УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.2

по предмету «Основы алгоритмизации и программирования»

Вариант 18

Выполнил:

Егоров А.С.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Дана непустая последовательность символов, требуется построить и напечатать множество, элементами которого являются встречающиеся в последовательности знаки арифметических операций, скобки и чётные цифры.

**Код программы Delphi:**

Program Program2;

{$APPTYPE CONSOLE}

{$R \*.res}

Uses

System.SysUtils;

Const

COUNT\_SYMBOLS : Integer = 255;

KEY\_SYMBOLS: Set Of Char = ['[', ']', '{', '}', '(', ')', '+', '-', '\*',

'/', '%'];

Type

TSet = Set Of Char;

// 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 OutputSet(Set1: TSet); Virtual; Abstract;

End;

// class ConsoleWriter

Type

TConsoleWriter = Class(TWriter)

Public

Procedure OutputSet(Set1: TSet); Override;

End;

Procedure TConsoleWriter.OutputSet(Set1: TSet);

Var

I: Integer;

Begin

Writeln('Function found this symbols in the inputted string.');

For I := 0 To COUNT\_SYMBOLS Do

If Chr(I) In Set1 Then

Write(Chr(I), ' ');

Writeln;

End;

// class FileWriter

Type

TFileWriter = Class(TWriter)

Private

OutFile: TextFile;

FileName: String;

FileStatus: Boolean;

Public

Procedure OutputSet(Set1: TSet); 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.OutputSet(Set1: TSet);

Var

IsCorrect: Boolean;

I: Integer;

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,

'Function found this symbols in the inputted string.');

For I := 0 To COUNT\_SYMBOLS Do

If Chr(I) In Set1 Then

Write(OutFile, Chr(I), ' ');

Writeln(OutFile);

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 FindSymbols(Str: String): TSet;

Var

I, Size, CurCode: Integer;

Answer: TSet;

CurChar: Char;

Begin

Answer := [];

Size := Length(Str);

For I := 1 To Size Do

Begin

CurChar := Str[I];

CurCode := Ord(CurChar);

If (CurChar In KEY\_SYMBOLS) Then

Answer := Answer + [CurChar]

// 48 - code of '0'

// 57 - code of '9'

Else If (47 < CurCode) And (CurCode < 58) Then

Begin

Dec(CurCode, 48);

If Curcode Mod 2 = 0 Then

Answer := Answer + [CurChar];

End;

End;

Result := Answer;

End;

Procedure InputTask();

Var

I: Integer;

Begin

Writeln('This program finding the symbols in the string, which you', #39,

'll input.', #13#10, 'Symbols: ');

For I := 0 To COUNT\_SYMBOLS Do

If Chr(I) In KEY\_SYMBOLS Then

Write(Chr(I), ' ');

Writeln(#13#10, 'And digits that divided without remainder by 2.');

End;

Var

Reader1: TReader;

Writer1: TWriter;

Answer: TSet;

Begin

InputTask();

Reader1 := InputMethod();

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

Writer1 := OutputMethod();

Writer1.OutputSet(Answer);

// free memory

Reader1 := Nil;

Writer1 := Nil;

End.

**Код программы С++:**

#include <iostream>

#include <string>

#include <fstream>

#include <set>

const std::set<char> KEY\_SYMBOLS = { '[',']','{','}','(',')','+','-','\*','/','%' };

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 printSet(const std::set<char>&);

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();

status = true;

}

else

{

status = false;

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

does not exist! Try again.\n";

}

return status;

}

bool File::isNotEmpty()

{

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

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

{

status = true;

}

else

{

status = false;

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

}

fileStream->close();

return status;

}

bool File::isFileText()

{

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

if (type == ".txt")

status = true;

else

{

status = false;

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

}

return status;

}

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

bool File::isFileWorking()

{

fileStream->open(fileName);

if (fileStream->good())

{

status = true;

}

else

{

status = false;

switch (fileCode)

{

case((size\_t)0):

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

break;

case((size\_t)1):

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

break;

default:

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

again.\n";

break;

}

}

fileStream->close();

return status;

}

bool File::isGood()

{

if (this->isFileExist() && this->isFileText() && this->isFileWorking())

{

if (fileCode == 0 && this->isNotEmpty())

{ }

}

return status;

}

std::string File::getString()

{

fileStream->open(fileName);

std::string inputtedString = "";

\*fileStream >> inputtedString;

fileStream->close();

return inputtedString;

}

void File::printSet(const std::set<char>& set)

{

fileStream->open(fileName);

if (set.size() != (size\_t)0)

{

\*fileStream << "Function found this symbols in the string.\n";

for (auto& symbol : set)

{

\*fileStream << symbol << " ";

}

\*fileStream << std::endl;

}

else

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

fileStream->close();

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 outputSet(std::set<char>&) = 0;

protected:

};

class ConsoleWriter : public Writer

{

public:

void outputSet(std::set<char>&) override;

private:

};

void ConsoleWriter::outputSet(std::set<char>& set)

{

if (set.size() != (size\_t)0)

{

std::cout << "Function found this symbols in the string.\n";

for (auto& symbol : set)

{

std::cout << symbol << " ";

}

std::cout << std::endl;

}

else

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

}

class FileWriter : public Writer

{

public:

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

{ };

void outputSet(std::set<char>&) override;

private:

std::string fileName;

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

};

void FileWriter::outputSet(std::set<char>& set)

{

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->printSet(set);

delete out;

}

void inputTask()

{

std::cout << "This program finding the symbols in the string, which you'll

input.\n" << "Symbols: ";

for (auto& keySymbols : KEY\_SYMBOLS)

{

std::cout << keySymbols << " ";

}

std::cout << "\nAnd digits that divided without remainder by 2." << std::endl;

}

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;

}

std::set<char> findSymbols(std::string s)

{

std::set<char> answer = {};

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

{

if (KEY\_SYMBOLS.find(s[i]) != KEY\_SYMBOLS.end() || (std::isdigit(s[i]) &&

((int)(s[i] - '0') % 2 == 0)))

answer.insert(s[i]);

}

return answer;

}

int main()

{

// inputing info about Task

inputTask();

Reader\* reader = inputMethod();

std::set<char> answer = findSymbols(reader->inputString());

Writer\* writer = outputMethod();

writer->outputSet(answer);

delete reader;

delete writer;

return 0;

}

**Код программы Java:**

**Код Main.java**

package org.example;  
  
import java.util.HashSet;  
import java.util.Scanner;  
import java.util.Set;  
  
public class Main {  
 static Scanner in = new Scanner(System.in);  
  
 private final static HashSet<Character> KEY\_SYMBOLS =

new HashSet<>(Set.of('[',']','{','}','(',')','+','-','\*','/','%'));  
  
 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;  
 }  
  
 static public HashSet<Character> findSymbols(String str) {  
 HashSet<Character> answer = new HashSet<>();  
 char curChar = '\0';  
 for (int i = 0; i < str.length(); i++){  
 curChar = str.charAt(i);  
 if (KEY\_SYMBOLS.contains(curChar) || (Character.isDigit(curChar) &&

(int)(curChar) % 2 == 0) ) {  
 answer.add(curChar);  
 }  
 }  
 return answer;  
 }  
  
 static public void inputTask(){  
 System.out.println("This program finding the symbols in the string, which

you'll input.\n" + "Symbols: ");  
 for (char symbol: KEY\_SYMBOLS) {  
 System.out.print(symbol + " ");  
 }  
 System.out.println("\nAnd digits that divided without remainder by 2.");  
 }  
  
 static public void main(String[] args) {  
 inputTask();  
 Reader reader = inputMethod();  
 HashSet<Character> answer = findSymbols(reader.inputString());  
 Writer writer = outputMethod();  
 writer.outputSet(answer);  
 in.close();  
 }  
}

**Код Reader.java**

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

**Код ConsoleReader.java**

package org.example;  
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**

package org.example;  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.IOException;  
  
public class FileReader extends Reader {  
 private String fileName;  
  
 private boolean status;  
 FileReader()  
 {}  
 FileReader(String fileName){  
 this.fileName = fileName;  
 }  
 public boolean isFileGood() {  
 status = false;  
 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.");  
 }  
 // if file can't Read  
 else if (!inputfile.canRead()) {  
 System.err.println("The program can't read this file! Try again.");  
 }  
 else {  
 status = true;  
 }  
 return status;  
 }  
 @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**

package org.example;  
  
import java.util.HashSet;  
  
abstract public class Writer {  
 abstract public void outputSet(HashSet<Character> set);  
}

**Код ConsoleWriter.java**

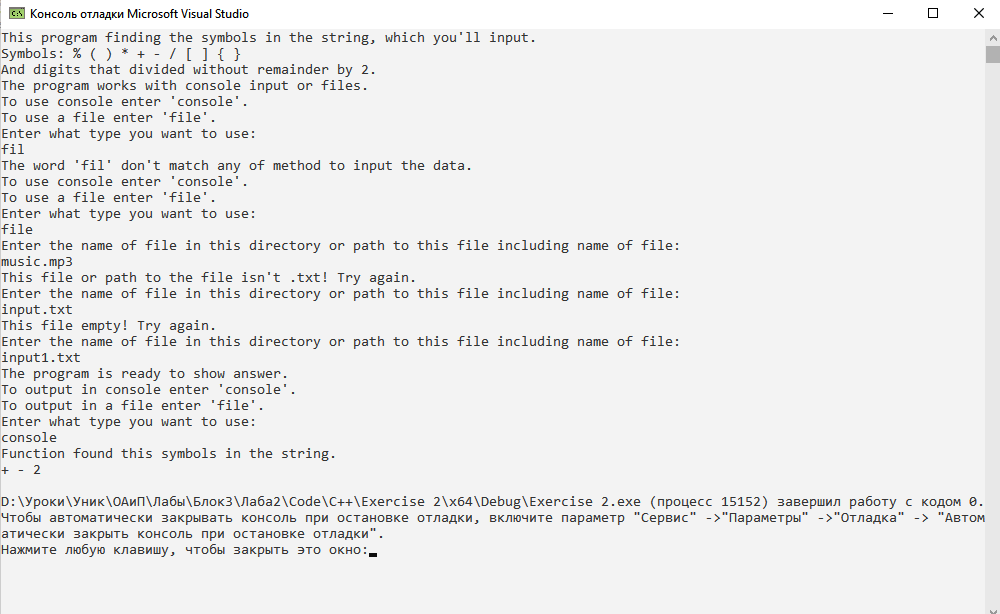
package org.example;  
  
import java.util.HashSet;  
  
public class ConsoleWriter extends Writer {  
 @Override  
 public void outputSet(HashSet<Character> set) {  
 if (!set.isEmpty()) {  
 System.out.println("Function found this symbols in the inputted string.");  
 for (char symbol: set) {  
 System.out.print(symbol + " ");  
 }  
 System.out.println();  
 }  
 else {  
 System.out.println("Function didn't find symbols in string.");  
 }  
 }  
}

**Код FileWriter.java**

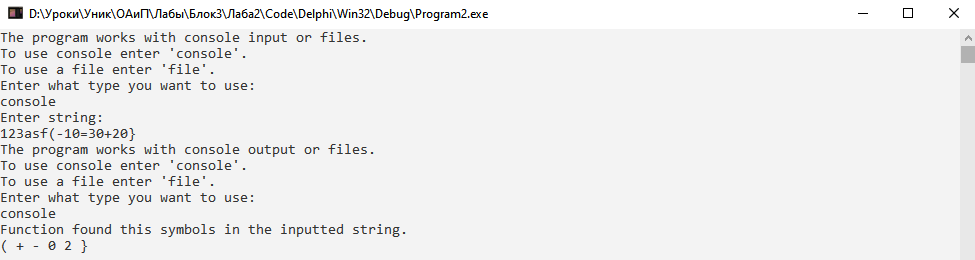
package org.example;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.IOException;  
import java.util.HashSet;  
  
public class FileWriter extends Writer {  
  
 private String fileName;  
  
 private boolean status;  
 FileWriter()  
 {}  
 FileWriter(String fileName){  
 this.fileName = fileName;  
 }  
 public boolean isFileGood()  
 {  
 status = false;  
 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 {  
 status = true;  
 }  
 return status;  
 }  
  
 @Override  
 public void outputSet(HashSet<Character> set) {  
 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 (!set.isEmpty()){  
 out.write("Function found this symbols in the inputted string.\n");  
 for (Character symbol: set) {  
 out.write(symbol + " ");  
 }  
 out.write('\n');  
 }  
 else{  
 out.write("Function didn't find symbols in string.\n");  
 }  
 System.out.println("Answer has been wrote successfully.");  
 isIncorrect = false;  
 }  
 catch (IOException ex) {  
 System.err.println("Oops! Something went wrong.");  
 }  
 }  
 }while(isIncorrect);  
 }  
}

**Скриншоты:**

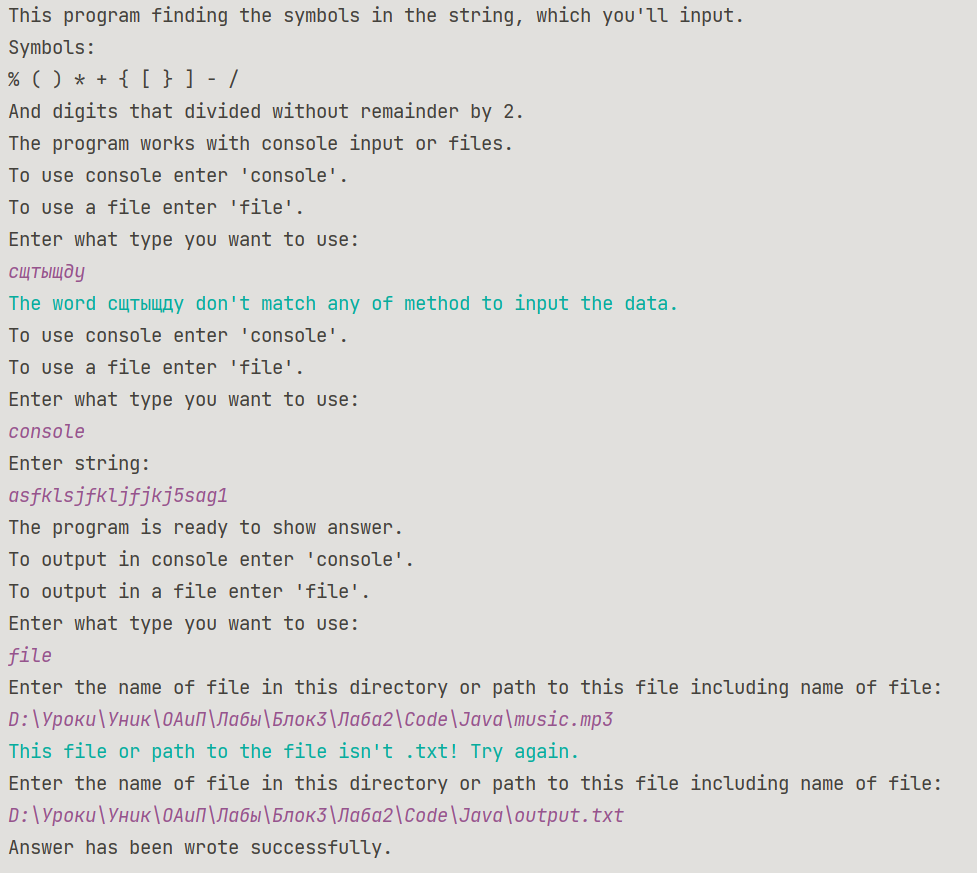
**C++:**



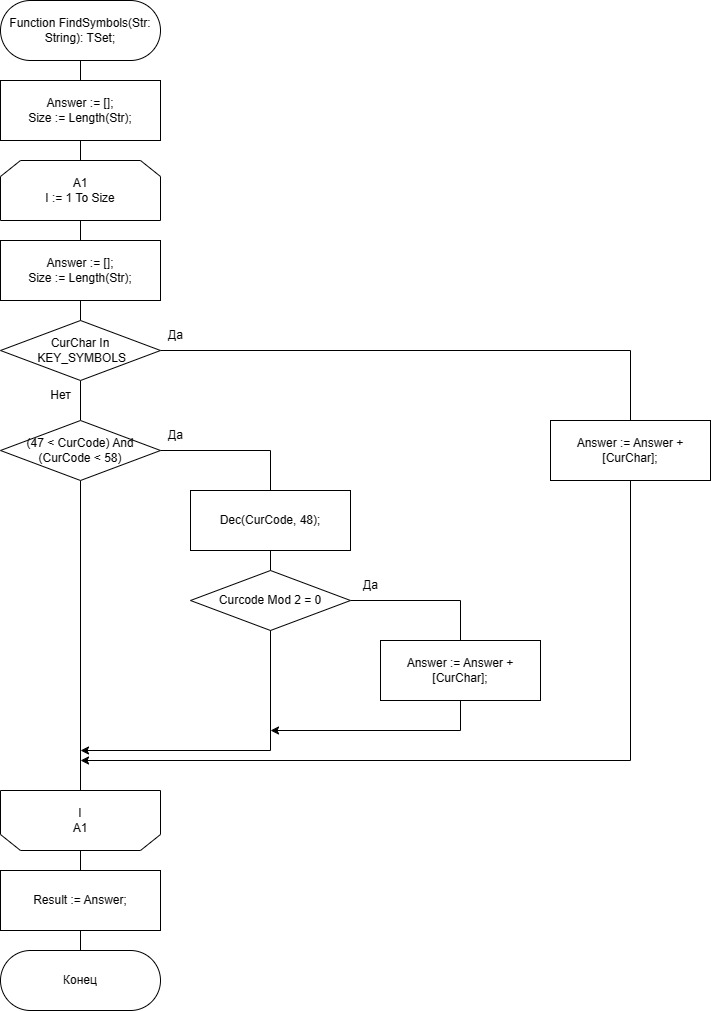
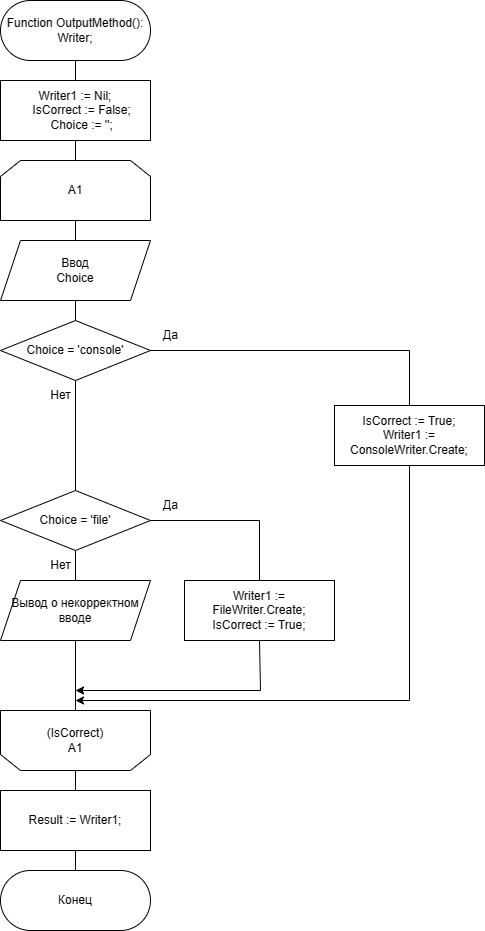
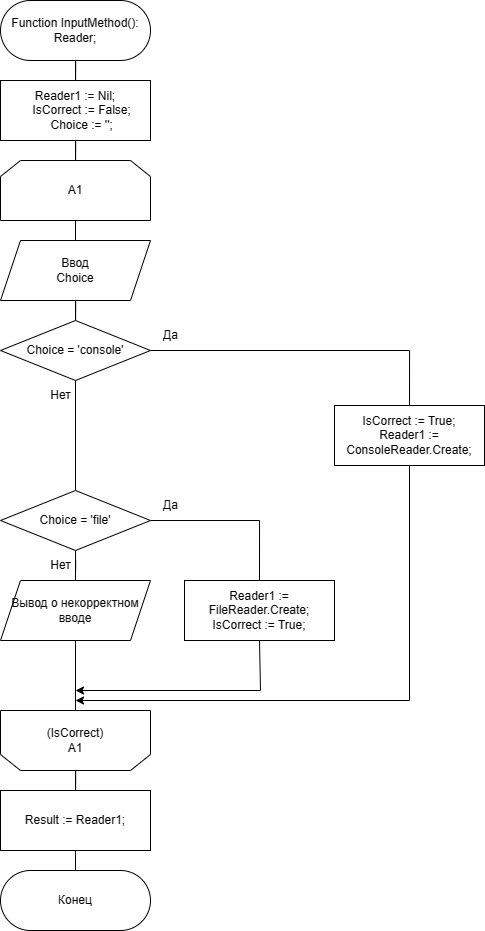
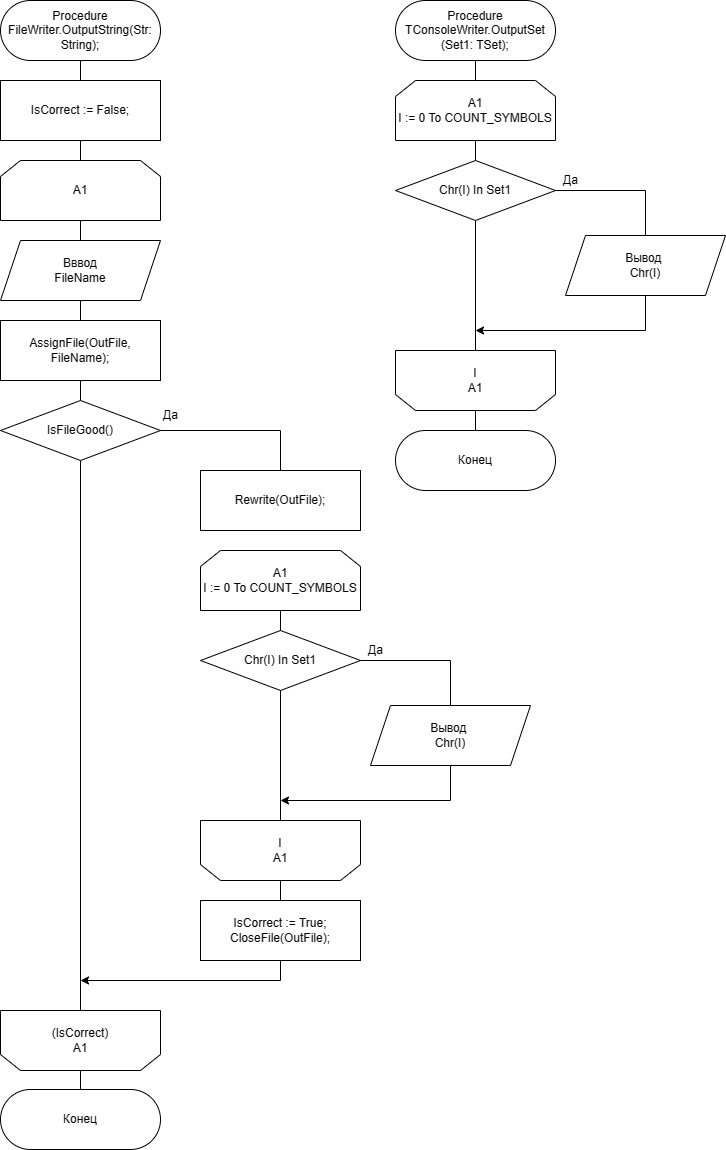
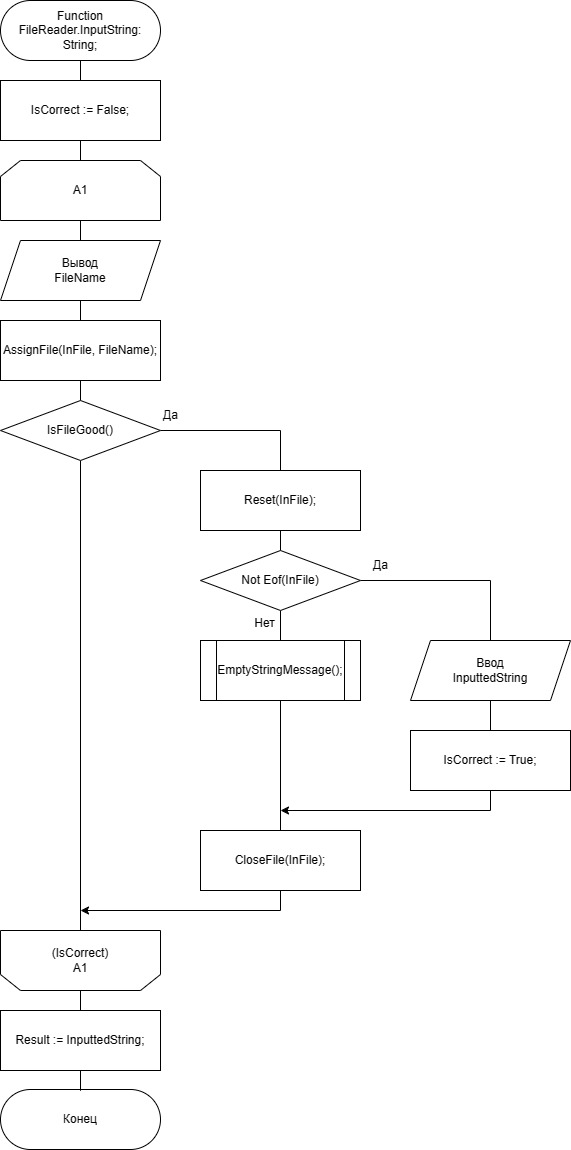
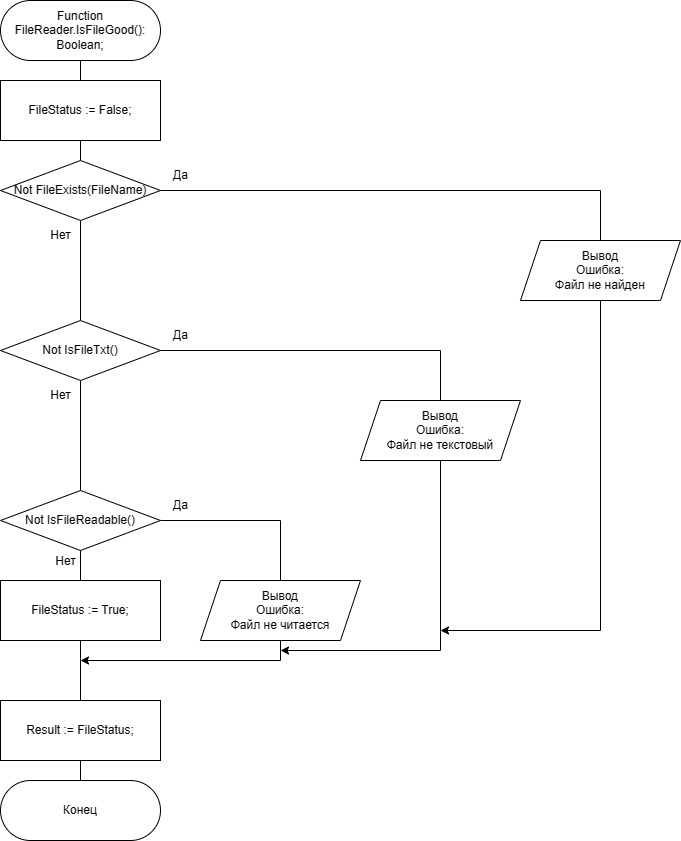
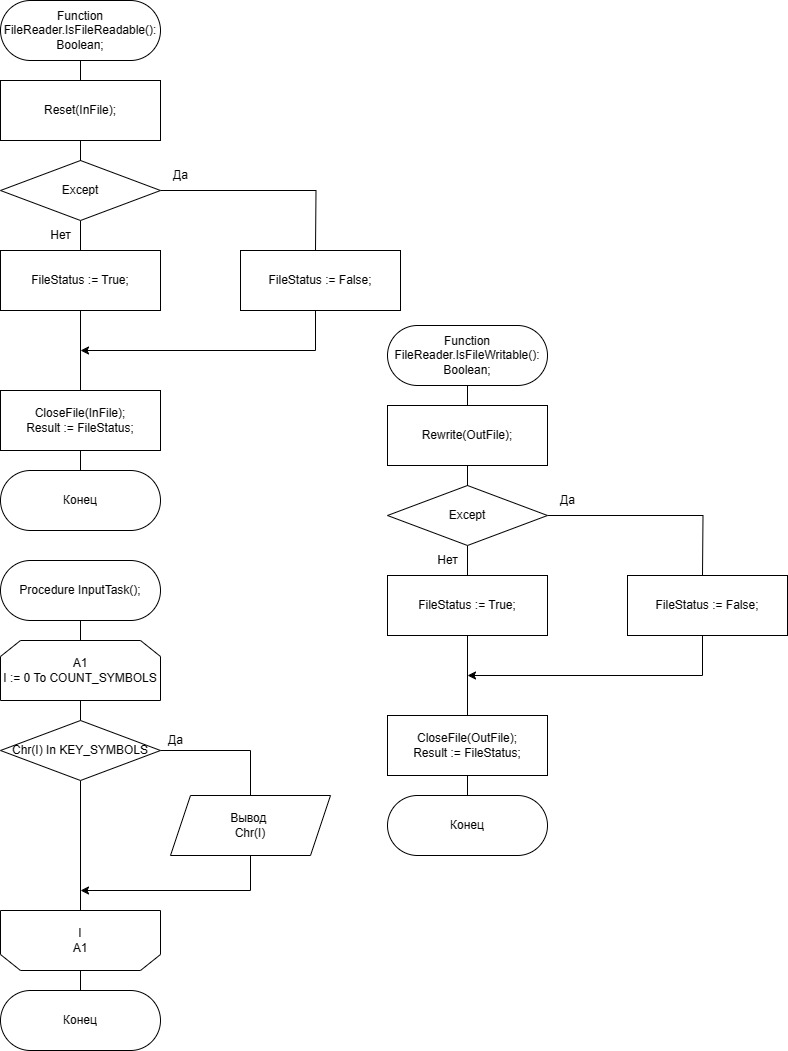
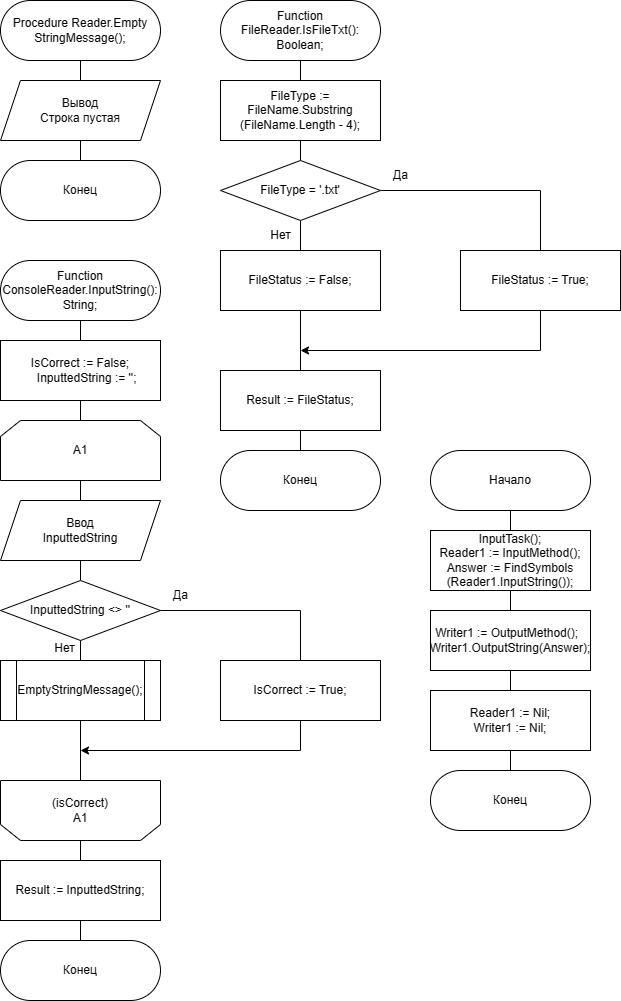
**Delphi:**



**Java:**



**Блок-схема:**



**Юнит Тесты (Unit Tests)**

**Код Unit Tests**

**FileWriterTest.java**

package org.example;  
  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class FileWriterTest {  
 private static FileWriter writer;  
  
 @Test  
 void isFileGood() {  
 writer =

new FileWriter("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба2\\input414.txt");  
 assertFalse(writer.isFileGood());  
 writer =

new FileWriter("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба2\\music.mp3");  
 assertFalse(writer.isFileGood());  
 writer =

new FileWriter("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба2\\output.txt");  
 assertTrue(writer.isFileGood());  
 }  
}

**FileReaderTest.java**

package org.example;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
  
  
class FileReaderTest {  
 private static FileReader reader;  
 @Test  
 void isFileGood() {  
 reader =

new FileReader("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба2\\input122.txt");  
 assertFalse(reader.isFileGood());  
 reader =

new FileReader("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба2\\music.mp3");  
 assertFalse(reader.isFileGood());  
 reader =

new FileReader("D:\\Уроки\\Уник\\ОАиП\\Лабы\\Блок3\\Лаба2\\input.txt");  
 assertTrue(reader.isFileGood());  
 }  
}

**FindSymbolsTest.java**

package org.example;  
  
import org.junit.jupiter.api.Test;  
  
  
import java.util.HashSet;  
import java.util.Set;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class FindSymbolsTest {  
  
 private static HashSet<Character> answer;  
  
 @Test  
 void findSymbolsTest() {  
 //test only digits  
 answer = new HashSet<>(Main.findSymbols("fsasf2525safsaf"));  
 assertEquals(answer, Set.of('2'));  
 answer.clear();  
 //test empty  
 answer = new HashSet<>(Main.findSymbols("asf9jf7io53,u."));  
 assertEquals(answer, Set.of());  
 answer.clear();  
 //test digits and symbols 1  
 answer = new HashSet<>(Main.findSymbols("safasf8sf2-+10"));  
 assertEquals(answer, Set.of('-','+','0','2','8'));  
 answer.clear();  
 //test digits and symbols 2  
 answer = new HashSet<>(Main.findSymbols("sa(at[asf'sf7%-10"));  
 assertEquals(answer, Set.of('%','-','(','[','0'));  
 answer.clear();  
 //test digits and symbols 3  
 answer = new HashSet<>(Main.findSymbols("{pjop]+11-34=-23/5=4.6)"));  
 assertEquals(answer, Set.of('{',']',')','-','+','/','2','4','6'));  
 answer.clear();  
 }  
}

**Скриншоты:**

