УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №3.2

по предмету «Основы алгоритмизации и программирования»

Вариант 12

Выполнил:

Галуха П. А.

Гр. 351005

Проверил:

Данилова Г. В.

Минск 2023

**Задание:**

Написать программу, которая формирует множество Y={X1X2X3} и выделить из него подмножество Y1, которое представляет все цифры, входящие в Y. Например, дано X1={‘s’, ‘v’, ‘e’, ‘t’, ‘a’,’ ‘2’}; X2={‘\*’, ’-‘, ‘\*’, ‘-‘}; X3={‘1’, ‘4’, ‘7’, ‘a’, ‘b’, ‘c’}.

**Код программы Delphi:**

Program Lab32;

Uses

System.SysUtils;

Type

TSet = Set Of AnsiChar;

ERRORS\_CODE = (CORRECT,

INCORRECT\_SET\_LENGTH,

INCORRECT\_SET\_El,

INCORRECT\_CHOICE,

IS\_NOT\_TXT,

IS\_NOT\_EXIST,

IS\_NOT\_READABLE,

IS\_NOT\_WRITEABLE,

INCORRECT\_SET\_AMOUNT);

Const

MIN\_S = 1;

MAX\_S = 85;

AMOUNT\_S = 3;

ERRORS: Array [1 .. 8] Of String = ('Длина множества не попадает в диапазон!',

'Элементы множества разделяются пробелом!',

'Некорректный выбор!',

'Расширение файла не .txt!',

'Проверьте корректность ввода пути к файлу!',

'Файл закрыт для чтения!',

'Файл закрыт для записи!',

'Неправильное число множеств в файле');

Procedure PrintTask();

Begin

WriteLn('Данная программа находит числа в множестве.'#13#10);

End;

Function IsCorrectSetLen(SSetEl: String) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

SetLen: Integer;

Begin

Error := CORRECT;

SetLen := (Length(SSetEl) + 1) Div 2;

If (SetLen < MIN\_S) Or (SetLen > MAX\_S) Then

Error := INCORRECT\_SET\_LENGTH;

IsCorrectSetLen := Error;

End;

Function IsCorrectSetEl(SSetEl: String) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

I: Integer;

Begin

Error := CORRECT;

I := 2;

While (Error = CORRECT) And (I <= Length(SSetEl)) Do

Begin

If SSetEl[I] <> ' ' Then

Error := INCORRECT\_SET\_El;

Inc(I, 2);

End;

IsCorrectSetEl := Error;

End;

Procedure FillSet(SSetEl: String; Var SetEl: TSet);

Var

I: Integer;

Begin

I := 1;

While I <= Length(SSetEl) Do

Begin

Include(SetEl, AnsiChar(SSetEl[I]));

Inc(I, 2);

End;

End;

Function ChooseOption(Amount: Integer) : Integer;

Var

Error: ERRORS\_CODE;

SOption: String;

IOption: Integer;

Begin

IOption := 1;

Repeat

Error := CORRECT;

ReadLn(SOption);

Try

IOption := StrToInt(SOption)

Except

Error := INCORRECT\_CHOICE;

End;

If (Error = CORRECT) And ((IOption < 1) Or (IOption > Amount)) Then

Error := INCORRECT\_CHOICE;

If Error <> CORRECT Then

WriteLn(ERRORS[Ord(Error)], #13#10'Повторите попытку: ');

Until Error = CORRECT;

ChooseOption := IOption;

End;

Function GetPartStr(Str: String; PosStart, PosEnd: Integer) : String;

Var

PartStr: String;

I: Integer;

Begin

partStr := '';

For I := PosStart To PosEnd Do

PartStr := PartStr + Str[I];

GetPartStr := PartStr;

End;

Function IsFileTXT(PathToFile: String) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

If (Length(PathToFile) < 5) Or (GetPartStr(PathToFile, Length(PathToFile) - 3, Length(PathToFile)) <> '.txt') Then

Error := IS\_NOT\_TXT;

IsFileTXT := Error;

End;

Function IsExist(PathToFile: String) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

If Not FileExists(PathToFile) Then

Error := IS\_NOT\_EXIST;

IsExist := Error;

End;

Function IsReadable(Var F: TextFile) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Reset(F);

CloseFile(F);

Except

Error := IS\_NOT\_READABLE;

End;

IsReadable := Error;

End;

Function IsWriteable(Var F: TextFile) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := CORRECT;

Try

Append(F);

CloseFile(F);

Except

Error := Is\_NOT\_WRITEABLE;

End;

IsWriteable := Error;

End;

Function IsCorrectSetAmount(Var F: TextFile) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Count, StopAmount: Integer;

Begin

Error := CORRECT;

Count := 0;

StopAmount := AMOUNT\_S + 1;

Reset(F);

While (Not EOF(F)) And (Count <> StopAmount) Do

Begin

ReadLn(F);

Inc(Count);

End;

CloseFile(F);

If Count <> AMOUNT\_S Then

Error := INCORRECT\_SET\_AMOUNT;

IsCorrectSetAmount := Error;

End;

Function IsCorrectFileSet(Var F: TextFile) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

SSetEl: String;

Begin

Error := IsCorrectSetAmount(F);

Reset(F);

While (Error = CORRECT) And (Not EOF(F)) Do

Begin

ReadLn(F, SSetEl);

Error := IsCorrectSetLen(SSetEl);

If Error = CORRECT Then

Error := IsCorrectSetEl(SSetEl);

End;

CloseFile(F);

IsCorrectFileSet := Error;

End;

Procedure GetFileNormalReading(Var F: TextFile);

Var

Error: ERRORS\_CODE;

PathToFile: String;

Begin

Write('Введите путь к файлу с расширением .txt с тремя множествами, с

длинами[', MIN\_S, '; ', MAX\_S, ']: '#13#10);

Repeat

ReadLn(PathToFile);

Error := IsFileTXT(PathToFile);

If Error = CORRECT Then

Error := IsExist(PathToFile);

If Error = CORRECT Then

AssignFile(F, PathToFile);

If Error = CORRECT Then

Error := IsReadable(F);

If Error = CORRECT Then

Error := IsCorrectFileSet(F);

If Error <> CORRECT Then

WriteLn(ERRORS[Ord(Error)], #13#10'Повторите попытку: ');

Until Error = CORRECT;

End;

Procedure GetFileNormalWriting(Var F: TextFile);

Var

Error: ERRORS\_CODE;

PathToFile: String;

Begin

Write('Введите путь к файлу с расширением .txt для получения результата: '#13#10);

Repeat

ReadLn(PathToFile);

Error := IsFileTXT(PathToFile);

If Error = CORRECT Then

Error := IsExist(PathToFile);

If Error = CORRECT Then

AssignFile(F, PathToFile);

If Error = CORRECT Then

Error := IsWriteable(F);

If Error <> CORRECT Then

WriteLn(ERRORS[Ord(Error)], #13#10'Повторите попытку: ');

Until Error = CORRECT;

End;

Procedure ReadFileSet(Var F: TextFile; Var SetEl: TSet);

Var

SSetEl: String;

Begin

ReadLn(F, SSetEl);

FillSet(SSetEl, SetEl);

End;

Function IsCorrectConsoleSet(SSetEl: String) : ERRORS\_CODE;

Var

Error: ERRORS\_CODE;

Begin

Error := IsCorrectSetLen(SSetEl);

If Error = CORRECT Then

Error := IsCorrectSetEl(SSetEl);

IsCorrectConsoleSet := Error;

End;

Procedure ReadConsoleSet(Num: Integer; Var SetEl: TSet);

Var

Error: ERRORS\_CODE;

SSetEl: String;

Begin

Write('Введите множество Х', Num, ' через пробелы: ');

Repeat

ReadLn(SSetEl);

Error := IsCorrectConsoleSet(SSetEl);

If Error <> CORRECT Then

Write(ERRORS[Ord(Error)], #13#10'Повторите попытку: ');

Until Error = CORRECT;

FillSet(SSetEl, SetEl);

End;

Procedure ReadSets(Var X1: TSet; Var X2: TSet; Var X3: TSet);

Var

RF: TextFile;

Option: Integer;

Begin

WriteLn('Вы хотите: ');

WriteLn('Вводить множества через файл - 1');

WriteLn('Вводить множества через консоль - 2');

Option := ChooseOption(2);

If Option = 1 Then

Begin

GetFileNormalReading(RF);

Reset(RF);

ReadFileSet(RF, X1);

ReadFileSet(RF, X2);

ReadFileSet(RF, X3);

CloseFile(RF);

End

Else

Begin

ReadConsoleSet(1, X1);

ReadConsoleSet(2, X2);

ReadConsoleSet(3, X3);

End;

End;

Procedure UniteSets(Var Y: TSet; X1: TSet; X2: TSet; X3: TSet);

Begin

Y := X1 + X2 + X3;

End;

Procedure FindNums(Var Y1: TSet; Y: TSet);

Var

Ch: AnsiChar;

Begin

For Ch in Y Do

If (Ch >= '0') And (Ch <= '9') Then

Include(Y1, Ch);

End;

Procedure PrintConsoleResult(Y1: TSet);

Var

Ch: AnsiChar;

Begin

Write(#13#10'Цифры в множестве: ');

If Y1 = [] Then

Write('цифр в множестве нет')

Else

For Ch in Y1 Do

Write('''', Ch, '''; ');

End;

Procedure PrintFileResult(Var F: TextFile; Y1: TSet);

Var

Ch: AnsiChar;

Begin

Append(F);

Write(F, #13#10'Цифры в множестве: ');

If Y1 = [] Then

Write(F, 'цифр в множестве нет')

Else

For Ch in Y1 Do

Write(F, '''', Ch, '''; ');

CloseFile(F);

End;

Procedure PrintResult(Y1: TSet);

Var

WF: TextFile;

Option: Integer;

Begin

WriteLn('Вы хотите: ');

WriteLn('Выводить множество через файл - 1');

WriteLn('Выводить множество через консоль - 2');

Option := ChooseOption(2);

If Option = 1 Then

Begin

GetFileNormalWriting(WF);

PrintFileResult(WF, Y1);

End

Else

PrintConsoleResult(Y1);

End;

Var

X1, X2, X3, Y, Y1: TSet;

Begin

PrintTask();

ReadSets(X1, X2, X3);

UniteSets(Y, X1, X2, X3);

FindNums(Y1, Y);

PrintResult(Y1);

ReadLn;

End.

**Код программы С++:**

#include <iostream>

#include <set>

#include <string>

#include <fstream>

enum ErrorsCode

{

CORRECT,

INCORRECT\_SET\_LENGTH,

INCORRECT\_SET\_El,

INCORRECT\_CHOICE,

IS\_NOT\_TXT,

IS\_NOT\_EXIST,

IS\_NOT\_READABLE,

IS\_NOT\_WRITEABLE,

INCORRECT\_SET\_AMOUNT,

};

const int

MIN\_S = 1,

MAX\_S = 85,

AMOUNT\_S = 3;

const std::string

ERRORS[] = { "",

"Длина множества не попадает в диапазон!",

"Элементы множества разделяются пробелом!",

"Некорректный выбор!",

"Расширение файла не .txt!",

"Проверьте корректность ввода пути к файлу!",

"Файл закрыт для чтения!",

"Файл закрыт для записи!",

"Неправильное число множеств в файле" };

void printTask()

{

std::cout << "Данная программа находит числа в множестве.\n\n";

}

ErrorsCode isCorrectSetLen(std::string sSetEl)

{

ErrorsCode error;

int setLen;

error = CORRECT;

setLen = (sSetEl.length() + 1) / 2;

if (setLen < MIN\_S || setLen > MAX\_S)

error = INCORRECT\_SET\_LENGTH;

return error;

}

ErrorsCode isCorrectSetEl(std::string sSetEl)

{

ErrorsCode error;

int i;

error = CORRECT;

i = 1;

while (error == CORRECT && i < sSetEl.length())

{

if (sSetEl[i] != ' ')

error = INCORRECT\_SET\_El;

i += 2;

}

return error;

}

void fillSet(std::string sSetEl, std::set<char>& setEl)

{

int i;

for (i = 0; i < sSetEl.length(); i += 2)

setEl.insert(sSetEl[i]);

}

int chooseOption(int amount)

{

ErrorsCode error;

int option;

option = 0;

do {

error = CORRECT;

std::cin >> option;

if (std::cin.fail())

{

error = INCORRECT\_CHOICE;

std::cin.clear();

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

}

if (error == CORRECT && std::cin.get() != '\n')

{

error = INCORRECT\_CHOICE;

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

}

if (error == CORRECT && (option < 1 || option > amount))

error = INCORRECT\_CHOICE;

if (error != CORRECT)

std::cout << ERRORS[error] << "\nПовторите попытку: \n";

} while (error != CORRECT);

return option;

}

std::string getPartStr(std::string str, int posStart, int posEnd)

{

std::string partStr;

int i;

partStr = "";

for (i = posStart; i <= posEnd; i++)

partStr = partStr + str[i];

return partStr;

}

ErrorsCode isFileTXT(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

if (pathToFile.length() < 5 ||

getPartStr(pathToFile, pathToFile.length() - 4, pathToFile.length() - 1) != ".txt")

error = IS\_NOT\_TXT;

return error;

}

ErrorsCode isExist(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

std::ifstream file(pathToFile);

if (!file.good())

error = IS\_NOT\_EXIST;

file.close();

return error;

}

ErrorsCode isReadable(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

std::ifstream file(pathToFile);

if (!file.is\_open())

error = IS\_NOT\_READABLE;

file.close();

return error;

}

ErrorsCode isWriteable(std::string pathToFile)

{

ErrorsCode error;

error = CORRECT;

std::ofstream file(pathToFile, std::ios::app);

if (!file.is\_open())

error = IS\_NOT\_WRITEABLE;

file.close();

return error;

}

ErrorsCode isCorrectSetAmount(std::string pathToFile)

{

ErrorsCode error;

int count, stopAmount;

error = CORRECT;

count = 0;

stopAmount = AMOUNT\_S + 1;

std::ifstream file(pathToFile);

while (!file.eof() && count != stopAmount)

{

file.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

count++;

}

file.close();

if (count != AMOUNT\_S)

error = INCORRECT\_SET\_AMOUNT;

return error;

}

ErrorsCode isCorrectFileSet(std::string pathToFile)

{

ErrorsCode error;

std::string sSetEl;

int i;

error = isCorrectSetAmount(pathToFile);

std::ifstream file(pathToFile);

while (error == CORRECT && !file.eof())

{

std::getline(file, sSetEl);

error = isCorrectSetLen(sSetEl);

if (error == CORRECT)

error = isCorrectSetEl(sSetEl);

}

file.close();

return error;

}

void getFileNormalReading(std::string& pathToFile)

{

ErrorsCode error;

std::cout << "Введите путь к файлу с расширением .txt с тремя множествами, с

длинами[" << MIN\_S << "; " << MAX\_S << "]: \n";

do

{

std::getline(std::cin, pathToFile);

error = isFileTXT(pathToFile);

if (error == CORRECT)

error = isExist(pathToFile);

if (error == CORRECT)

error = isReadable(pathToFile);

if (error == CORRECT)

error = isCorrectFileSet(pathToFile);

if (error != CORRECT)

std::cout << ERRORS[error] << "\nПовторите попытку: ";

} while (error != CORRECT);

}

void getFileNormalWriting(std::string& pathToFile)

{

ErrorsCode error;

std::cout << "Введите путь к файлу с расширением .txt для получения результата: \n";

do

{

std::getline(std::cin, pathToFile);

error = isFileTXT(pathToFile);

if (error == CORRECT)

error = isExist(pathToFile);

if (error == CORRECT)

error = isWriteable(pathToFile);

if (error != CORRECT)

std::cout << ERRORS[error] << "\nПовторите попытку: ";

} while (error != CORRECT);

}

void readFileSet(std::ifstream& file, std::set<char>& setEl)

{

std::string sSetEl;

std::getline(file, sSetEl);

fillSet(sSetEl, setEl);

}

ErrorsCode isCorrectConsoleSet(std::string sSetEl)

{

ErrorsCode error;

error = isCorrectSetLen(sSetEl);

if (error == CORRECT)

error = isCorrectSetEl(sSetEl);

return error;

}

void readConsoleSet(int num, std::set<char>& setEl)

{

ErrorsCode error;

std::string sSetEl;

std::cout << "Введите множество Х" << num << " через пробелы : ";

do

{

std::getline(std::cin, sSetEl);

error = isCorrectConsoleSet(sSetEl);

if (error != CORRECT)

std::cout << ERRORS[error] << "\nПовторите попытку: ";

} while (error != CORRECT);

fillSet(sSetEl, setEl);

}

void readSets(std::set<char>& x1, std::set<char>& x2, std::set<char>& x3)

{

std::string pathToFile;

int option;

std::cout << "Вы хотите: \n";

std::cout << "Вводить множества через файл - 1\n";

std::cout << "Вводить множества через консоль - 2\n";

option = chooseOption(2);

if (option == 1)

{

getFileNormalReading(pathToFile);

std::ifstream file(pathToFile);

readFileSet(file, x1);

readFileSet(file, x2);

readFileSet(file, x3);

file.close();

}

else

{

readConsoleSet(1, x1);

readConsoleSet(2, x2);

readConsoleSet(3, x3);

}

}

void uniteSets(std::set<char>& y, std::set<char> x1, std::set<char> x2, std::set<char> x3)

{

for (const auto& element : x1) {

y.insert(element);

}

for (const auto& element : x2) {

y.insert(element);

}

for (const auto& element : x3) {

y.insert(element);

}

}

void findNums(std::set<char>& y1, std::set<char> y)

{

for (const auto& element : y)

if (element >= '0' && element <= '9')

y1.insert(element);

}

void printConsoleResult(std::set<char> y1)

{

std::cout << "\nЦифры в множестве: ";

if (y1.empty())

std::cout << "элементов, удовлетворяющих условию, нет!";

else

for (const auto& element : y1)

std::cout << "'" << element << "'; ";

}

void printFileResult(std::string pathToFile, std::set<char> y1)

{

std::ofstream file(pathToFile, std::ios::app);

file << "\nЦифры в множестве: ";

if (y1.empty())

file << "элементов, удовлетворяющих условию, нет!";

else

for (const auto& element : y1)

file << "'" << element << "'; ";

file.close();

}

void printResult(std::set<char> y1)

{

std::string pathToFile;

int option;

std::cout << "Вы хотите: \n";

std::cout << "Выводить множесто через файл - 1\n";

std::cout << "Выводить множесто через консоль - 2\n";

option = chooseOption(2);

if (option == 1)

{

getFileNormalWriting(pathToFile);

printFileResult(pathToFile, y1);

}

else

printConsoleResult(y1);

}

int main()

{

setlocale(LC\_ALL, "RU");

std::set<char> x1;

std::set<char> x2;

std::set<char> x3;

std::set<char> y;

std::set<char> y1;

printTask();

readSets(x1, x2, x3);

uniteSets(y, x1, x2, x3);

findNums(y1, y);

printResult(y1);

return 0;

}

**Код программы Java:**

import java.util.Scanner;

import java.io.File;

import java.io.FileWriter;

import java.util.Set;

import java.util.HashSet;

public class Main {

public enum ErrorsCode {

CORRECT,

INCORRECT\_SET\_LENGTH,

INCORRECT\_SET\_El,

INCORRECT\_CHOICE,

IS\_NOT\_TXT,

IS\_NOT\_EXIST,

IS\_NOT\_READABLE,

IS\_NOT\_WRITEABLE,

INCORRECT\_SET\_AMOUNT,

}

public static final int

MIN\_S = 1,

MAX\_S = 85,

AMOUNT\_S = 3;

public static final String[]

ERRORS = { "",

"Длина множества не попадает в диапазон!",

"Элементы множества разделяются пробелом!",

"Некорректный выбор!",

"Расширение файла не .txt!",

"Проверьте корректность ввода пути к файлу!",

"Файл закрыт для чтения!",

"Файл закрыт для записи!",

"Неправильное число множеств в файле" };

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

public static void printTask() {

System.out.print("Данная программа находит числа в множестве.\n\n");

}

public static ErrorsCode isCorrectSetLen(String sSetEl) {

ErrorsCode error;

int setLen;

error = ErrorsCode.CORRECT;

setLen = (sSetEl.length() + 1) / 2;

if (setLen < MIN\_S || setLen > MAX\_S)

error = ErrorsCode.INCORRECT\_SET\_LENGTH;

return error;

}

public static ErrorsCode isCorrectSetEl(String sSetEl) {

ErrorsCode error;

int i;

error = ErrorsCode.CORRECT;

i = 1;

while (error == ErrorsCode.CORRECT && i < sSetEl.length()) {

if (sSetEl.charAt(i) != ' ')

error = ErrorsCode.INCORRECT\_SET\_El;

i += 2;

}

return error;

}

public static void fillSet(String sSetEl, Set<Character> setEl) {

int i;

for (i = 0; i < sSetEl.length(); i += 2)

setEl.add(sSetEl.charAt(i));

}

public static int chooseOption(int amount) {

ErrorsCode error;

String sOption;

int iOption;

iOption = 0;

do {

error = ErrorsCode.CORRECT;

sOption = scanConsole.nextLine();

try {

iOption = Integer.parseInt(sOption);

} catch (NumberFormatException e) {

error = ErrorsCode.INCORRECT\_CHOICE;

}

if (error == ErrorsCode.CORRECT && (iOption < 1 || iOption > amount))

error = ErrorsCode.INCORRECT\_CHOICE;

if (error != ErrorsCode.CORRECT)

System.out.print(ERRORS[error.ordinal()] + "\nПовторите попытку: \n");

} while (error != ErrorsCode.CORRECT);

return iOption;

}

public static String getPartStr(String str, int posStart, int posEnd) {

String partStr;

int i;

partStr = "";

for (i = posStart; i <= posEnd; i++)

partStr = partStr + str.charAt(i);

return partStr;

}

public static ErrorsCode isFileTXT(String pathToFile) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

if (pathToFile.length() < 5 ||

!getPartStr(pathToFile, pathToFile.length() - 4,

pathToFile.length() - 1).equals(".txt"))

error = ErrorsCode.IS\_NOT\_TXT;

return error;

}

public static ErrorsCode isExist(File file) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

if (!file.exists())

error = ErrorsCode.IS\_NOT\_EXIST;

return error;

}

public static ErrorsCode isReadable(File file) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

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

} catch (Exception e) {

error = ErrorsCode.IS\_NOT\_READABLE;

}

return error;

}

public static ErrorsCode isWriteable(File file) {

ErrorsCode error;

error = ErrorsCode.CORRECT;

if (!file.canWrite())

error = ErrorsCode.IS\_NOT\_WRITEABLE;

return error;

}

public static ErrorsCode isCorrectSetAmount(File file) {

ErrorsCode error;

int count, stopAmount;

error = ErrorsCode.CORRECT;

count = 0;

stopAmount = AMOUNT\_S + 1;

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

while (scanFile.hasNextLine() && count != stopAmount) {

scanFile.nextLine();

count++;

}

} catch (Exception e) {}

if (count != AMOUNT\_S)

error = ErrorsCode.INCORRECT\_SET\_AMOUNT;

return error;

}

public static ErrorsCode isCorrectFileSet(File file) {

ErrorsCode error;

String sSetEl;

error = isCorrectSetAmount(file);

sSetEl = "";

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

while (error == ErrorsCode.CORRECT && scanFile.hasNextLine()) {

sSetEl = scanFile.nextLine();

error = isCorrectSetLen(sSetEl);

if (error == ErrorsCode.CORRECT)

error = isCorrectSetEl(sSetEl);

}

} catch (Exception e) {}

return error;

}

public static File getFileNormalReading() {

File file;

ErrorsCode error;

String pathToFile;

System.out.print("Введите путь к файлу с расширением .txt с тремя

множествами, с длинами[" + MIN\_S + "; " + MAX\_S + "]: \n");

do {

pathToFile = scanConsole.nextLine();

error = isFileTXT(pathToFile);

file = new File(pathToFile);

if (error == ErrorsCode.CORRECT)

error = isExist(file);

if (error == ErrorsCode.CORRECT)

error = isReadable(file);

if (error == ErrorsCode.CORRECT)

error = isCorrectFileSet(file);

if (error != ErrorsCode.CORRECT)

System.out.print(ERRORS[error.ordinal()] + "\nПовторите попытку: ");

} while (error != ErrorsCode.CORRECT);

return file;

}

public static File getFileNormalWriting() {

File file;

ErrorsCode error;

String pathToFile;

System.out.print("Введите путь к файлу с расширением .txt для получения

результата: \n");

do {

pathToFile = scanConsole.nextLine();

error = isFileTXT(pathToFile);

file = new File(pathToFile);

if (error == ErrorsCode.CORRECT)

error = isExist(file);

if (error == ErrorsCode.CORRECT)

error = isWriteable(file);

if (error != ErrorsCode.CORRECT)

System.out.print(ERRORS[error.ordinal()] + "\nПовторите попытку: \n");

} while (error != ErrorsCode.CORRECT);

return file;

}

public static void readFileSet(Scanner scanFile, Set<Character> setEl) {

String sSetEl;

sSetEl = scanFile.nextLine();

fillSet(sSetEl, setEl);

}

public static ErrorsCode isCorrectConsoleSet(String sSetEl) {

ErrorsCode error;

error = isCorrectSetLen(sSetEl);

if (error == ErrorsCode.CORRECT)

error = isCorrectSetEl(sSetEl);

return error;

}

public static void readConsoleSet(int num, Set<Character> setEl) {

ErrorsCode error;

String sSetEl;

System.out.print("Введите множество Х" + num + " через пробелы : ");

do {

sSetEl = scanConsole.nextLine();

error = isCorrectConsoleSet(sSetEl);

if (error != ErrorsCode.CORRECT)

System.out.print(ERRORS[error.ordinal()] + "\nПовторите попытку: ");

} while (error != ErrorsCode.CORRECT);

fillSet(sSetEl, setEl);

}

public static void readSets(Set<Character> x1, Set<Character> x2, Set<Character> x3) {

File file;

int option;

System.out.print("Вы хотите: \n");

System.out.print("Вводить множества через файл - 1\n");

System.out.print("Вводить множества через консоль - 2\n");

option = chooseOption(2);

if (option == 1) {

file = getFileNormalReading();

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

readFileSet(scanFile, x1);

readFileSet(scanFile, x2);

readFileSet(scanFile, x3);

} catch (Exception e) {}

}

else {

readConsoleSet(1, x1);

readConsoleSet(2, x2);

readConsoleSet(3, x3);

}

}

public static void uniteSets(Set<Character> y, Set<Character> x1,

Set<Character> x2, Set<Character> x3) {

y.addAll(x1);

y.addAll(x2);

y.addAll(x3);

}

public static void findNums(Set<Character> y1, Set<Character> y) {

for (char element : y)

if (element >= '0' && element <= '9')

y1.add(element);

}

public static void printConsoleResult(Set<Character> y1) {

System.out.print("\nЦифры в множестве: ");

if (y1.isEmpty())

System.out.print("элементов, удовлетворяющих условию, нет!");

else

for (char element : y1)

System.out.print("'" + element + "'; ");

}

public static void printFileResult(Set<Character> y1, File file) {

try(FileWriter writer = new FileWriter(file, true)) {

writer.write("\nЦифры в множестве: ");

if (y1.isEmpty())

writer.write("элементов, удовлетворяющих условию, нет!");

else

for (char element : y1)

writer.write("'" + element + "'; ");

} catch (Exception e) {}

}

public static void printResult(Set<Character> y1) {

File file;

int option;

System.out.print("Вы хотите: \n");

System.out.print("Выводить множесто через файл - 1\n");

System.out.print("Выводить множесто через консоль - 2\n");

option = chooseOption(2);

if (option == 1) {

file = getFileNormalWriting();

printFileResult(y1, file);

}

else

printConsoleResult(y1);

}

public static void main(String[] args) {

Set<Character> x1 = new HashSet<>();

Set<Character> x2 = new HashSet<>();

Set<Character> x3 = new HashSet<>();

Set<Character> y = new HashSet<>();

Set<Character> y1 = new HashSet<>();

printTask();

readSets(x1, x2, x3);

uniteSets(y, x1, x2, x3);

findNums(y1, y);

printResult(y1);

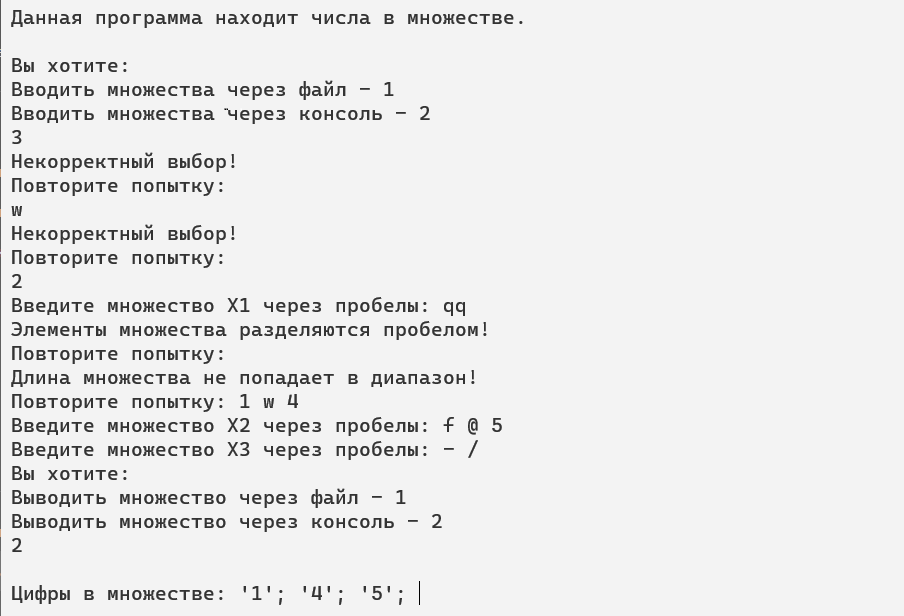
scanConsole.close();

}

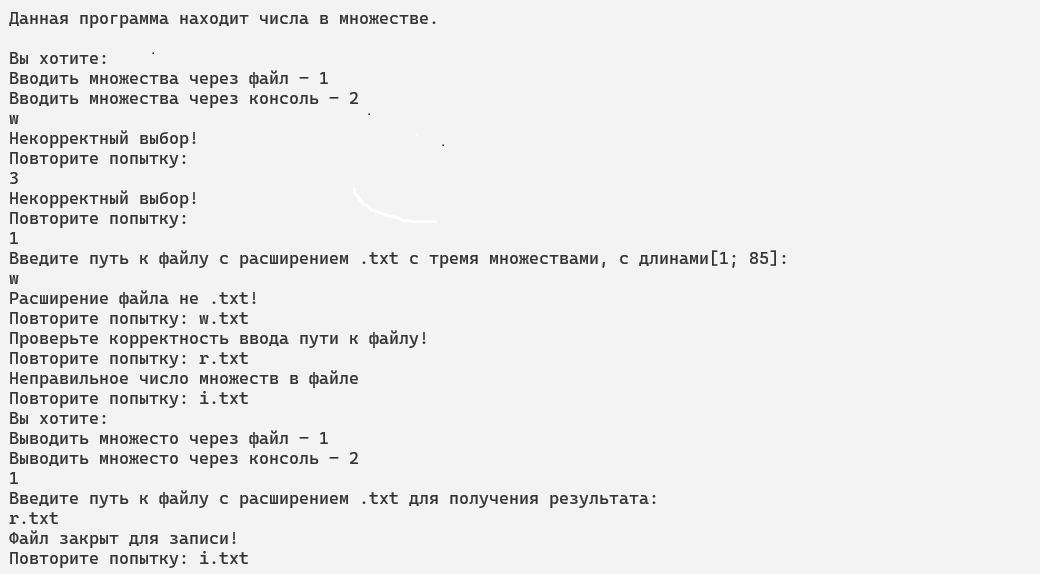
}

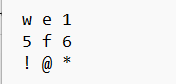
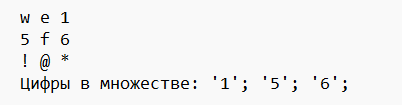
**Скриншоты:**

**Delphi:**

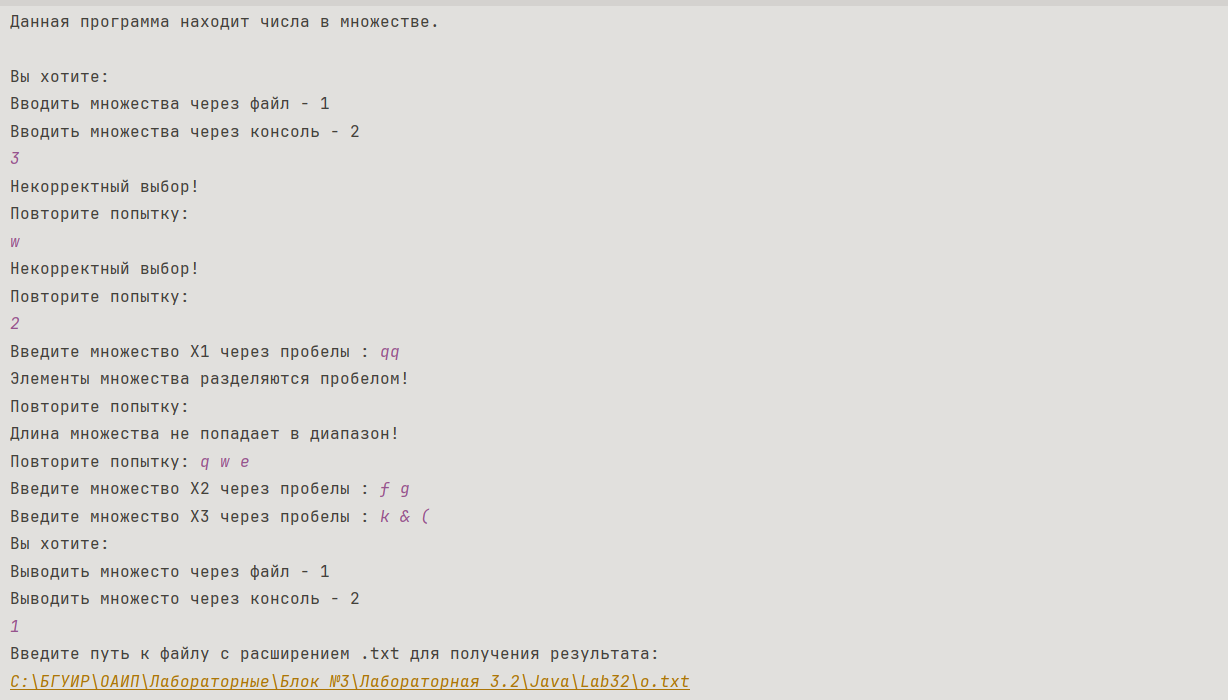


**C++:**



**Java:**



**Блок-схем**



