УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.1

по предмету «Основы алгоритмизации и программирования»

Вариант 20

Выполнил:

Захвей И.В.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Задана строка символов, состоящая из букв, цифр, точек, символов «+»

и «-». Выделить подстроку, состоящую из цифр, соответствующую

целому числу (т.е. начинается со знака «+» или «-» и внутри подстроки

нет букв и точки).

**Код программы Delphi:**

Program Project3;

Uses

System.SysUtils;

Type

ERRORS\_CODE = (SUCCESS, INCORRECT\_DATA, EMPTY\_LINE, NOT\_TXT, FILE\_NOT\_EXIST,

INCORRECT\_DATA\_FILE, A\_LOT\_OF\_DATA\_FILE, FILE\_NOT\_AVAILABLE);

Const

DIGITS = ['0' .. '9'];

ERRORS: Array [0 .. 7] Of String = ('Successfull', 'Data is not correct',

'Line is empty, please be careful',

'This is not a .txt file',

'This file is not exist',

'Data in file is not correct',

'There is only one line in file should be',

'File is can not be opened');

Procedure PrintInf();

Begin

Writeln('Program selects a substring consisting of digits corresponding ',

'to an integer', #10#13, '(starts with a "+" or "-" ',

'and there are no letters and dots inside the substring)');

End;

Function GetNumFromLine(Line: String): String;

Var

IsNumbNotExist: Boolean;

I, Size: Integer;

Numb: String;

Begin

Numb := 'not exist';

IsNumbNotExist := True;

Size := Length(Line) + 1;

I := 1;

While I < Size Do

Begin

If (IsNumbNotExist And ((Line[I] = '+') Or (Line[I] = '-'))) Then

Begin

Numb := Line[I];

Inc(I);

While ((I < Size) And (Line[I] In DIGITS)) Do

Begin

Numb := Numb + Line[I];

Inc(I);

End;

IsNumbNotExist := Length(Numb) = 1;

If IsNumbNotExist Then

Numb := 'not exist';

End

Else

Inc(I);

End;

GetNumFromLine := Numb;

End;

Function InpChoice(Var Choice: Integer): Integer;

Var

Err: Integer;

ChoiceStr: String;

Begin

Err := Ord(SUCCESS);

Readln(ChoiceStr);

If (ChoiceStr = '1') Or (ChoiceStr = '2') Then

Choice := StrToInt(ChoiceStr)

Else If (Length(ChoiceStr) > 0) Then

Err := Ord(INCORRECT\_DATA)

Else

Err := Ord(EMPTY\_LINE);

InpChoice := Err;

End;

Function InpValidLine(Var Line: String): Integer;

Var

Err: Integer;

Begin

Err := Ord(SUCCESS);

Readln(Line);

If Length(Line) = 0 Then

Err := Ord(EMPTY\_LINE);

InpValidLine := Err;

End;

Function UserChoice(): Integer;

Var

Choice: Integer;

Err: Integer;

Begin

Writeln('Choose a way of input/output of data', #13#10, '1 -- Console',

#13#10, '2 -- File');

Repeat

Err := InpChoice(Choice);

If (Err > 0) then

Writeln(ERRORS[Err], #10#13, 'Please, enter again');

Until (Err = 0);

UserChoice := Choice;

End;

Procedure InputFromConsole(Var Line: String);

Var

Err: Integer;

Begin

Writeln('Enter the line');

Repeat

Err := InpValidLine(Line);

If (Err > 0) then

Writeln(ERRORS[Err], #10#13, 'Please, enter again');

Until (Err = 0);

End;

Function FileAvailable(Name: String; ForReset: Boolean): Integer;

Var

Err: Integer;

MyFile: TextFile;

Begin

Err := Ord(SUCCESS);

AssignFile(MyFile, Name);

If ForReset Then

Try

Try

Reset(MyFile);

Finally

CloseFile(MyFile);

End;

Except

Err := Ord(FILE\_NOT\_AVAILABLE);

End

Else

Try

Try

Rewrite(MyFile);

Finally

CloseFile(MyFile);

End;

Except

Err := Ord(FILE\_NOT\_AVAILABLE);

End;

FileAvailable := Err;

End;

Function GetLastFourChar(Line: String): String;

Var

Start, I, Size: Integer;

LastFourChar: String;

Begin

Size := Length(Line);

Start := Size - 3;

For I := Start To Size Do

LastFourChar := LastFourChar + Line[I];

GetLastFourChar := LastFourChar;

End;

Function FileTxt(Name: String): Integer;

Var

Err: Integer;

LastFourChar: String;

Begin

Err := Ord(SUCCESS);

If Length(Name) > 4 Then

Begin

LastFourChar := GetLastFourChar(Name);

If LastFourChar <> '.txt' Then

Err := Ord(NOT\_TXT);

End

Else

Err := Ord(NOT\_TXT);

FileTxt := Err;

End;

Function FileExist(Name: String): Integer;

Var

Err: Integer;

Begin

Err := Ord(SUCCESS);

If Not FileExists(Name) Then

Err := Ord(FILE\_NOT\_EXIST);

FileExist := Err;

End;

Function GetFileName(ForReset: Boolean): String;

Var

IsCorrect: Boolean;

ErrExist, ErrTxt, ErrAvailable: Integer;

FileName: String;

Begin

Repeat

IsCorrect := True;

Readln(FileName);

ErrExist := FileExist(FileName);

ErrTxt := FileTxt(FileName);

If (ErrExist > 0) Then

Begin

Writeln(ERRORS[ErrExist]);

IsCorrect := False;

End

Else If (ErrTxt > 0) Then

Begin

Writeln(ERRORS[ErrTxt]);

IsCorrect := False;

End;

If ((ErrExist = 0) And (ErrTxt = 0)) Then

Begin

ErrAvailable := FileAvailable(FileName, ForReset);

If (ErrAvailable > 0) Then

Begin

Writeln(ERRORS[ErrAvailable]);

IsCorrect := False;

End;

End;

Until IsCorrect;

GetFileName := FileName;

End;

Function ReadFile(Var Line: String; Name: String): Integer;

Var

Err: Integer;

InfFile: TextFile;

Begin

AssignFile(InfFile, Name);

Reset(InfFile);

Err := Ord(SUCCESS);

Read(InfFile, Line);

If Not EoF(InfFile) Then

Err := Ord(A\_LOT\_OF\_DATA\_FILE);

If Length(Line) = 0 Then

Err := Ord(EMPTY\_LINE);

CloseFile(InfFile);

ReadFile := Err;

End;

Procedure InputFromFile(Var Line: String);

Var

Err: Integer;

FileName: String;

Begin

Writeln('Enter full path to file');

Repeat

FileName := GetFileName(True);

Err := ReadFile(Line, FileName);

If (Err > 0) then

Writeln(ERRORS[Err], #10#13, 'Please, enter full path again');

Until (Err = 0);

Writeln('Reading is successfull');

End;

Function InputInf(): String;

Var

Choice, Num: Integer;

Line: String;

Begin

Choice := UserChoice();

If (Choice = 1) Then

InputFromConsole(Line)

Else

InputFromFile(Line);

InputInf := Line;

End;

Procedure OutputInConsole(Line, Num: String);

Begin

Writeln('Default line', #13#10, Line);

Writeln('Substring', #13#10, Num);

End;

Procedure OutputInFile(Line, Num: String);

Var

FileName: String;

MyFile: TextFile;

Begin

Writeln('Enter full path to file');

FileName := GetFileName(False);

AssignFile(MyFile, FileName);

Rewrite(MyFile);

Writeln(MyFile, 'Default line', #13#10, Line);

Writeln(MyFile, 'Substring', #13#10, Num);

CloseFile(MyFile);

Writeln('Writing is successfull');

End;

Procedure OutputInf(Line, Num: String);

Var

Choice: Integer;

Begin

Choice := UserChoice();

If (Choice = 1) Then

OutputInConsole(Line, Num)

Else

OutputInFile(Line, Num);

End;

Var

Num, Line: String;

Begin

PrintInf();

Line := InputInf();

Num := GetNumFromLine(Line);

OutputInf(Line, Num);

Readln;

End.

**Код программы С++:**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

enum ErrorsCode

{

SUCCESS,

INCORRECT\_DATA,

EMPTY\_LINE,

NOT\_TXT,

FILE\_NOT\_EXIST,

A\_LOT\_OF\_DATA\_FILE,

};

const string ERRORS[] = { "Successfull",

"Data is not correct, or number is too large\n",

"Line is empty, please be careful\n",

"This is not a .txt file\n",

"This file is not exist\n",

"There is only one line in file should be\n"};

void printInf()

{

cout << "Program selects a substring consisting of digits corresponding "

<< "to an integer \n(starts with a '+' or '-' "

<< "and there are no letters and dots inside the substring\n";

}

string getNumFromLine(string line)

{

string numb;

int i, size;

bool isNumbNotExist;

isNumbNotExist = true;

size = line.length();

i = 0;

numb = "not exist";

while (i < size)

{

if (isNumbNotExist && (line[i] == '+' || line[i] == '-'))

{

numb = line[i];

i++;

while (i < size && isdigit(line[i]))

numb += line[i++];

isNumbNotExist = numb.length() == 1;

if (isNumbNotExist)

numb = "not exist";

}

else

++i;

}

return numb;

}

int inpChoice(int& choice)

{

int err;

string choiceStr;

err = SUCCESS;

getline(cin, choiceStr);

if (choiceStr == "1" || choiceStr == "2")

choice = stoi(choiceStr);

else

err = choiceStr.length() > 0 ? INCORRECT\_DATA : EMPTY\_LINE;

return err;

}

int userChoice()

{

int choice;

cout << "Choose a way of input/output of data\n"

<< "1 -- Console\n"

<< "2 -- File\n";

int err;

do

{

err = inpChoice(choice);

if (err > 0)

cout << ERRORS[err] << "Please, enter again\n";

} while (err > 0);

return choice;

}

int inpValidLine(string& line)

{

int err;

err = SUCCESS;

getline(cin, line);

if (line.length() == 0)

err = EMPTY\_LINE;

return err;

}

void inputFromConsole(string& line)

{

cout << "Enter the line\n";

int err;

do

{

err = inpValidLine(line);

if (err > 0)

cout << ERRORS[err] << "Please, enter again\n";

} while (err > 0);

}

int readFile(string &line, string fileName)

{

int err = SUCCESS;

bool isCorrect = true;

ifstream file(fileName);

getline(file, line);

if (!file.eof())

err = A\_LOT\_OF\_DATA\_FILE;

if (line.length() == 0)

err = EMPTY\_LINE;

file.close();

return err;

}

int isFileExist(string nameOfFile)

{

int err;

ifstream file(nameOfFile);

err = file.is\_open() ? SUCCESS : FILE\_NOT\_EXIST;

file.close();

return err;

}

string getLastFourChar(string line)

{

string lastFourChar;

int start, i, size;

size = line.length();

start = size - 4;

for (i = start; i < size; i++)

lastFourChar += line[i];

return lastFourChar;

}

int thisIsTxtFile(string& fileName)

{

int err = SUCCESS;

string lastFourChar;

if (fileName.length() > 4)

{

lastFourChar = getLastFourChar(fileName);

if (lastFourChar != ".txt")

err = NOT\_TXT;

}

else

err = NOT\_TXT;

return err;

}

string getFileName()

{

bool isIncorrect;

string name;

int errExist, errTxt;

cout << "Enter full path to file\n";

do

{

getline(cin, name);

errExist = isFileExist(name);

errTxt = thisIsTxtFile(name);

isIncorrect = false;

if (errTxt > 0)

{

cout << ERRORS[errTxt];

isIncorrect = true;

}

else if (errExist > 0)

{

cout << ERRORS[errExist];

isIncorrect = true;

}

} while (isIncorrect);

return name;

}

void inputFromFile(string &line)

{

string fileName;

int err;

do

{

fileName = getFileName();

err = readFile(line, fileName);

if (err > 0)

cout << ERRORS[err] << "Please, enter full path again\n";

} while (err > 0);

cout << "Reading is successfull\n";

}

string inputInf()

{

string line;

int choice = userChoice();

if (choice == 1)

inputFromConsole(line);

else

inputFromFile(line);

return line;

}

void outputInConsole(string line, string num)

{

cout << "Default line\n" << line << endl;

cout << "Substring\n" << num << endl;

}

void outputInFile(string line, string num)

{

string fileName = getFileName();

ofstream file(fileName);

file << "Default line\n" << line << endl;

file << "Substring\n" << num << endl;

cout << "Writing is successfull\n";

file.close();

}

void outputInf(string line, string num)

{

int choice = userChoice();

if (choice == 1)

outputInConsole(line, num);

else

outputInFile(line, num);

}

int main()

{

string num, line;

printInf();

line = inputInf();

num = getNumFromLine(line);

outputInf(line, num);

return 0;

}

**Код программы Java:**

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Scanner;

public class Main {

static enum Codes {

SUCCESS,

INCORRECT\_DATA,

EMPTY\_LINE,

NOT\_TXT,

FILE\_NOT\_EXIST,

A\_LOT\_OF\_DATA\_FILE,

IN\_OUT\_FILE\_EXCEPTION;

}

static final String[] ERRORS ={"Successfull",

"Data is not correct, or number is too large",

"Line is empty, please be careful",

"This is not a .txt file",

"This file is not exist",

"There is only one line in file should be",

"Exception with output/input from the file"};

static void printInf() {

System.out.println("Program selects a substring consisting of digits

corresponding" + "to an integer\n(starts with a '+' or '-' "

+ "and there are no letters and dots inside the substring");

}

static String getNumFromLine(String line) {

String numb;

int i, size;

boolean isNumbNotExist;

isNumbNotExist = true;

size = line.length();

i = 0;

numb = "not exist";

while (i < size) {

if (isNumbNotExist && (line.charAt(i) == '+' || line.charAt(i) == '-')) {

numb = String.valueOf(line.charAt(i));

i++;

while (i < size && Character.isDigit(line.charAt(i)))

numb += line.charAt(i++);

isNumbNotExist = numb.length() == 1;

if (isNumbNotExist)

numb = "not exist";

}

else

++i;

}

return numb;

}

static int inputChoice(Scanner input,int[] choice){

int err;

String choiceStr;

err = Codes.SUCCESS.ordinal();

choiceStr = input.nextLine();

if (choiceStr.equals("1") || choiceStr.equals("2")) {

choice[0] = Integer.parseInt(choiceStr);

} else {

err = choiceStr.isEmpty() ? Codes.EMPTY\_LINE.ordinal() :

Codes.INCORRECT\_DATA.ordinal();

}

return err;

}

static int userChoice(Scanner input) {

int[] choice = {0};

System.out.println("Choose a way of input/output of data\n"

+ "1 -- Console\n"

+ "2 -- File");

int err;

do {

err = inputChoice(input, choice);

if (err > 0) {

System.err.println(ERRORS[err]);

System.out.println("Please, enter again");

}

} while (err > 0);

return choice[0];

}

static int inpValidLine(String[] line, Scanner input) {

int err;

err = Codes.SUCCESS.ordinal();

line[0] = input.nextLine();

if (line[0].isEmpty()) {

err = Codes.EMPTY\_LINE.ordinal();

}

return err;

}

static int readFile(String[] line, String fileName) throws IOException{

int errCode;

Path path = Paths.get(fileName);

Scanner file = new Scanner(path);

errCode = Codes.SUCCESS.ordinal();

line[0] = file.nextLine();

if (line[0].isEmpty())

errCode = Codes.EMPTY\_LINE.ordinal();

if (file.hasNextLine()) {

errCode = Codes.A\_LOT\_OF\_DATA\_FILE.ordinal();

}

file.close();

return errCode;

}

static int isFileExist(String fileName) {

File file = new File(fileName);

int err;

err = file.exists() ? Codes.SUCCESS.ordinal() : Codes.FILE\_NOT\_EXIST.ordinal();

return err;

}

static int thisIsTxtFile(String fileName) {

int err;

err = fileName.endsWith(".txt") ? Codes.SUCCESS.ordinal() :

Codes.NOT\_TXT.ordinal();

return err;

}

static String getFileName(Scanner input) {

boolean isIncorrect;

String fileName;

int errTxt, errExist;

System.out.println("Enter full path to file");

do {

isIncorrect = false;

fileName = input.nextLine();

errTxt = thisIsTxtFile(fileName);

errExist = isFileExist(fileName);

if (errTxt > 0) {

isIncorrect = true;

System.err.println(ERRORS[errTxt]);

}

else if (errExist > 0) {

isIncorrect = true;

System.err.println(ERRORS[errExist]);

}

} while (isIncorrect);

return fileName;

}

static void inputFromFile(Scanner input, String[] line){

int err;

do {

String fileName = getFileName(input);

try {

err = readFile(line, fileName);

} catch (IOException e) {

err = Codes.IN\_OUT\_FILE\_EXCEPTION.ordinal();

}

if (err > 0) {

System.err.println(ERRORS[err]);

System.out.println("Please, enter full path again");

}

} while (err > 0);

System.out.println("Reading is successfull");

}

static void inputFromConsole(Scanner input, String[] line) {

System.out.println("Enter the line");

int err;

do {

err = inpValidLine(line, input);

if (err > 0) {

System.err.println(ERRORS[err]);

System.out.println("Please, enter again");

}

} while (err > 0);

}

static String inputInf(Scanner input){

int choice = userChoice(input);

String[] line = {""};

if (choice == 1) {

inputFromConsole(input, line);

} else {

inputFromFile(input, line);

}

return line[0];

}

static void writeInConsole(String line, String num) {

System.out.println("Default line\n" + line);

System.out.println("Substring\n" + num);

}

static void writeInFile(Scanner input, String line, String num) throws IOException {

String fileName = getFileName(input);

PrintWriter file = new PrintWriter(fileName);

file.println("Default line\n" + line);

file.println("Substring\n" + num);

file.close();

System.out.println("Writing is successfull");

}

static void outputInf(String line, String num, Scanner input) throws IOException {

int choice = userChoice(input);

if (choice == 1) {

writeInConsole(line, num);

} else {

try {

writeInFile(input, line, num);

} catch (IOException e) {

throw new IOException(ERRORS[Codes.IN\_OUT\_FILE\_EXCEPTION.ordinal()]);

}

}

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

String line, num;

printInf();

line = inputInf(input);

num = getNumFromLine(line);

try {

outputInf(line, num, input);

} catch (Exception e) {

System.err.println(e.getMessage());

}

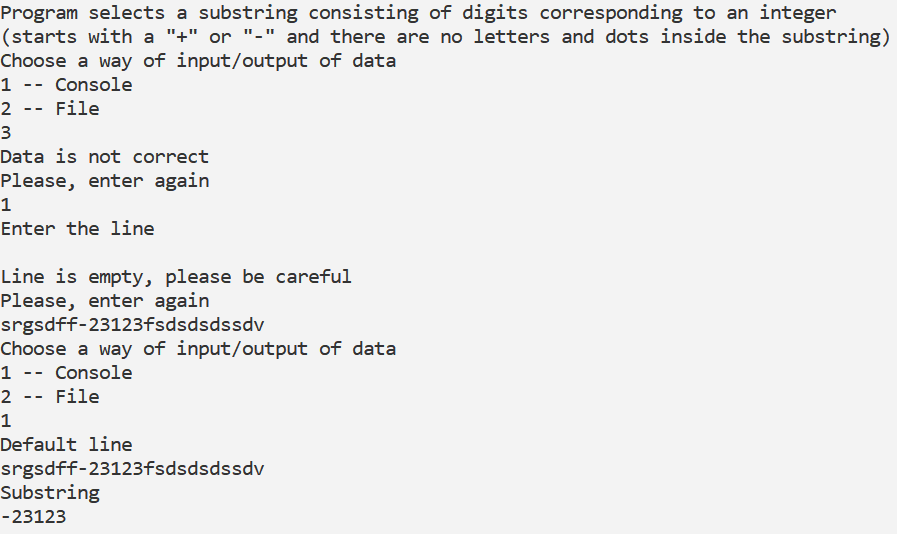
input.close();

}

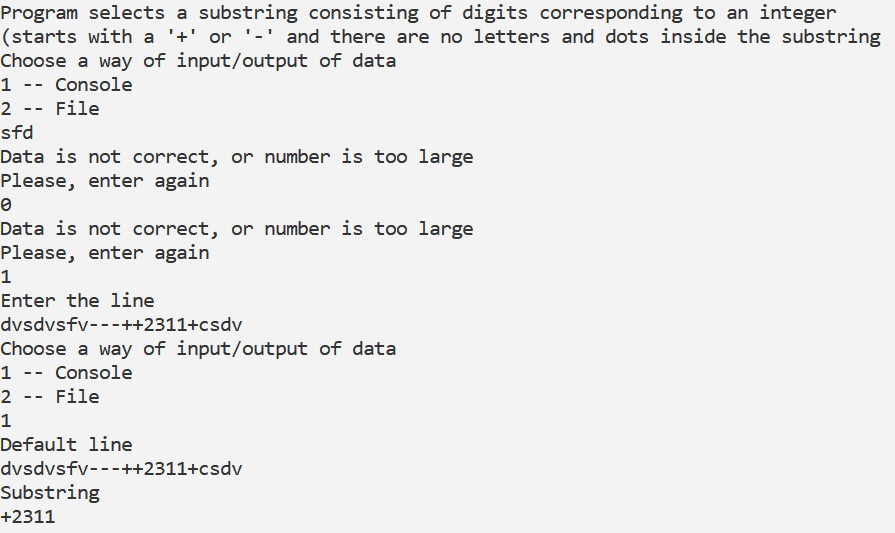
}

**Скриншоты:**

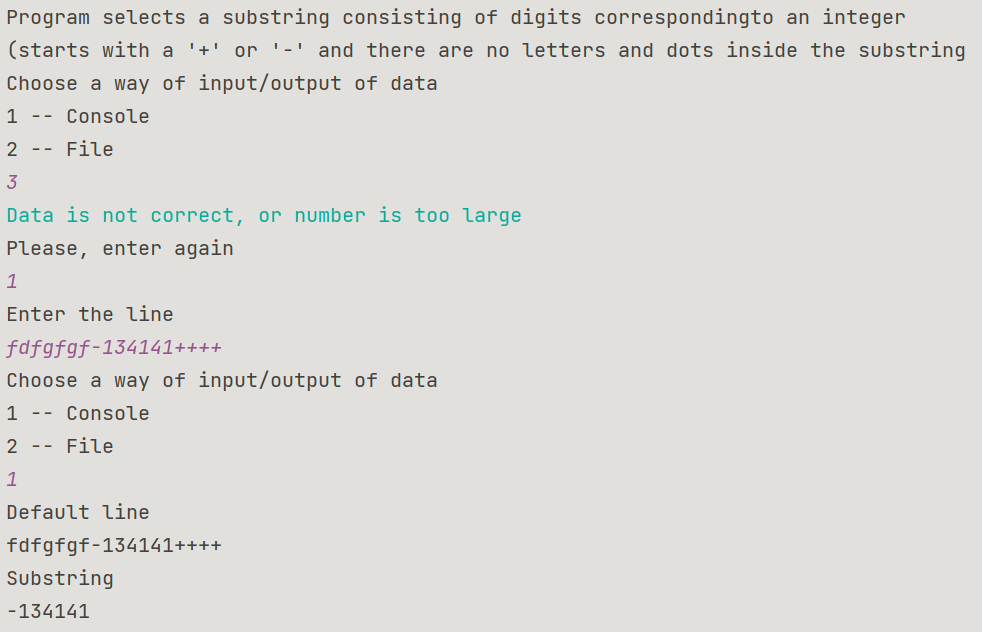
**Delphi:**

****

**C++:**

****

**Java:**

****

**Блок-схема:**



