УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.1

по предмету «Основы алгоритмизации и программирования»

Вариант 18

Выполнил:

Егоров А.С.

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 Task1;

{$APPTYPE CONSOLE}

{$R \*.res}

Uses

System.SysUtils;

// class reader

Type

TReader = Class

Public

Function InputString(): String; Virtual; Abstract;

Procedure EmptyStringMessage(); Virtual;

End;

Procedure TReader.EmptyStringMessage();

Begin

Writeln('Your string Empty! Try again.');

End;

// class consoleReader

Type

TConsoleReader = Class(TReader)

Public

Function InputString(): String; Override;

End;

Function TConsoleReader.InputString(): String;

Var

InputtedString: String;

IsCorrect: Boolean;

Begin

IsCorrect := False;

InputtedString := '';

Repeat

Writeln('Enter string:');

Readln(InputtedString);

If (InputtedString <> '') Then

IsCorrect := True

Else

EmptyStringMessage();

Until IsCorrect;

Result := InputtedString;

End;

// class FileReader

Type

TFileReader = Class(TReader)

Private

FileName: String;

InFile: TextFile;

FileStatus: Boolean;

Public

Function InputString(): String; Override;

Function IsFileGood(): Boolean;

Function IsFileTxt(): Boolean;

Function IsFileReadable(): Boolean;

End;

Function TFileReader.IsFileTxt(): Boolean;

Var

FileType: String;

Begin

FileType := FileName.Substring(FileName.Length - 4);

If FileType = '.txt' Then

FileStatus := True

Else

FileStatus := False;

Result := FileStatus;

End;

Function TFileReader.IsFileReadable(): Boolean;

Begin

Try

Reset(InFile);

FileStatus := True;

Except

FileStatus := False;

End;

CloseFile(InFile);

Result := FileStatus;

End;

Function TFileReader.IsFileGood(): Boolean;

Begin

FileStatus := False;

If (Not FileExists(FileName)) Then

Writeln('This file or the path to the file is specified incorrectly or does not

exist! Try again.')

Else If (Not IsFileTxt()) Then

Writeln('This file or path to the file isn', #39, 't .txt! Try again.')

Else If (Not IsFileReadable()) Then

Writeln('The program can', #39, 't read this file! Try again.')

Else

FileStatus := True;

Result := FileStatus;

End;

Function TFileReader.InputString: String;

Var

InputtedString: String;

IsCorrect: Boolean;

Begin

IsCorrect := False;

Repeat

Writeln('Enter the name of file in this directory or path to this file

including name of file:');

Readln(FileName);

AssignFile(InFile, FileName);

If IsFileGood() Then

Begin

Reset(InFile);

If Not Eof(InFile) Then

Begin

Read(InFile, InputtedString);

IsCorrect := True;

End

Else

EmptyStringMessage();

CloseFile(InFile);

End;

Until IsCorrect;

Result := InputtedString;

End;

// class Writer

Type

TWriter = Class

Public

Procedure OutputString(Str: String); Virtual; Abstract;

End;

// class ConsoleWriter

Type

TConsoleWriter = Class(TWriter)

Public

Procedure OutputString(Str: String); Override;

End;

Procedure TConsoleWriter.OutputString(Str: String);

Begin

Writeln('The number in the string is ', Str, '.');

End;

// class FileWriter

Type

TFileWriter = Class(TWriter)

Private

OutFile: TextFile;

FileName: String;

FileStatus: Boolean;

Public

Procedure OutputString(Str: String); Override;

Function IsFileGood(): Boolean;

Function IsFileTxt(): Boolean;

Function IsFileWritable(): Boolean;

End;

Function TFileWriter.IsFileTxt(): Boolean;

Var

FileType: String;

Begin

FileType := FileName.Substring(FileName.Length - 4);

If FileType = '.txt' Then

FileStatus := True

Else

FileStatus := False;

Result := FileStatus;

End;

Function TFileWriter.IsFileWritable(): Boolean;

Begin

Try

Rewrite(OutFile);

FileStatus := True;

Except

FileStatus := False;

End;

CloseFile(OutFile);

Result := FileStatus;

End;

Function TFileWriter.IsFileGood(): Boolean;

Begin

FileStatus := False;

If (Not FileExists(FileName)) Then

Writeln('This file or the path to the file is specified incorrectly or does not

exist! Try again.')

Else If (Not IsFileTxt()) Then

Writeln('This file or path to the file isn', #39, 't .txt! Try again.')

Else If (Not IsFileWritable()) Then

Writeln('The program can', #39, 't write from this file! Try again.')

Else

FileStatus := True;

Result := FileStatus;

End;

Procedure TFileWriter.OutputString(Str: String);

Var

IsCorrect: Boolean;

Begin

IsCorrect := False;

Repeat

Writeln('Enter the name of file in this directory or path to this file

including name of file:');

Readln(FileName);

AssignFile(OutFile, FileName);

If IsFileGood() Then

Begin

Rewrite(OutFile);

Writeln(OutFile, 'The number in the string is ', Str, '.');

Writeln('Answer has been wrote successfully.');

IsCorrect := True;

CloseFile(OutFile);

End;

Until IsCorrect;

End;

// other function

Function InputMethod(): TReader;

Var

IsCorrect: Boolean;

Choice: String;

Reader1: TReader;

Begin

Reader1 := Nil;

IsCorrect := False;

Choice := '';

Writeln('The program works with console input or files.');

Repeat

Writeln('To use console enter ', #39, 'console', #39, '.', #13#10,

'To use a file enter ', #39, 'file', #39, '.'#13#10,

'Enter what type you want to use: ');

Readln(Choice);

If Choice = 'console' Then

Begin

IsCorrect := True;

Reader1 := TConsoleReader.Create;

End

Else If Choice = 'file' Then

Begin

Reader1 := TFileReader.Create;

IsCorrect := True;

End

Else

Writeln('The word ', Choice, ' don', #39,

't match any of method to input the data.');

Until IsCorrect;

Result := Reader1;

End;

Function OutputMethod(): TWriter;

Var

IsCorrect: Boolean;

Choice: String;

Writer1: TWriter;

Begin

Writer1 := Nil;

IsCorrect := False;

Choice := '';

Writeln('The program works with console output or files.');

Repeat

Writeln('To use console enter ', #39, 'console', #39, '.', #13#10,

'To use a file enter ', #39, 'file', #39, '.'#13#10,

'Enter what type you want to use: ');

Readln(Choice);

If Choice = 'console' Then

Begin

Writer1 := TConsoleWriter.Create;

IsCorrect := True;

End

Else If Choice = 'file' Then

Begin

Writer1 := TFileWriter.Create;

IsCorrect := True;

End

Else

Writeln('The word ', Choice, ' don', #39,

't match any of method to output the data.');

Until IsCorrect;

Result := Writer1;

End;

Function FindNumberInString(Str: String): String;

Var

IsNumber, WasNumber: Boolean;

I, Size: Integer;

Number: String;

Begin

IsNumber := False;

WasNumber := False;

Number := '';

Size := Str.Length;

For I := 1 To Size Do

Begin

If (Not WasNumber) And ((Str[I] = '+') Or (Str[I] = '-')) Then

Begin

IsNumber := True;

Number := '';

Number := Number + Str[I];

End

Else If (IsNumber) And (47 < Ord(Str[I])) And (Ord(Str[I]) < 58) Then

Begin

Number := Number + Str[I];

WasNumber := True;

End

Else If IsNumber And WasNumber Then

Begin

IsNumber := False;

End

Else If Not WasNumber Then

IsNumber := False;

End;

If Not WasNumber Then

Number := '';

Result := Number;

End;

Var

Reader1: TReader;

Writer1: TWriter;

Answer: String;

Begin

Reader1 := InputMethod();

Answer := FindNumberInString(Reader1.InputString());

Writer1 := OutputMethod();

Writer1.OutputString(Answer);

// free memory

Reader1 := Nil;

Writer1 := Nil;

End.

**Код программы С++:**

#include<iostream>

#include<string>

#include<fstream>

class Reader

{

public:

virtual std::string inputString() = 0;

void emptyStringMessage();

protected:

};

void Reader::emptyStringMessage()

{

std::cout << "Your string Empty! Try again.\n";

}

class ConsoleReader : public Reader

{

public:

std::string inputString() override;

private:

};

std::string ConsoleReader::inputString()

{

std::string inputtedString = "\0";

bool isInCorrect = true;

do

{

std::cout << "Enter string:\n";

std::cin >> inputtedString;

if (inputtedString != "\0")

isInCorrect = false;

else

emptyStringMessage();

} while (isInCorrect);

return inputtedString;

}

class File

{

public:

File(std::string fileName, size\_t fileCode) : fileName(fileName), fileCode(fileCode)

{

if (fileCode == 0)

fileStream = new std::fstream(fileName, std::ios::in);

else

fileStream = new std::fstream(fileName);

}

~File()

{

delete fileStream;

};

bool isFileExist();

bool isFileText();

bool isFileWorking();

bool isNotEmpty();

bool isGood();

std::string getString();

void printString(const std::string);

private:

// code info

// 0 - file for reading

// 1 - file for writting

// 2 - file for reading and writting

//

size\_t fileCode;

std::string fileName;

std::fstream\* fileStream;

bool status = false;

};

bool File::isFileExist()

{

if (fileStream->is\_open())

{

fileStream->close();

return true;

}

else

std::cerr << "This file or the path to the file is specified incorrectly or

does not exist! Try again.\n";

return false;

}

bool File::isNotEmpty()

{

fileStream->open(fileName, std::ios::in);

if (fileStream->peek() != std::fstream::traits\_type::eof())

{

fileStream->close();

return true;

}

else

std::cerr << "This file empty! Try again.\n";

fileStream->close();

return false;

}

bool File::isFileText()

{

std::string type = fileName.substr(fileName.length() - (size\_t)(4));

if (type == ".txt")

return true;

else

std::cerr << "This file or path to the file isn't .txt! Try again.\n";

return false;

}

// if we can read(write) from(to) this file

bool File::isFileWorking()

{

fileStream->open(fileName);

if (fileStream->good())

{

fileStream->close();

return true;

}

else if (fileCode == 0)

std::cerr << "The program can't read from this file! Try again.\n";

else if (fileCode == 1)

std::cerr << "The program can't write down this file! Try again.\n";

else

std::cerr << "The program can't write down or read from this file! Try

again.\n";

fileStream->close();

return false;

}

bool File::isGood()

{

if (fileCode == 0 && this->isFileExist() && this->isFileText()

&& this->isFileWorking() && this->isNotEmpty())

status = true;

else if (fileCode > 0 && this->isFileExist() && this->isFileText()

&& this->isFileWorking())

status = true;

else

status = false;

return status;

}

std::string File::getString()

{

fileStream->open(fileName);

std::string inputtedString = "\0";

\*fileStream >> inputtedString;

fileStream->close();

return inputtedString;

}

void File::printString(const std::string str)

{

fileStream->open(fileName);

if (str == "\0")

\*fileStream << "Function didn't find number in string.\n";

else

\*fileStream << "The number in the string is " << str << std::endl;

std::cout << "Answer has been wrote successfully.\n";

}

class FileReader : public Reader

{

public:

std::string inputString() override;

private:

std::string fileName;

size\_t fileCode = (size\_t)0; // check info about code in class File

};

std::string FileReader::inputString()

{

File\* in = nullptr;

std::string inputtedString;

do

{

delete in;

std::cout << "Enter the name of file in this directory or path to this file

including name of file:\n";

std::cin >> fileName;

in = new File(fileName, fileCode);

} while (!in->isGood());

inputtedString = in->getString();

delete in;

return inputtedString;

}

class Writer

{

public:

virtual void outputString(std::string) = 0;

protected:

};

class ConsoleWriter : public Writer

{

public:

void outputString(std::string) override;

private:

};

void ConsoleWriter::outputString(std::string str)

{

if (str == "\0")

std::cout << "Function didn't find number in string.\n";

else

std::cout << "The number in the string is " << str << std::endl;

}

class FileWriter : public Writer

{

public:

FileWriter() : fileCode((size\_t)1)

{ };

void outputString(std::string) override;

private:

std::string fileName;

size\_t fileCode = (size\_t)1; // check info about code in class File

};

void FileWriter::outputString(std::string str)

{

File\* out = nullptr;

do

{

delete out;

std::cout << "Enter the name of file in this directory or path to this file

including name of file:\n";

std::cin >> fileName;

out = new File(fileName, fileCode);

} while (!out->isGood());

out->printString(str);

delete out;

}

std::string findNumberInString(std::string s)

{

bool isNumber = false;

bool wasNumber = false;

std::string number = "\0";

for (size\_t i = 0; i < s.length(); i++)

{

if (!wasNumber && (s[i] == '+' || s[i] == '-'))

{

isNumber = true;

number = "";

number += s[i];

}

else if (isNumber && std::isdigit(s[i]))

{

number += s[i];

wasNumber = true;

}

else if (isNumber && wasNumber)

{

isNumber = false;

}

else if (!wasNumber)

{

isNumber = false;

}

}

// if number == "-" or number == "+" we return empty number

return (wasNumber) ? number : "\0";

}

Reader\* inputMethod()

{

bool isIncorrect = true;

Reader\* reader = nullptr;

std::string choice = "\0";

// asking what the type user want to use

std::cout << "The program works with console input or files.\n";

do

{

std::cout << "To use console enter 'console'.\n"

<< "To use a file enter 'file'.\n"

<< "Enter what type you want to use: \n";

std::cin >> choice;

if (choice == "console")

{

reader = new ConsoleReader();

isIncorrect = false;

}

else if (choice == "file")

{

reader = new FileReader();

isIncorrect = false;

}

else // wrong input

std::cerr << "The word '" << choice << "' don't match any of method to

input the data.\n";

} while (isIncorrect);

return reader;

}

Writer\* outputMethod()

{

bool isIncorrect = true;

std::string choice = "\0";

Writer\* writter = nullptr;

// asking what the type user want to use

std::cout << "The program is ready to show answer.\n";

do

{

std::cout << "To output in console enter 'console'.\n"

<< "To output in a file enter 'file'.\n"

<< "Enter what type you want to use:\n";

std::cin >> choice;

if (choice == "console")

{

writter = new ConsoleWriter();

isIncorrect = false;

}

else if (choice == "file")

{

writter = new FileWriter();

isIncorrect = false;

}

else // wrong input

std::cerr << "The word '" << choice << "' don't match any of method to

output the data.\n";

} while (isIncorrect);

return writter;

}

int main()

{

Reader\* reader = inputMethod();

std::string answer = findNumberInString(reader->inputString());

Writer\* writer = outputMethod();

writer->outputString(answer);

delete reader;

delete writer;

return 0;

}

**Код программы Java:**

**Код Main.java**

import java.util.Scanner;

public class Main {  
 public final static Scanner in = new Scanner(System.in);  
  
 static Reader inputMethod(){  
 Reader reader = null;  
 boolean isIncorrect = true;  
 String choice = "";  
 System.out.println("The program works with console input or files.");  
 do {  
 System.out.print("""  
 To use console enter 'console'.  
 To use a file enter 'file'.  
 Enter what type you want to use:  
 """);  
 choice = in.nextLine();  
 if (choice.equals("console")) {  
 reader = new ConsoleReader();  
 isIncorrect = false;  
 }  
 else if (choice.equals("file")) {  
 reader = new FileReader();  
 isIncorrect = false;  
 }  
 // wrong input  
 else {  
 System.err.println("The word " + choice + " don't match any of method

to input the data.");  
 }  
 } while(isIncorrect);  
 return reader;  
 }  
  
 static Writer outputMethod() {  
 Writer writer = null;  
 boolean isIncorrect = true;  
 String choice = "";  
 System.out.println("The program is ready to show answer.");  
 do {  
 System.out.print("""  
 To output in console enter 'console'.  
 To output in a file enter 'file'.  
 Enter what type you want to use:  
 """);  
 choice = in.nextLine();  
 if (choice.equals("console"))  
 {  
 writer = new ConsoleWriter();  
 isIncorrect = false;  
 }  
 else if (choice.equals("file"))  
 {  
 writer = new FileWriter();  
 isIncorrect = false;  
 }  
 else // wrong input  
 System.err.println("The word '" + choice + "' don't match any of method

to output the data.");  
 } while(isIncorrect);  
 return writer;  
 }  
  
 public static String findNumberInString(String s) {  
 boolean isNumber = false;  
 boolean wasNumber = false;  
 String number = "";  
 for (int i = 0; i < s.length(); i++)  
 {  
 if (!wasNumber && (s.charAt(i) == '+' || s.charAt(i) == '-')) {  
 isNumber = true;  
 number = "";  
 number += s.charAt(i);  
 }  
 else if (isNumber && Character.isDigit(s.charAt(i))) {  
 number += s.charAt(i);  
 wasNumber = true;  
 }  
 else if (isNumber && wasNumber) {  
 isNumber = false;  
 }  
 else if (!wasNumber) {  
 isNumber = false;  
 }  
 }  
 // if number == "-" or number == "+" we return empty number  
 return (wasNumber) ? number : "\0" ;  
 }  
  
 public static void main(String[] args)  
 {  
  
 Reader reader = inputMethod();  
 String answer= findNumberInString(reader.inputString());  
 Writer writer = outputMethod();  
 writer.outputString(answer);  
 in.close();  
 }  
}

**Код Reader.java**

public abstract class Reader {  
 public abstract String inputString();  
 public void emptyStringMessage() {  
 System.err.println("Your string Empty! Try again.");  
 }  
}

**Код ConsoleReader.java**

public class ConsoleReader extends Reader {  
  
 @Override  
 public String inputString() {  
 String inputtedString = "";  
 boolean isIncorrect = true;  
 System.out.println("Enter string:");  
 do {  
 inputtedString = Main.in.nextLine();  
 if (!inputtedString.isEmpty())  
 isIncorrect = false;  
 else  
 emptyStringMessage();  
 } while (isIncorrect);  
 return inputtedString;  
 }  
}

**Код FileReader.java**

import java.io.BufferedReader;  
import java.io.File;  
import java.io.IOException;  
  
public class FileReader extends Reader {  
 private String fileName;  
  
 public boolean isFileGood()  
 {  
 File inputfile = new File(fileName);  
 // if file doesn't exist  
 if (!inputfile.exists()) {  
 System.err.println("This file or the path to the file is specified

incorrectly or does not exist! Try again.");  
 }  
 // if file isn't .txt  
 else if (!fileName.endsWith(".txt")) {  
 System.err.println("This file or path to the file isn't .txt! Try again.");  
 }  
 else if (!inputfile.canRead()) {  
 System.err.println("The program can't read this file! Try again.");  
 }  
 else {  
 return true;  
 }  
 return false;  
 }  
 @Override  
 public String inputString() {  
 boolean isIncorrect = true;  
 String inputtedString = "\0";  
 do {  
 System.out.println("Enter the name of file in this directory or path to

this file including name of file:");  
 fileName = Main.in.nextLine();  
 if (isFileGood()) {  
 try (BufferedReader reader = new BufferedReader(new

java.io.FileReader(fileName))){  
 inputtedString = reader.readLine();  
 if (inputtedString == null){  
 emptyStringMessage();  
 }  
 else {  
 isIncorrect = false; // to exit this loop  
 }  
 }  
 catch (IOException ex) {  
 System.err.println("Oops! Something went wrong.");  
 }  
 }  
 }while(isIncorrect);  
 return inputtedString;  
 }  
}

**Код Writer.java**

abstract public class Writer {  
 abstract public void outputString(String s);  
}

**Код ConsoleWriter.java**

public class ConsoleWriter extends Writer {  
 @Override  
 public void outputString(String s) {  
 if (s.isEmpty()) {  
 System.out.println("Function didn't find number in string.");  
 }  
 else {  
 System.out.println("The number in the string is " + s);  
 }  
 }  
}

**Код FileWriter.java**

import java.io.BufferedWriter;  
import java.io.File;  
import java.io.IOException;  
  
public class FileWriter extends Writer {  
  
 private String fileName;  
 public boolean isFileGood()  
 {  
 File inputfile = new File(fileName);  
 // if file doesn't exist  
 if (!inputfile.exists()) {  
 System.err.println("This file or the path to the file is specified

incorrectly or does not exist! Try again.");  
 }  
 // if file isn't .txt  
 else if (!fileName.endsWith(".txt")) {  
 System.err.println("This file or path to the file isn't .txt! Try again.");  
 }  
 else if (!inputfile.canWrite()) {  
 System.err.println("The program can't read this file! Try again.");  
 }  
 else {  
 return true;  
 }  
 return false;  
 }  
  
 @Override  
 public void outputString(String s) {  
 boolean isIncorrect = true;  
 do {  
 System.out.println("Enter the name of file in this directory or path to

this file including name of file:");  
 fileName = Main.in.nextLine();  
 if (isFileGood()) {  
 try (BufferedWriter out = new BufferedWriter(new

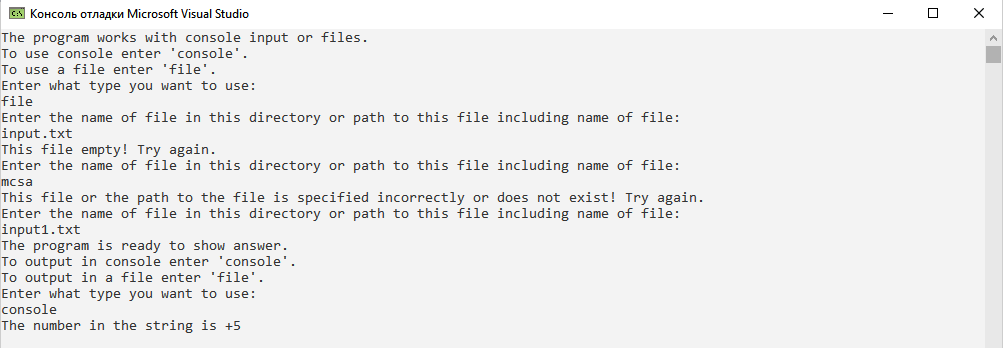
java.io.FileWriter(fileName))){  
 if (s.isEmpty()){  
 out.write("Function didn't find number in string.\n");  
 }  
 else{  
 out.write("The number in the string is" + s + "\n");  
 }  
 System.out.println("Answer has been wrote successfully.");  
 isIncorrect = false;  
 }  
 catch (IOException ex) {  
 System.err.println("Oops! Something went wrong.");  
 }  
 }  
 }while(isIncorrect);  
 }  
}

**Скриншоты:**

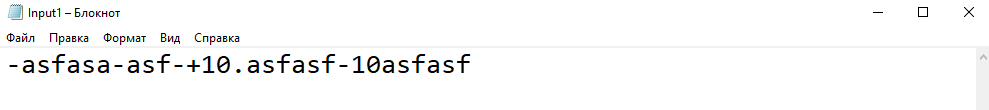
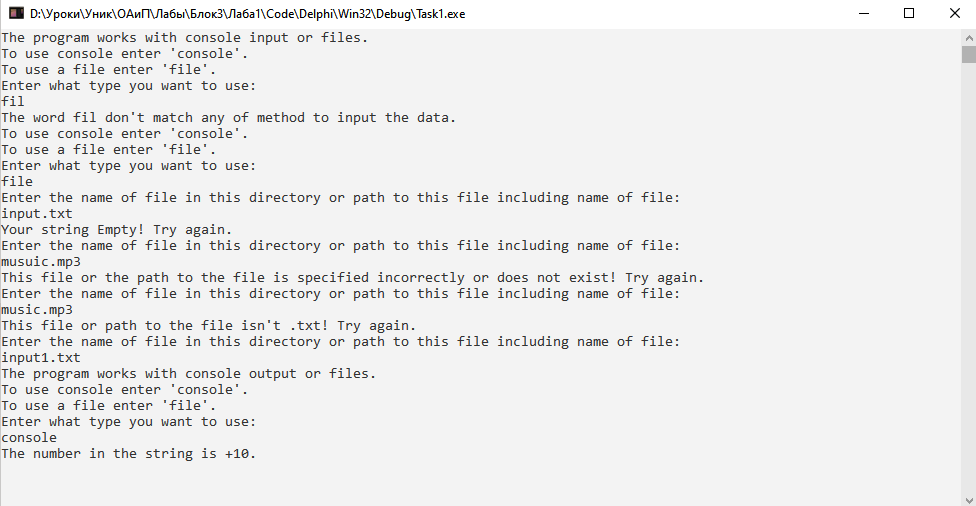
**C++:**

****

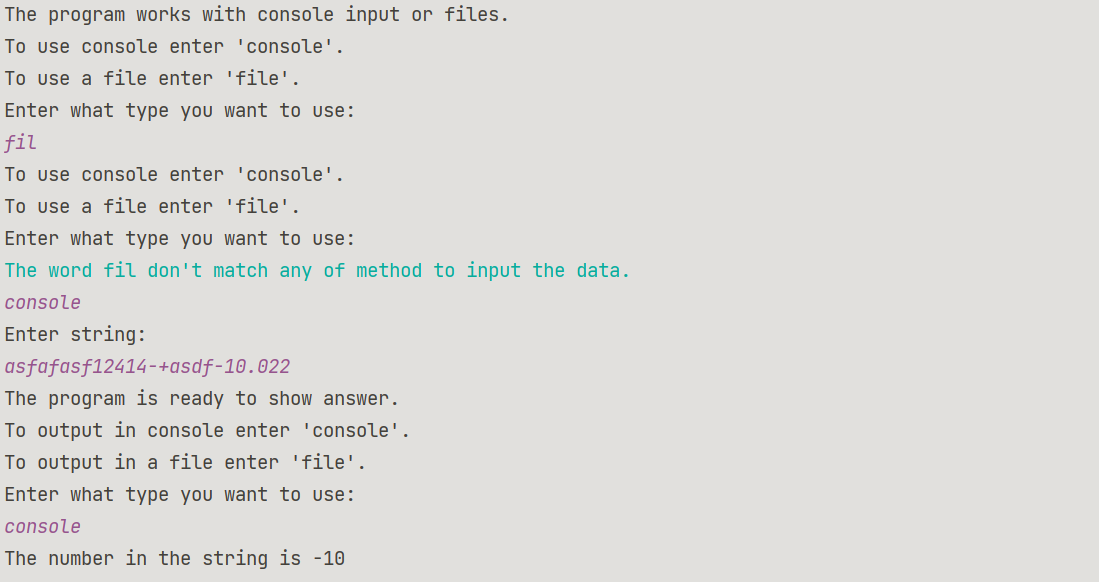
****

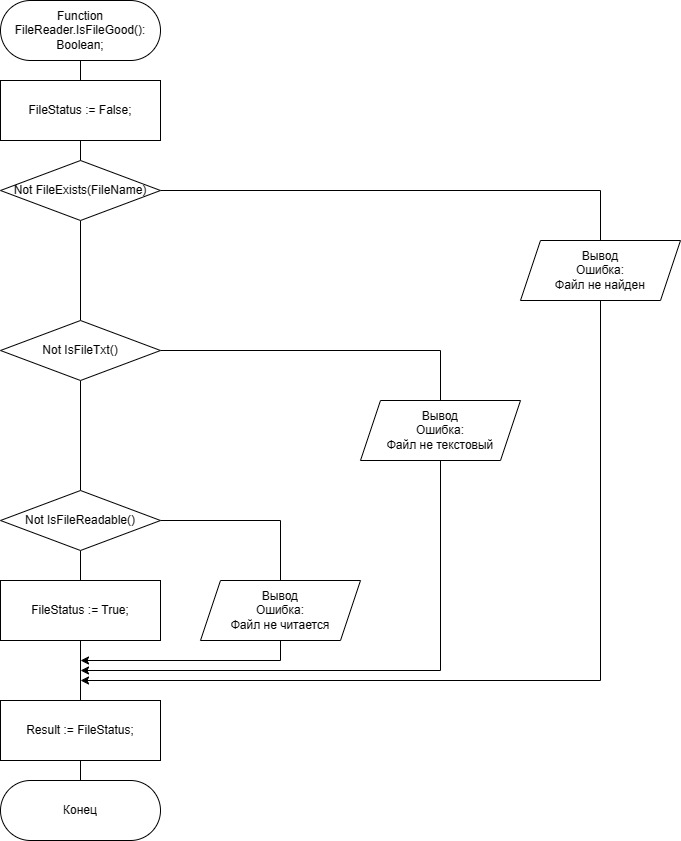


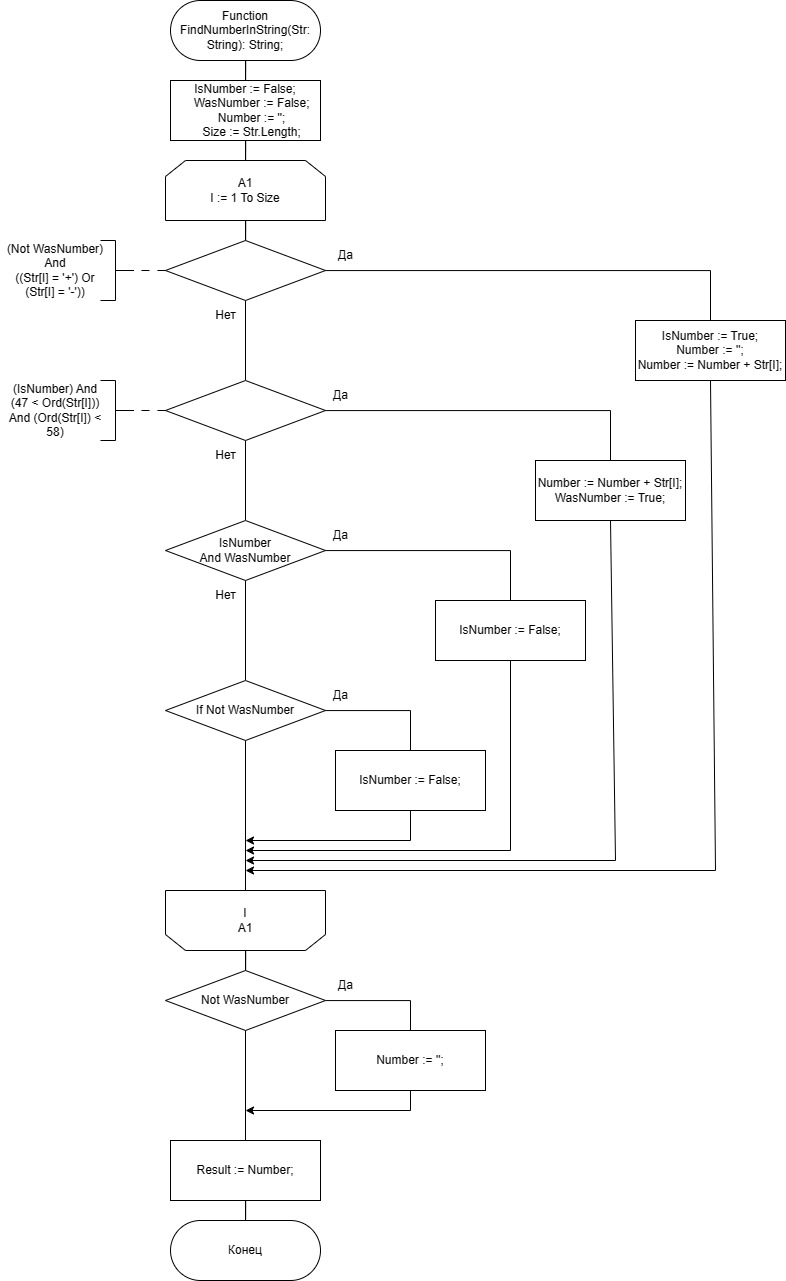
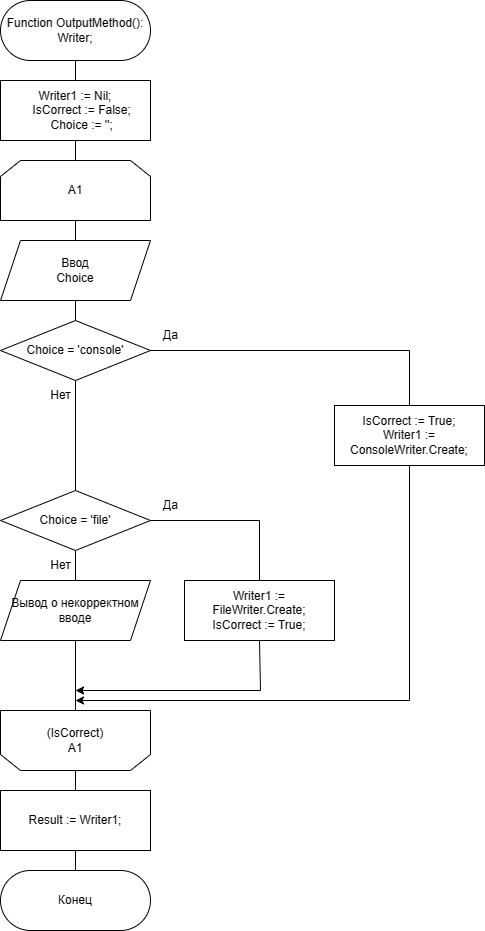
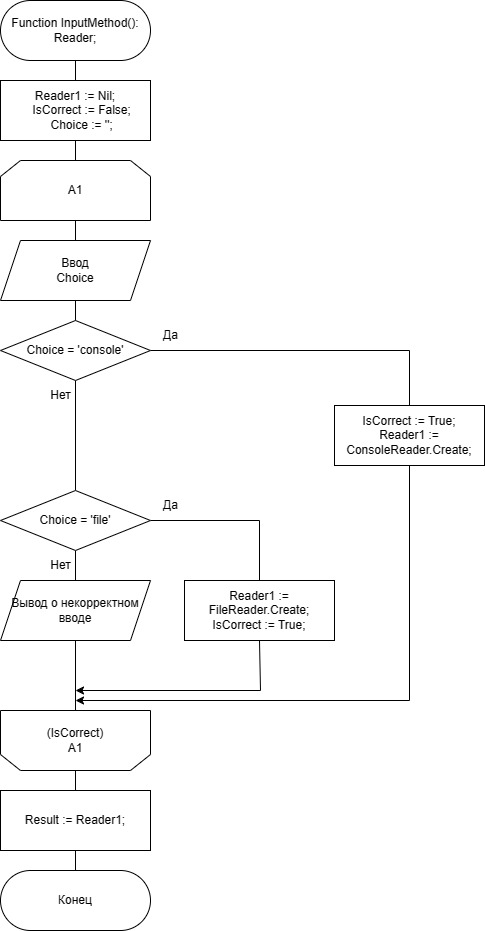
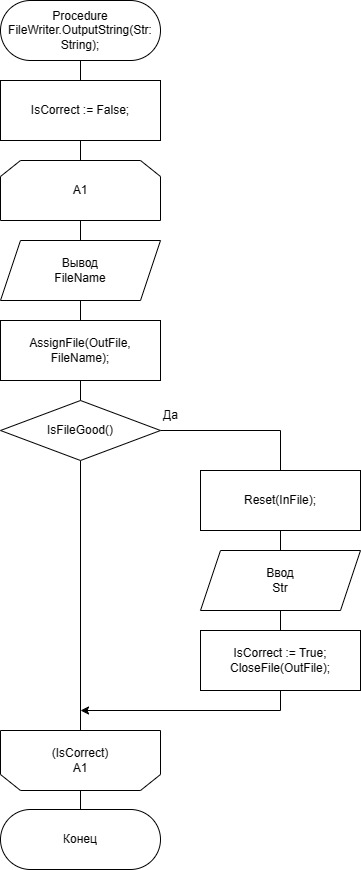
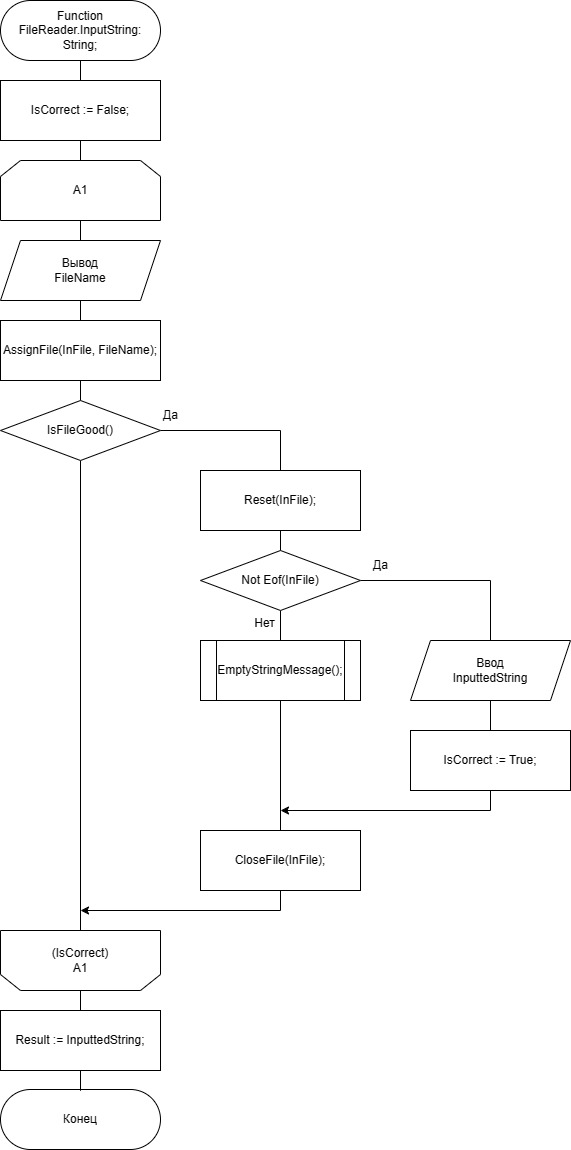
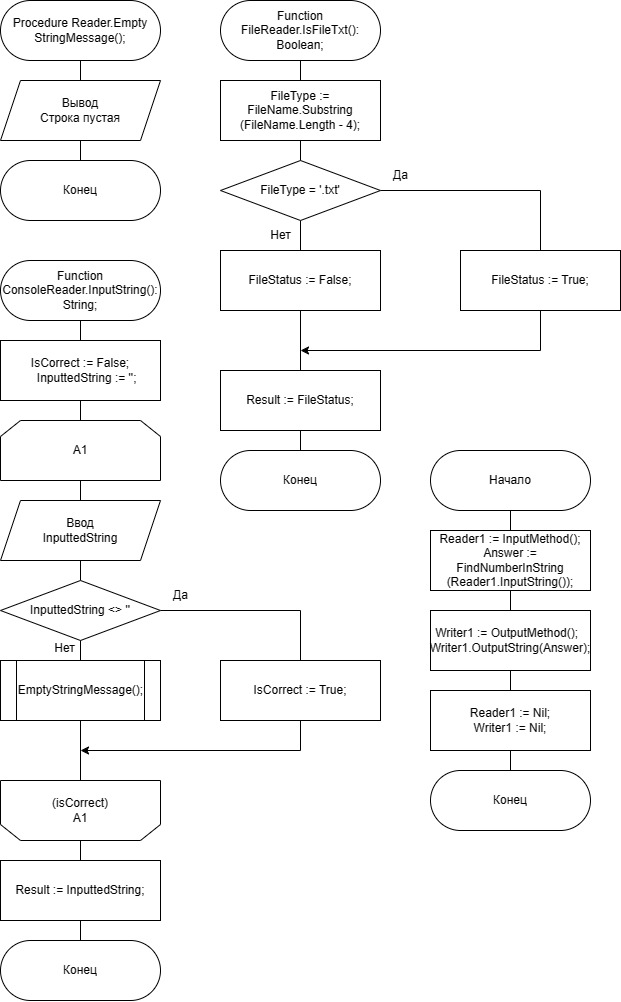
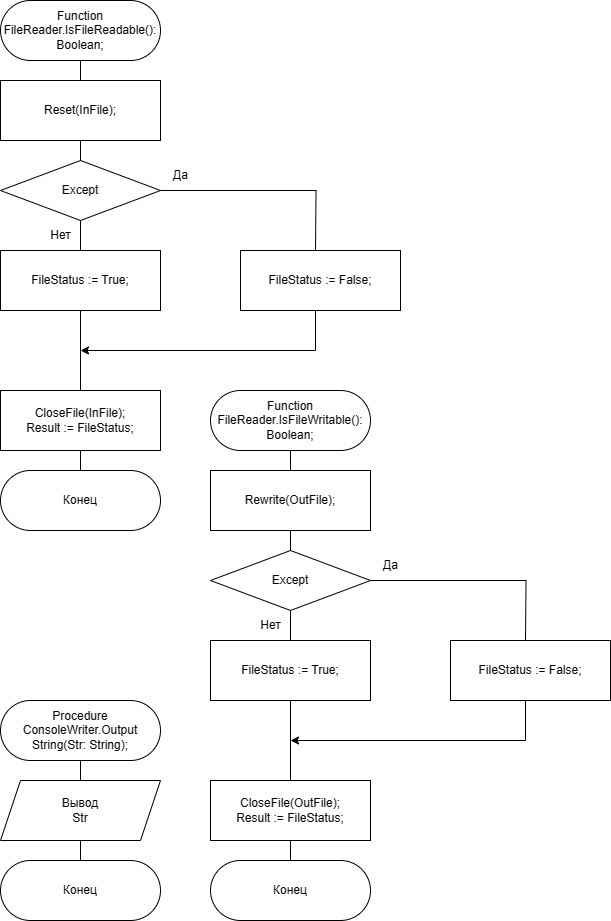
**Delphi:**



**Java:**



**Блок-схема: **



**Юнит Тесты (Unit Tests)**

**Код Unit Tests**

#include "pch.h"

#include "CppUnitTest.h"

#include "..\Exercise 1\Exercise 1.cpp" //attach our task

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

//Checking the function findNumberInString()

TEST\_CLASS(MainFunction)

{

public:

TEST\_METHOD(TestMethod1)

{

Assert::AreEqual(findNumberInString("asfasfa+10asf"), (std::string)"+10");

}

TEST\_METHOD(TestMethod2)

{

Assert::AreEqual(findNumberInString("asffa\_-125.245241"),

(std::string)"-125");

}

TEST\_METHOD(TestMethod3)

{

Assert::AreEqual(findNumberInString("hjkjhlk-+-+25asfasf+10"),

(std::string)"+25");

}

TEST\_METHOD(TestMethod4)

{

Assert::AreEqual(findNumberInString("-1fsafasfsaf"), (std::string)"-1");

}

TEST\_METHOD(TestMethod5)

{

Assert::AreEqual(findNumberInString("ыавыфаasf-.222222+++++5"),

(std::string)"+5");

}

TEST\_METHOD(TestMethod6)

{

Assert::AreEqual(findNumberInString("asfsf12fassff-.222-+-+fasfasf"),

(std::string)"\0");

}

};

//Checking the functions which check the correct of file

File\* testFile = nullptr;

//checking the function isFileTxt();

TEST\_CLASS(IsFileTxt)

{

public:

TEST\_METHOD(TestMethod1)

{

testFile = new File("music.mp3", (size\_t)0);

Assert::AreEqual(testFile->isFileText(), false);

delete testFile;

}

TEST\_METHOD(TestMethod2)

{

testFile = new File("input.txt", (size\_t)0);

Assert::AreEqual(testFile->isFileText(), true);

delete testFile;

}

TEST\_METHOD(TestMethod3)

{

testFile = new File("output.txt", (size\_t)1);

Assert::AreEqual(testFile->isFileText(), true);

delete testFile;

}

};

//checking the function isFileExist()

//to check this function you need some files on your device

TEST\_CLASS(IsFileExist)

{

public:

TEST\_METHOD(TestMethod1)

{

testFile = new

File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\music.mp3",

(size\_t)0);

Assert::AreEqual(testFile->isFileExist(), true);

delete testFile;

}

TEST\_METHOD(TestMethod2)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input124.txt",

(size\_t)0);

Assert::AreEqual(testFile->isFileExist(), false);

delete testFile;

}

TEST\_METHOD(TestMethod3)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\output.txt",

(size\_t)1);

Assert::AreEqual(testFile->isFileExist(), true);

delete testFile;

}

};

//checking the function isNotEmpty()

//to check this function you need some files on your device

TEST\_CLASS(IsFileNotEmpty)

{

public:

TEST\_METHOD(TestMethod1)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input.txt",

(size\_t)0);

Assert::AreEqual(testFile->isNotEmpty(), false);

delete testFile;

}

TEST\_METHOD(TestMethod2)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input1.txt",

(size\_t)0);

testFile->isGood();

Assert::AreEqual(testFile->isNotEmpty(), true);

delete testFile;

}

};

//checking the function isGood()

//this function including 5 functions

// 1 - isFileExist()

// 2 - isFileTxt()

// 3 - isNotEmpty() - Only works when file has fileCode = 0

// 4 - isFileReadable() - Only works when file has fileCode = 0

// 5 - isFileWritable() - Only works when file has fileCode = 1

//

TEST\_CLASS(IsFileGood)

{

public:

TEST\_METHOD(TestMethod1)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\music.mp3",

(size\_t)0);

Assert::AreEqual(testFile->isGood(), false);

delete testFile;

}

TEST\_METHOD(TestMethod2)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input124.txt",

(size\_t)0);

Assert::AreEqual(testFile->isGood(), false);

delete testFile;

}

TEST\_METHOD(TestMethod3)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input.txt",

(size\_t)0);

Assert::AreEqual(testFile->isGood(), false);

delete testFile;

}

TEST\_METHOD(TestMethod4)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input1.txt",

(size\_t)0);

Assert::AreEqual(testFile->isGood(), true);

delete testFile;

}

};

//checking the function getString()

TEST\_CLASS(getString)

{

public:

TEST\_METHOD(TestMethod1)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input1.txt",

(size\_t)0);

testFile->isGood();

Assert::AreEqual(testFile->getString(),

(std::string)("gdsgdsdsf-.222222+++++5"));

delete testFile;

}

TEST\_METHOD(TestMethod2)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input2.txt",

(size\_t)0);

testFile->isGood();

Assert::AreEqual(testFile->getString(),

(std::string)("hjkjhlk-+-+25asfasf+10"));

delete testFile;

}

};

//checking the function isFileWroking()

TEST\_CLASS(isWorking)

{

public:

TEST\_METHOD(TestMethod1)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise

1\\outputRead.txt", (size\_t)0);

testFile->isGood();

Assert::AreEqual(testFile->isFileWorking(), false);

delete testFile;

}

TEST\_METHOD(TestMethod2)

{

testFile = new File("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба1\\Code\\C++\\Exercise 1\\input1.txt",

(size\_t)0);

testFile->isGood();

Assert::AreEqual(testFile->isFileWorking(), true);

delete testFile;

}

};

}

**Скриншоты:**

