УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.1

по предмету «Основы алгоритмизации и программирования»

Вариант 5

Выполнил:

Бражалович А. И.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Определить номер позиции последнего вхождения строки st1 в строку st2. Если такого нет, возвратить 0.

**Код программы Delphi:**

Program Lab3\_1;

Uses

System.SysUtils;

Const

MIN\_LENGTH = 1;

MAX\_LENGTH = 1000;

Procedure PrintTask();

Begin

WriteLn('Данная программа находит место последнего вхождения первой строки во

вторую:' + #13#10);

End;

Function ChooseFileInput() : Boolean;

Var

IsFileInput: Integer;

IsCorrect, Choose: Boolean;

Begin

IsFileInput := 0;

IsCorrect := False;

Choose := False;

Repeat

WriteLn('Вы хотите вводить строки 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 CheckLength(Str: String) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := True;

If (Length(Str) < MIN\_LENGTH) Or (Length(Str) > MAX\_LENGTH) Then

Begin

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

IsCorrect := False;

End;

CheckLength := IsCorrect;

End;

Function ReadPathToFile() : String;

Var

PathToFile: String;

IsCorrect: Boolean;

Begin

PathToFile := '';

IsCorrect := True;

Repeat

IsCorrect := True;

Write('Введите путь к файлу с расширением .txt:');

ReadLn(PathToFile);

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

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

IsCorrect: Boolean;

Begin

IsCorrect := False;

Reset(F);

If EOF(F) Then

IsCorrect := True;

CloseFile(F);

IsEmpty := IsCorrect;

End;

Function IsRightFileString(Var F: TextFile) : Boolean;

Var

IsCorrect: Boolean;

Begin

IsCorrect := False;

Reset(F);

Readln(F);

If EOF(F) Then

IsCorrect := True;

Readln(F);

If Not EOF(F) Then

IsCorrect := True;

CloseFile(F);

IsRightFileString := IsCorrect;

End;

Procedure GetFileNormalReading(Var F: TextFile);

Var

PathToFile: String;

IsCorrect: Boolean;

Begin

PathToFile := '';

Repeat

IsCorrect := True;

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 IsRightFileString(F) Then

Begin

IsCorrect := False;

WriteLn('Количество строк не совпадает!');

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 ReadFileString(Var F: TextFile) : String;

Var

Str: String;

Begin

Str := '';

ReadLn(F, Str);

ReadFileString := Str;

End;

Function ReadConsoleString(Num: Integer) : String;

Var

Str: String;

IsCorrect: Boolean;

Begin

Str := '';

IsCorrect := False;

Repeat

Write('Введите строку номер ', Num, ' [', MIN\_LENGTH, ' : ', MAX\_LENGTH, ']: ');

Readln(Str);

IsCorrect := CheckLength(Str);

Until IsCorrect;

ReadConsoleString := Str;

End;

Procedure ReadString(Var Str1: String; Var Str2: String);

Var

RF: TextFile;

IsCorrect: Boolean;

Begin

Repeat

IsCorrect := True;

If ChooseFileInput() Then

Begin

GetFileNormalReading(RF);

Reset(RF);

Str1 := ReadFileString(RF);

Str2 := ReadFileString(RF);

CloseFile(RF);

End

Else

Begin

Str1 := ReadConsoleString(1);

Str2 := ReadConsoleString(2);

End;

If Length(Str1) > Length(Str2) Then

Begin

IsCorrect := False;

Writeln('Длинна не соответствует условию!');

End;

Until IsCorrect;

End;

Function FindLastOccurrence(Str1, Str2: String): Integer;

Var

LengthOfStr2, LengthOfStr1, IndOfStr1, IndOfStr2, Place: Integer;

EndOfStr1: Boolean;

Begin

LengthOfStr1 := Length(Str1);

LengthOfStr2 := Length(Str2);

Place := 0;

IndOfStr1 := 0;

IndOfStr2 := 0;

EndOfStr1 := False;

While (LengthOfStr2 > 0) And (Place = 0) Do

Begin

If Str1[LengthOfStr1] = Str2[LengthOfStr2] Then

Begin

IndOfStr1 := LengthOfStr1;

IndOfStr2 := LengthOfStr2;

While (IndOfStr1 > 0) And (IndOfStr2 > 0) And (Str1[IndOfStr1] = Str2[IndOfStr2]) Do

Begin

Dec(IndOfStr1);

Dec(IndOfStr2);

End;

If (IndOfStr1 <> 0) Then

EndOfStr1 := True;

If (Not EndOfStr1) Then

Place := IndOfStr2 + 1;

End;

Dec(LengthOfStr2);

End;

FindLastOccurrence := Place;

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(Place: Integer);

Begin

Writeln('Номер позиции последнего вхождения строки st1 в строку st2 = ', Place);

End;

Procedure PrintFileResult(Var F: TextFile; Place: Integer);

Begin

Append(F);

WriteLn(F);

Writeln(F, 'Номер позиции последнего вхождения строки st1 в строку st2 = ', Place);

CloseFile(F);

End;

Procedure PrintResult(Place: Integer);

Var

WF: TextFile;

Begin

If ChooseFileOutput() Then

Begin

GetFileNormalWriting(WF);

PrintFileResult(WF, Place);

End

Else

PrintConsoleResult(Place);

End;

Var

Str1, Str2: String;

Place: Integer;

Begin

PrintTask();

ReadString(Str1, Str2);

Place := FindLastOccurrence(Str1, Str2);

PrintResult(Place);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <string.h>

#include <fstream>

#include <string>

using namespace std;

const int MIN\_LENGTH = 1;

const int MAX\_LENGTH = 1000;

void printTask()

{

cout << "Данная программа находит место последнего вхождения первой строки во вторую:\n\n";

}

bool chooseFileInput()

{

int isFileInput;

bool isCorrect, choose;

isFileInput = 0;

choose = false;

do {

isCorrect = true;

cout << "Вы хотите вводить строки 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 checkLength(string str)

{

bool isCorrect;

isCorrect = true;

if ((str.length() < MIN\_LENGTH) || (str.length() > MAX\_LENGTH))

{

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

isCorrect = false;

}

return isCorrect;

}

string readPathFile()

{

string pathToFile;

bool isCorrect;

pathToFile = "";

do

{

isCorrect = true;

cout << "Введите путь к файлу с расширением .txt: ";

cin >> pathToFile;

if (pathToFile.size() < 5 || pathToFile[pathToFile.length() - 4] != '.' || pathToFile[pathToFile.length() - 3] != 't' || pathToFile[pathToFile.length() - 2] != 'x' || pathToFile[pathToFile.length() - 1] != 't')

{

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 isRightFileString(string pathToFile)

{

bool isCorrect;

isCorrect = false;

ifstream file(pathToFile);

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

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

isCorrect = true;

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

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

isCorrect = true;

file.close();

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 && isEmpty(pathToFile))

{

isCorrect = false;

cout << "Файл пуст!\n";

}

if (isCorrect && !isRightFileString(pathToFile))

{

isCorrect = false;

cout << "Количество строк не совпадает с условием!\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);

}

string readFileString(ifstream& file)

{

string str;

getline(file, str);

return str;

}

string readConsoleString(int num)

{

string str;

bool isCorrect;

str = ' ';

do

{

isCorrect = true;

cout << "Введите строку номер " << num << " [" << MIN\_LENGTH << ":" << MAX\_LENGTH << "]: ";

cin >> str;

isCorrect = checkLength(str);

} while (!isCorrect);

return str;

}

void readString(string& str1, string& str2)

{

string pathToFile;

bool isCorrect;

pathToFile = "";

do

{

isCorrect = true;

if (chooseFileInput())

{

getFileNormalReading(pathToFile);

ifstream file(pathToFile);

str1 = readFileString(file);

str2 = readFileString(file);

file.close();

}

else

{

str1 = readConsoleString(1);

str2 = readConsoleString(2);

}

if (str1.length() > str2.length())

{

isCorrect = false;

cout << "Длинна не соответсвует условию!\n";

}

} while (!isCorrect);

}

int findLastOccurrence(string& str1, string& str2)

{

int lengthOfStr1;

int lengthOfStr2;

int indOfStr1;

int indOfStr2;

int place;

bool endOfStr1;

lengthOfStr1 = str1.length();

lengthOfStr2 = str2.length();

place = 0;

indOfStr1 = 0;

indOfStr2 = 0;

endOfStr1 = false;

while ((lengthOfStr2 > 0) && (place == 0))

{

if (str1[lengthOfStr1 - 1] == str2[lengthOfStr2 - 1])

{

indOfStr1 = lengthOfStr1 - 1;

indOfStr2 = lengthOfStr2 - 1;

endOfStr1 = false;

while ((indOfStr1 > 0) && (indOfStr2 > 0) && (str1[indOfStr1] == str2[indOfStr2]))

{

indOfStr1--;

indOfStr2--;

}

if (indOfStr1 != 0)

endOfStr1 = true;

if (!endOfStr1)

place = indOfStr2 + 1;

}

lengthOfStr2--;

}

return place;

}

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(int place)

{

cout << "Номер позиции последнего вхождения строки st1 в строку st2 = " << place;

}

void printFileResult(string pathToFile, int place)

{

fstream file(pathToFile, ios::app);

file << "\n";

file << "Номер позиции последнего вхождения строки st1 в строку st2 = " << place;

file.close();

}

void printResult(int place)

{

string pathToFile;

pathToFile = "";

if (chooseFileOutput())

{

getFileNormalWriting(pathToFile);

printFileResult(pathToFile, place);

}

else

printConsoleResult(place);

}

int main()

{

string str1;

string str2;

int place;

str1 = " ";

str2 = " ";

setlocale(LC\_ALL, "RU");

printTask();

readString(str1, str2);

place = findLastOccurrence(str1, str2);

printResult(place);

return 0;

}

**Код программы Java:**

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.util.Scanner;

import java.io.File;

import java.io.BufferedWriter;

import java.io.OutputStreamWriter;

public class Main {

public static final int MIN\_LENGTH = 1;

public static final int MAX\_LENGTH = 1000;

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

static File file;

public static void printTask() {

System.out.println("Данная программа находит место последнего вхождения первой строки во вторую:\n");

}

public static boolean chooseFileInput() {

int isFileInput;

boolean isCorrect, choose;

isFileInput = 0;

choose = true;

do {

System.out.println("Вы хотите вводить строки 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 checkLength(String str) {

boolean isCorrect;

isCorrect = true;

if (str.isEmpty() || str.length() > MAX\_LENGTH) {

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

isCorrect = false;

}

return isCorrect;

}

public static String readPathFile() {

boolean isCorrect;

String pathToFile;

isCorrect = true;

pathToFile = " ";

do {

System.out.println("Введите путь к файлу с расширением .txt: ");

pathToFile = scanConsole.nextLine();

if ((pathToFile.length() < 5) || (pathToFile.charAt(pathToFile.length() - 4) != '.') || (pathToFile.charAt(pathToFile.length() - 3) != 't') || (pathToFile.charAt(pathToFile.length() - 2) != 'x') || (pathToFile.charAt(pathToFile.length() - 1) != 't')) {

isCorrect = false;

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

}

} while (!isCorrect);

return pathToFile;

}

public static boolean isExists() {

boolean isCorrect;

isCorrect = false;

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 isRightFileString() {

boolean isCorrect;

isCorrect = false;

try (Scanner scanFile = new Scanner(file)) {

scanFile.nextLine();

if (!scanFile.hasNext())

isCorrect = true;

scanFile.nextLine();

if (!scanFile.hasNext())

isCorrect = true;

} catch (Exception e) {}

return isCorrect;

}

public static void getFileNormalReading() {

boolean isCorrect;

do {

isCorrect = true;

file = new File(readPathFile());

if (!isExists()) {

isCorrect = false;

System.out.println("Проверьте корректность ввода пути к файлу!\n");

}

if (isCorrect && !isAbleToReading()) {

isCorrect = false;

System.out.println("Файл закрыт для чтения!\n");

}

if (isCorrect && isEmpty()) {

isCorrect = false;

System.out.println("Файл пуст!\n");

}

if (isCorrect && !isRightFileString()) {

isCorrect = false;

System.out.println("Количество строк не совпадает с условием!\n");

}

} while (!isCorrect);

}

public static void getFileNormalWriting() {

boolean isCorrect;

do {

file = new File(readPathFile());

isCorrect = true;

if (!isExists()) {

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 String readFileString(Scanner scanFile) {

String str;

str = scanFile.nextLine();

return str;

}

public static String readConsoleString(int num) {

boolean isCorrect;

String str;

str = "";

do {

isCorrect = true;

System.out.println("Введите строку номер " + num + " [" + MIN\_LENGTH + ":" + MAX\_LENGTH + "]: ");

str = scanConsole.nextLine();

isCorrect = checkLength(str);

} while (!isCorrect);

return str;

}

public static String[] readString() {

String[] strings = new String[2];

boolean isCorrect;

do {

isCorrect = true;

if (chooseFileInput()) {

getFileNormalReading();

try (Scanner scanFile = new Scanner(file)) {

strings[0] = readFileString(scanFile);

strings[1] = readFileString(scanFile);

} catch (Exception e) {

}

} else {

strings[0] = readConsoleString(1);

strings[1] = readConsoleString(2);

}

if (strings[0].length() > strings[1].length()) {

isCorrect = false;

System.out.println("Длинна не соответсвует условию!\n");

}

}while (!isCorrect);

return strings;

}

public static int findLastOccurrence(String str1, String str2) {

int lengthOfStr1;

int lengthOfStr2;

int indOfStr1;

int indOfStr2;

int place;

boolean endOfStr1;

lengthOfStr1 = str1.length();

lengthOfStr2 = str2.length();

place = 0;

indOfStr1 = 0;

indOfStr2 = 0;

endOfStr1 = false;

while ((lengthOfStr2 > 0) && (place == 0))

{

if (str1.charAt(lengthOfStr1 - 1) == str2.charAt(lengthOfStr2 - 1))

{

indOfStr1 = lengthOfStr1 - 1;

indOfStr2 = lengthOfStr2 - 1;

endOfStr1 = false;

while ((indOfStr1 > 0) && (indOfStr2 > 0) && (str1.charAt(indOfStr1) == str2.charAt(indOfStr2)))

{

indOfStr1--;

indOfStr2--;

}

if (indOfStr1 != 0)

endOfStr1 = true;

if (!endOfStr1)

place = indOfStr2 + 1;

}

lengthOfStr2--;

}

return place;

}

public static boolean chooseFileOutput() {

int isFileOutput;

boolean isCorrect, choose;

isFileOutput = 0;

choose = true;

do {

System.out.println("Вы хотите выводить результат 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(int place) {

System.out.print("Номер позиции последнего вхождения строки st1 в строку st2 = " + place);

}

public static void printFileResult(int place) {

try (BufferedWriter writer = new BufferedWriter(

new OutputStreamWriter(Files.newOutputStream(file.toPath()), StandardCharsets.UTF\_8))) {

writer.write("Номер позиции последнего вхождения строки st1 в строку st2 = " + place);

} catch (Exception e) {}

}

public static void printResult(int place) {

if (chooseFileOutput()) {

getFileNormalWriting();

printFileResult(place);

}

else

printConsoleResult(place);

}

public static void main(String[] args) {

boolean nonGrowing;

String str1;

String str2;

String[] strings;

int place;

printTask();

strings = readString();

str1 = strings[0];

str2 = strings[1];

place = findLastOccurrence(str1, str2);

printResult(place);

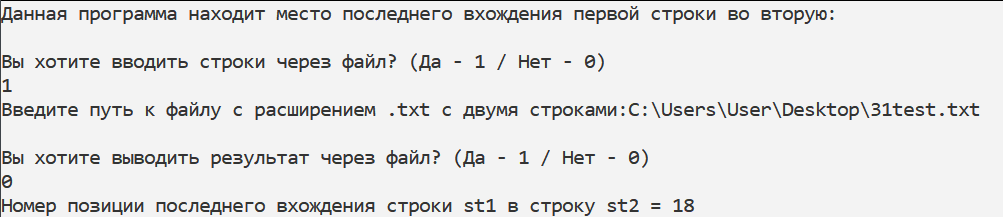
scanConsole.close();

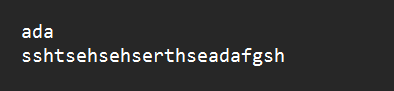
}

}

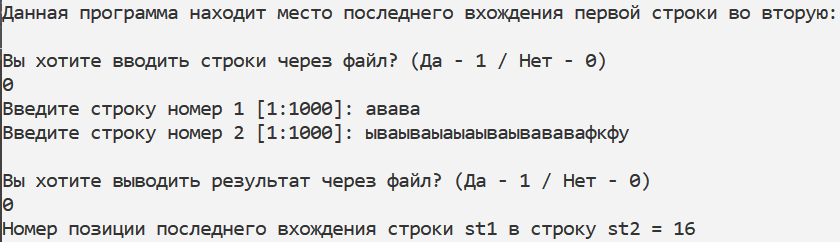
**Скриншоты:**

**Delphi:**

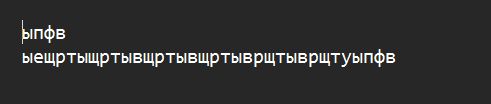
****

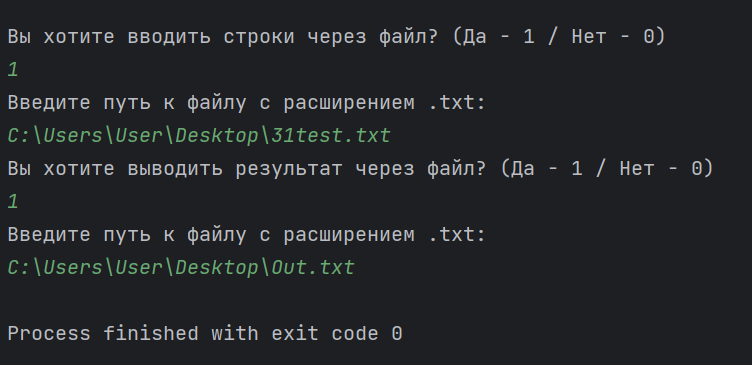
****

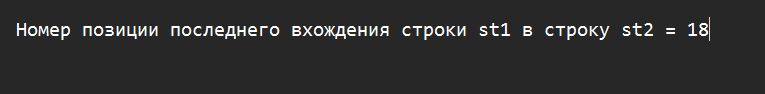
**C++:**

****

**Java:**

****

****

****

**Блок-схема:**











