);

IsCorrect := IsCorrectStringsInput(Path, FileWay, Str, K);

Until IsCorrect;

InputSystem := K;

End;

Var

Str: TMassive;

K, Result: Integer;

Begin

ConditionOutput();

K := InputSystem(Str);

Result := CalculationOfTheResult(K, Str[0], Str[1]);

ResultOutput(Result);

Readln;

End.

Код программы на **C++**:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

const int MIN\_K = 1;

const int ERR\_VALUE\_OF\_K = -1;

const int VALUE\_OF\_DEFAULT\_RESULT = -1;

const int STANDARD\_NUMBER\_OF\_STRINGS = 2;

const int MIN\_FILE\_WAY\_SIZE = 5;

const int FILE\_KEY = 1;

const int CONSOLE\_KEY = 2;

//text information output block

void conditionOutput()

{

cout << "The program determines the position number K \n"

"of the occurrence of the first line in the second.\n"

"If there are no matches, returns -1.\n\n";

}

void workWayConditionOutput()

{

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

FILE\_KEY << " Console: " << CONSOLE\_KEY << endl << endl;

}

void fileRestriction()

{

cout << "\n\*the first number in the file is the number\n"

"of the occurrence the index of which you want to find,\n"

"and after that the substring and the string\*\n"

"Write way to your file: ";

}

// choice of direction

int choosingWorkWay()

{

workWayConditionOutput();

int path = 0;

bool isIncorrect = true;

cout << "Please write were we should work: ";

do

{

cin >> path;

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

{

cerr << "Error. You should write a natural number.

Try again: ";

cin.clear();

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

}

else

{

if (path == CONSOLE\_KEY || path == FILE\_KEY)

{

isIncorrect = false;

}

else

{

cerr << "Error method. Try again: ";

}

}

} while (isIncorrect);

return path;

}

// input and check path to the file

bool pathCondition(string way)

{

if (way.size() < MIN\_FILE\_WAY\_SIZE)

{

cerr << "The path is too short. Try again: ";

return false;

}

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

if (bufstr != ".txt")

{

cerr << "Write .txt file. Try again: ";

return false;

}

return true;

}

string inputPath()

{

string way;

bool isIncorrect;

do

{

cin >> way;

isIncorrect = !pathCondition(way);

} while (isIncorrect);

return way;

}

bool isCanOpenFile(string way, ios\_base::openmode mode)

{

fstream file(way, mode);

file.close();

return file.good();

}

string inputPathToTheFile()

{

string fileWay;

bool isIncorrect = true;

do

{

fileWay = inputPath();

if (!isCanOpenFile(fileWay, ios::in))

{

cerr << "Can't open a file. Try write another way: ";

}

else

{

isIncorrect = false;

}

} while (isIncorrect);

return fileWay;

}

// input from file

int inputKFromFile(string fileWay)

{

int k;

ifstream file(fileWay, ios::in);

file >> k;

if (file.fail())

{

cerr << "First string is natural number. Try again: ";

k = ERR\_VALUE\_OF\_K;

}

else if (k < MIN\_K)

{

cerr << "Min position number is " << MIN\_K << ". Try again: ";

k = ERR\_VALUE\_OF\_K;

}

file.close();

return k;

}

void inputStringFromFile(ifstream& file, string\*& str)

{

char counter;

int i = 0;

while (file.get(counter))

{

if (counter != '\n')

{

str[i] += counter;

}

else

{

i++;

}

}

}

void settingTheCursor(ifstream& file)

{

int bufferInt;

char nextString;

file >> bufferInt;

file.get(nextString);

}

bool checkEndOfFile(ifstream& file)

{

if (!file.eof()) {

cerr << "In file should be only 1 number and 2 strings. Try again: ";

return false;

}

return true;

}

void sysOfInputStringsFromFile(ifstream& file, string\*& str)

{

settingTheCursor(file);

inputStringFromFile(file, str);

}

/// input from console

bool checkKCondition(int k)

{

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

{

cerr << "First string is natural number. Try again: ";

cin.clear();

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

}

else if (k < MIN\_K)

{

cerr << "Min position number is " << MIN\_K << ". Try again: ";

}

else

{

return false;

}

return true;

}

int inputKFromConsole()

{

int k;

cout << "The position numbers of which occurrence you want to find: ";

bool isIncorrect;

do

{

cin >> k;

isIncorrect = checkKCondition(k);

} while (isIncorrect);

return k;

}

string inputStringFromConsole() {

string str = "";

char current;

bool isCorrect = true;

while (isCorrect && cin.get(current))

{

if (current != '\n')

{

str += current;

}

else

{

isCorrect = false;

}

}

return str;

}

bool isCorrectInput(string str1, string str2, bool isItEndOfFile)

{

if (str1 == "" || str2 == "")

{

cerr << "Bad strings input. Try again: ";

return false;

}

return isItEndOfFile;

}

void sysOfInputStringsFromConsole(string\*& str) {

cout << "Write your first string: ";

str[0] = inputStringFromConsole();

cout << "Write your second string: ";

str[1] = inputStringFromConsole();

}

/// search for result

bool isStringsEqual(string str1, string str2, int i) {

for (int j = 1; j < str1.size(); j++)

{

if (str2[i + j] != str1[j])

{

return false;

}

}

return true;

}

int calculationOfTheResult(int k, string str1, string str2)

{

if (str2.size() < str1.size())

{

return VALUE\_OF\_DEFAULT\_RESULT;

}

for (int i = 0; i < str2.size(); i++)

{

if (str2[i] == str1[0])

{

bool isCorrect = isStringsEqual(str1, str2, i);

if (--k == 0 && isCorrect)

{

return i + 1;

}

}

}

return VALUE\_OF\_DEFAULT\_RESULT;

}

/// output systeme

void outputFromFile(int result)

{

bool isIncorrect = true;

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

do

{

string fileWay = inputPathToTheFile();

if (isCanOpenFile(fileWay, ios::out))

{

ofstream file(fileWay, ios::out);

file << result;

file.close();

cout << "Check your file.";

isIncorrect = false;

}

else

{

cerr << "Can't open a file. Try write another way: ";

}

} while (isIncorrect);

}

void outputFromConsole(int result)

{

cout << result << endl;

}

void resultOutput(int result)

{

cout << "\nYou need to choose where to write information from.\n";

int path = choosingWorkWay();

path == CONSOLE\_KEY ? outputFromConsole(result) : outputFromFile(result);

}

// block of distributive functions

string inputFileWay(int path)

{

return path == CONSOLE\_KEY ? "" : inputPathToTheFile();

}

int kInput(int path, string fileWay)

{

return path == CONSOLE\_KEY ? inputKFromConsole() : inputKFromFile(fileWay);

}

bool isCorrectStringsInput(int path, string fileWay, string\*& str, int k) {

if (k == ERR\_VALUE\_OF\_K)

{

return false;

}

if (path == CONSOLE\_KEY)

{

sysOfInputStringsFromConsole(str);

return true;

}

else

{

ifstream file(fileWay, ios::in);

sysOfInputStringsFromFile(file, str);

bool isItEndOfFile = checkEndOfFile(file);

file.close();

return isCorrectInput(str[0], str[1], isItEndOfFile);

}

}

// input distributive

int inputSystem(string\*& str)

{

cout << "\nYou need to choose where to read information from.\n";

int path = choosingWorkWay();

if (path == FILE\_KEY)

{

fileRestriction();

}

int k;

bool isIncorrect;

do

{

string fileWay = inputFileWay(path);

for (int i = 0; i < STANDARD\_NUMBER\_OF\_STRINGS; i++)

{

str[i] = "";

}

k = kInput(path, fileWay);

isIncorrect = !isCorrectStringsInput(path, fileWay, str, k);

} while (isIncorrect);

return k;

}

// main

int main()

{

conditionOutput();

string\* str = new string[STANDARD\_NUMBER\_OF\_STRINGS];

int k = inputSystem(str);

int result = calculationOfTheResult(k, str[0], str[1]);

delete[] str;

resultOutput(result);

return 0;

}

Код программы на **Java**:

package lab3;  
  
import java.io.File;  
import java.io.FileWriter;  
import java.util.Scanner;  
  
public class Lab1 {  
 static Scanner in = new Scanner(System.in);  
 static final int MIN\_K = 1;  
 static final int ERR\_VALUE\_OF\_K = -1;  
 static final int VALUE\_OF\_DEFAULT\_RESULT = -1;  
 static final int STANDARD\_NUMBER\_OF\_STRINGS = 2;  
 static final int MIN\_FILE\_WAY\_SIZE = 5;  
 static final int FILE\_KEY = 1;  
 static final int CONSOLE\_KEY = 2;  
 //text information output block  
 static void conditionOutput()  
 {  
 System.out.println("""  
 The program determines the position number K  
 of the occurrence of the first line in the second.  
 If there are no matches, returns -1.  
 """);  
 }  
 static void workWayConditionOutput()  
 {  
 System.out.printf("""  
 Where will we work through:  
 File: %d Console: %d  
   
 """, FILE\_KEY, CONSOLE\_KEY);  
 }  
 static void fileRestriction()  
 {  
 System.out.print("""  
   
 \*the first number in the file is the number  
 of the occurrence the index of which you want to find,  
 and after that the substring and the string\*  
 Write way to your file:\s""");  
 }  
 // choice of direction  
 static int choosingWorkWay()  
 {  
 workWayConditionOutput();  
  
 int path = 0;  
 boolean isIncorrect = true;  
 System.out.print("Please write were we should work: ");  
 do  
 {  
 boolean isCorrect = false;  
 try {  
 path = Integer.parseInt(in.nextLine());  
 isCorrect = true;  
 } catch (Exception error) {  
 System.err.print("Error. You should write a natural number.

Try again: ");  
 }  
 if ((path == CONSOLE\_KEY || path == FILE\_KEY))  
 {  
 isIncorrect = false;  
 }  
 else if (isCorrect)  
 {  
 System.err.print("Error method. Try again: ");  
 }  
 } while (isIncorrect);  
  
 return path;  
 }  
 // input and check path to the file  
 static boolean pathCondition(String way)  
 {  
 if (way.length() < MIN\_FILE\_WAY\_SIZE)  
 {  
 System.err.print("The path is too short. Try again: ");  
 return false;  
 }  
 String bufstr = way.substring(way.length() - 4);  
 if (!bufstr.equals(".txt"))  
 {  
 System.err.print("Write .txt file. Try again: ");  
 return false;  
 }  
 return true;  
 }  
 static String inputPath()  
 {  
 String way;  
 boolean isIncorrect;  
 do  
 {  
 way = in.nextLine();  
 isIncorrect = !pathCondition(way);  
 } while (isIncorrect);  
  
 return way;  
 }  
 static boolean isCanOpenFile(String way)  
 {  
 File file = new File(way);  
 return file.canRead();  
 }  
 static String inputPathToTheFile()  
 {  
 String fileWay;  
 boolean isIncorrect = true;  
  
 do  
 {  
 fileWay = inputPath();  
 if (!isCanOpenFile(fileWay))  
 {  
 System.err.print("Can't open a file. Try write another way: ");  
 }  
 else  
 {  
 isIncorrect = false;  
 }  
 } while (isIncorrect);  
  
 return fileWay;  
 }  
 // input from file  
 static int inputKFromFile(String fileWay) {  
 int k = 0;  
 boolean isCorrect = false;  
  
 try (Scanner fileScanner = new Scanner(new File(fileWay))){  
 k = Integer.parseInt(fileScanner.nextLine());  
 isCorrect = true;  
 } catch(Exception error)  
 {  
 System.err.print("First string is natural number. Try again: ");  
 }  
  
 if (k < MIN\_K && isCorrect)  
 {  
 System.err.printf("Min position number is %d. Try again: ", MIN\_K);  
 k = ERR\_VALUE\_OF\_K;  
 }  
  
 return k;  
 }  
 static void inputStringFromFile(Scanner fileScanner, String[] str)  
 {  
 str[0] = fileScanner.nextLine();  
 str[1] = fileScanner.nextLine();  
 }  
 static void settingTheCursor(Scanner fileScanner)  
 {  
 fileScanner.nextLine();  
 }  
 static boolean checkEndOfFile(Scanner fileScanner)  
 {  
 if (fileScanner.hasNextLine()) {  
 System.err.print("In file should be only 1 number and 2 strings.

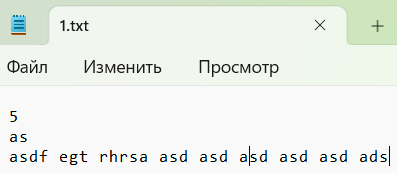
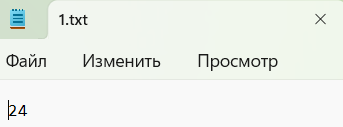
Try again: ");  
 return false;  
 }  
  
 return true;  
 }  
 static void sysOfInputStringsFromFile(Scanner fileScanner, String[] str)  
 {  
 settingTheCursor(fileScanner);  
 inputStringFromFile(fileScanner, str);  
 }  
 /// input from console  
 static boolean checkKCondition(int k, boolean isCorrect)  
 {  
 if (!isCorrect)  
 return true;  
  
 if (k < MIN\_K)  
 {  
 System.err.printf("Min position number is %d. Try again: ", MIN\_K);  
 }  
 else  
 {  
 return false;  
 }  
  
 return true;  
 }  
 static int inputKFromConsole()  
 {  
 int k = 0;  
 System.out.print("The position numbers of which occurrence you want to

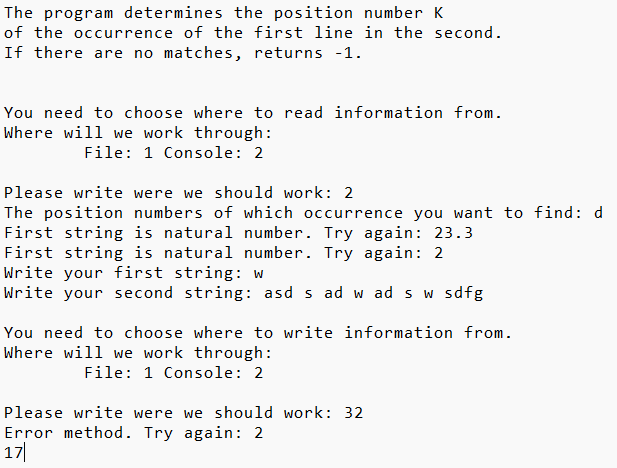
find: ");  
 boolean isIncorrect;  
 do  
 {  
 boolean isCorrect = false;  
 try {  
 k = Integer.parseInt(in.nextLine());  
 isCorrect = true;  
 } catch(Exception error){  
 System.err.print("First string is natural number. Try again: ");  
 }  
 isIncorrect = checkKCondition(k, isCorrect);  
 } while (isIncorrect);  
  
 return k;  
 }  
 static String inputStringFromConsole() {  
 return in.nextLine();  
 }  
 static boolean isCorrectInput(String str1, String str2, boolean isItEndOfFile)  
 {  
 if (str1.isEmpty() || str2.isEmpty())  
 {  
 System.err.print("Bad strings input. Try again: ");  
 return false;  
 }  
  
 return isItEndOfFile;  
 }  
 static void sysOfInputStringsFromConsole(String[] str) {  
 System.out.print("Write your first string: ");  
 str[0] = inputStringFromConsole();  
 System.out.print("Write your second string: ");  
 str[1] = inputStringFromConsole();  
 }  
 /// search for result  
 static boolean isStringsEqual(String str1, String str2, int i) {  
 for (int j = 1; j < str1.length(); j++)  
 {  
 if (str2.charAt(i + j) != str1.charAt(j))  
 {  
 return false;  
 }  
 }  
  
 return true;  
 }  
 static int calculationOfTheResult(int k, String str1, String str2)  
 {  
 if (str2.length() < str1.length())  
 {  
 return VALUE\_OF\_DEFAULT\_RESULT;  
 }  
  
 for (int i = 0; i < str2.length(); i++)  
 {  
 if (str1.charAt(0) == str2.charAt(i))  
 {  
 boolean isCorrect = isStringsEqual(str1, str2, i);  
 if (--k == 0 && isCorrect)  
 {  
 return i + 1;  
 }  
 }  
 }  
  
 return VALUE\_OF\_DEFAULT\_RESULT;  
 }  
 /// output  
 static void outputFromFile(int result)  
 {  
 boolean isIncorrect = true;  
 System.out.print("Write way to your file: ");  
 do  
 {  
 String fileWay = inputPathToTheFile();  
 File file = new File(fileWay);  
 StringBuilder builder;  
 if (file.canWrite())  
 {  
 try  
 {  
 FileWriter writer = new FileWriter(fileWay);  
 builder = new StringBuilder();  
 builder.append(result);  
 writer.write(builder.toString());  
 writer.close();  
 System.out.print("Check your file.");  
 isIncorrect = false;  
 } catch (Exception error)  
 {  
 System.err.print("Can't write in this file. Try again: ");  
 }  
 }  
 else  
 {  
 System.err.print("Can't open a file. Try write another way: ");  
 }  
 } while (isIncorrect);  
 }  
 static void outputFromConsole(int result)  
 {  
 System.out.println(result);  
 }  
 static void resultOutput(int result)  
 {  
 System.out.println("You need to choose where to write information from.");  
 int path = choosingWorkWay();  
 if (path == CONSOLE\_KEY)  
 {  
 outputFromConsole(result);  
 } else  
 {  
 outputFromFile(result);  
 }  
 }  
 // block of distributive functions  
 static String inputFileWay(int path)  
 {  
 return path == CONSOLE\_KEY ? "" : inputPathToTheFile();  
 }  
 static int kInput(int path, String fileWay)  
 {  
 return path == CONSOLE\_KEY ? inputKFromConsole()

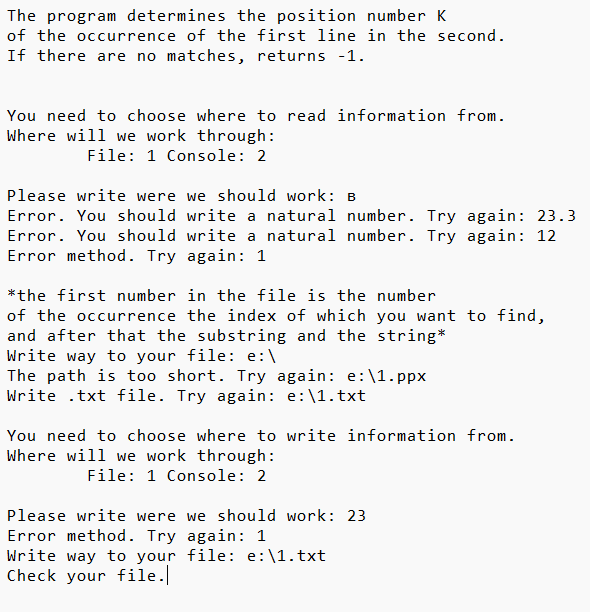
: inputKFromFile(fileWay);  
 }  
 static boolean isCorrectStringsInput(int path, String fileWay,

String[] str, int k) {  
 if (k == ERR\_VALUE\_OF\_K)  
 {  
 return false;  
 }  
  
 if (path == CONSOLE\_KEY)  
 {  
 sysOfInputStringsFromConsole(str);  
 return true;  
 }  
 else  
 {  
 boolean isItEndOfFile = true;  
 try (Scanner fileScanner = new Scanner(new File(fileWay))){  
 sysOfInputStringsFromFile(fileScanner, str);  
 isItEndOfFile = checkEndOfFile(fileScanner);  
 } catch(Exception error){  
 System.err.print("Error file read opening. Try again: ");  
 }  
 return isCorrectInput(str[0], str[1], isItEndOfFile);  
 }  
 }  
 // input distributive  
 static int inputSystem(String[] str)  
 {  
 System.out.println("You need to choose where to read information from.");  
 int path = choosingWorkWay();  
  
 if (path == FILE\_KEY)  
 {  
 fileRestriction();  
 }  
  
 int k;  
 boolean isIncorrect;  
 do  
 {  
 String fileWay = inputFileWay(path);  
 for (int i = 0; i < STANDARD\_NUMBER\_OF\_STRINGS; i++)  
 str[i] = "";  
 k = kInput(path, fileWay);  
 isIncorrect = !isCorrectStringsInput(path, fileWay, str, k);  
 } while (isIncorrect);  
  
 return k;  
 }  
 // main  
 public static void main(String[] args)  
 {  
 conditionOutput();  
  
 String[] str = new String[STANDARD\_NUMBER\_OF\_STRINGS];  
 int k = inputSystem(str);  
  
 int result = calculationOfTheResult(k, str[0], str[1]);  
  
 resultOutput(result);  
  
 in.close();  
 }  
}

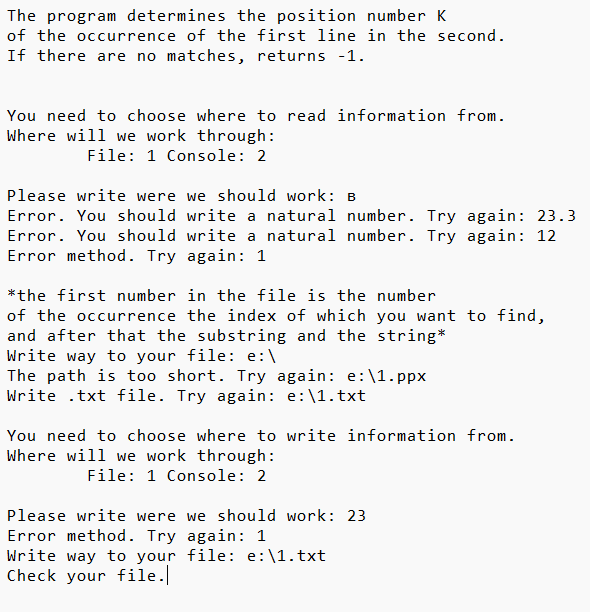
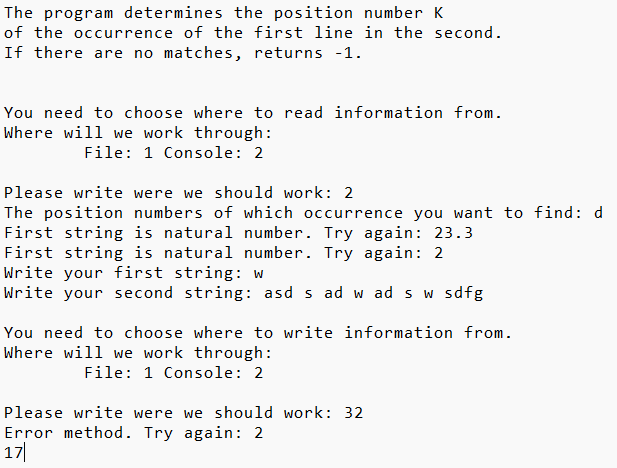
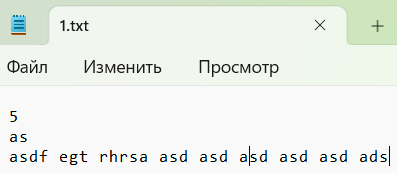
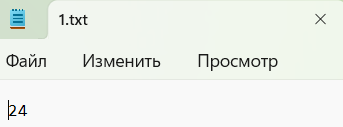
Результат на **Delphi**:



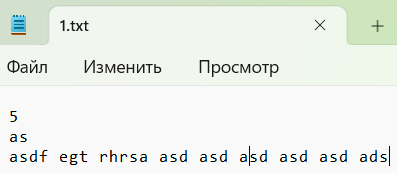
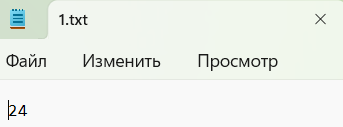


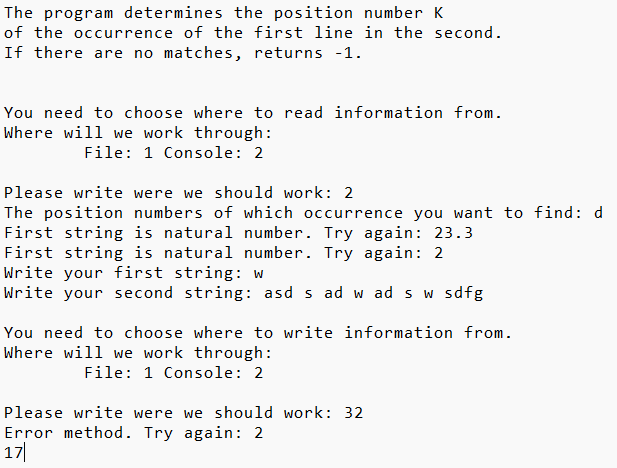


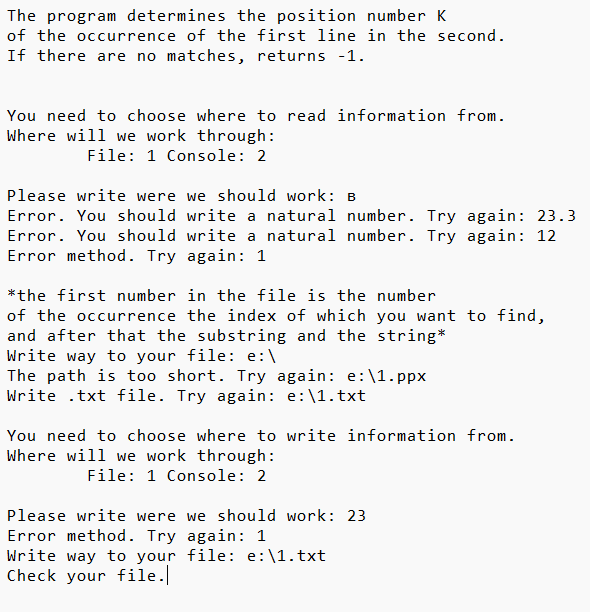
Результат на **Java**:



Результат на **C++**:







Блок-схема:

