МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**ЛАБОРАТОРНАЯ РАБОТА № 4**

по дисциплине

**«Объектно-ориентированное программирование»**

на тему:

**«Контейнеры STL»**

*Вариант № 14*

Выполнил:

Студент группы

Ктбо2-8

Жалнин Д. И.

Проверил:

Тарасов С. А.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Таганрог 2020

# Оглавление

[Оглавление 2](#_Toc59067188)

[1 Формулировка задания 3](#_Toc59067189)

[2 Спецификация классов 4](#_Toc59067190)

[2.1 Класс Elections 4](#_Toc59067191)

[2.2 Класс FileReader 5](#_Toc59067192)

[2.3 Класс Interactor 5](#_Toc59067193)

[3 Диаграмма классов 6](#_Toc59067194)

[4 Листинг программы 7](#_Toc59067195)

[4.1 lab4.cpp 7](#_Toc59067196)

[4.2 Interactor.h 7](#_Toc59067197)

[4.3 Interactor.cpp 7](#_Toc59067198)

[4.4 Elections.h 8](#_Toc59067199)

[4.5 Elections.cpp 8](#_Toc59067200)

[4.6 FileReader.h 9](#_Toc59067201)

[4.7 FileReader.cpp 10](#_Toc59067202)

# 1 Формулировка задания

В лабораторной работе должна быть создана программа, содержащая класс «Выборы(Elections)» с полями: ФИО кандидата, дата рождения, место работы, рейтинг предварительных опросов. Необходимо вывести сведения о кандидатах, рейтинг которых превышает вводимое число. Объекты класса должны храниться в одном из контейнеров STL.

При выполнении работы необходимо:

− разработать соответствующие классы, конструкторы, поля и методы

− поля класса сделать закрытыми; для чтения и изменения их значений определить открытые методы;

– предусмотреть во всех вариантах консольный ввод данных для создания объектов и консольный вывод результатов.

# 2 Спецификация классов

## 2.1 Класс Elections

Приватные поля:

**std::string fullName;**

**std::string birthDate;**

**std::string work;**

**double rating;**

/\* Поля, содержащие ФИО, дату рождения, место работы и рейтинг человека \*/

Публичные поля:

**Elections(std::string fullname, std::string birthDate, std::string work, const std::string &rating);**

/\* Конструктор, принимающий необходимую для создания объекта информацию, которая была считана с текстового файла \*/

**std::string toString();**

/\* Метод, возвращающий информацию, хранимую в объекте в виде строки (если быть точным в JSON формате) \*/

**bool operator<(const Elections &e);**

/\* Переопределенный оператор «меньше», необходимый для правильной сортировки «вектора» с объектами данного класса\*/

**std::string getFullName();**

**std::string getWork();**

**std::string getBirthDate();**

**double getRating();**

// Методы, возвращающие значения полей объекта

## 2.2 Класс FileReader

Класс, реализующий чтение информации из файла, ее парсинг и создание объекта **std::vector<Elections>**. Так как доступ к файловой системе желательно производить из одного места в программе (во избежание ошибок ввода вывода), класс не содержит публичных конструкторов и взаимодействие с ним происходит через статический метод **read**.

Считывание происходит из файла «in.txt» в директории проекта.

#pragma once

**#include <iostream>**

**#include <fstream>**

**#include <string>**

**#include <vector>**

**#include "Elections.h"**

**class FileReader {**

**private:**

**FileReader()= default;**

**public:**

**static std::vector<Elections> read();**

/\* Метод преобразования информации из файла в контейнер с помощью построчного считывания ее из файла функцией **std::ifstream** и разделения командой **substr**. \*/

**};**

## 2.3 Класс Interactor

Класс-обертка, реализующий взаимодействие пользователя с основной логикой программы.

**#include "FileReader.h"**

**#include <iostream>**

**#include "vector"**

**#include <algorithm>**

**using namespace std;**

**class Interactor {**

**private:**

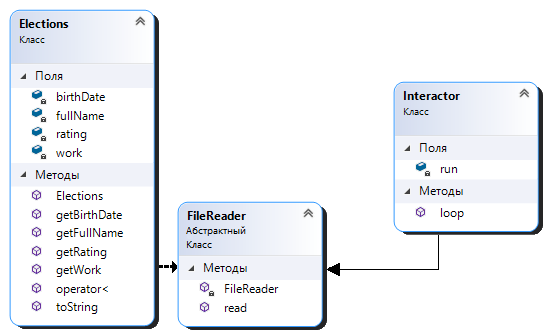
**bool run = true;**

**public:**

**void loop();**

**};**

# 3 Диаграмма классов



*Рисунок 1 – UML диаграмма классов.*

# 4 Листинг программы

## 4.1 lab4.cpp

#include "Interactor.h"

#include <vld.h>

#include <locale.h>

int main() {

setlocale(LC\_ALL, "Ru");

auto\* interactor = new Interactor();

interactor->loop();

delete interactor;

return 0;

}

## 4.2 Interactor.h

#pragma once

#include "FileReader.h"

#include <iostream>

#include "vector"

#include <algorithm>

using namespace std;

class Interactor {

private:

bool run = true;

public:

void loop();

};

## 4.3 Interactor.cpp

#include "Interactor.h"

void Interactor::loop() {

double t;

vector <Elections>v;

while (run) {

bool init = false;

cout << "\nContinue with the past vector\nLoad file - \"1\"\nExit - \"0\"\n";

cin >> t;

if (t == 0) {

run = false;

}

else if (t == 1) {

try {

v = FileReader::read();

cout << "\nLoading has been completed!\n\n";

}

catch (std::exception& e) {

cout << e.what() << "\n";

continue;

}

cout << "Enter the limiter (rating)\n";

cin >> t;

std::sort(v.begin(), v.end());

init = true;

}

else if (t == 2) {

if (!init) {

cout << "Vector has not been initialized!\n";

}

auto i = v.begin();

auto end = v.end();

cout << "\*\*\*BEGIN\*\*\*\n";

while (i != end && (\*i).getRating() > t) {

cout << (\*i).toString() << "\n\n";

i++;

}

cout << "\*\*\*END\*\*\*\n";

v.clear();

}

}

}

## 4.4 Elections.h

#pragma once

#include <string>

class Elections {

private:

std::string fullName;

std::string birthDate;

std::string work;

double rating;

public:

Elections(std::string fullname, std::string birthDate, std::string work, const std::string &rating);

std::string toString();

bool operator<(const Elections &e);

std::string getFullName();

std::string getWork();

std::string getBirthDate();

double getRating();

};

## 4.5 Elections.cpp

#include "Elections.h"

Elections::Elections(std::string fullname, std::string birthDate, std::string work, const std::string &rating) {

this->fullName = std::move(fullname);

this->birthDate = std::move(birthDate);

this->work = std::move(work);

this->rating = std::stod(rating);

}

double Elections::getRating() {

return rating;

}

std::string Elections::getBirthDate() {

return birthDate;

}

std::string Elections::getWork() {

return work;

}

std::string Elections::getFullName() {

return fullName;

}

bool Elections::operator<(const Elections &e) {

if (rating > e.rating) {

return true;

} else {

if (rating == e.rating && fullName > e.fullName) {

return true;

} else {

if (rating == e.rating && fullName == e.fullName && work > e.work) {

return true;

} else {

if (rating == e.rating && fullName == e.fullName && work == e.work && birthDate > e.birthDate) {

return true;

}

}

}

}

return false;

}

std::string Elections::toString() {

return "Elections { "

"fullName: '" + fullName +

"', birthDate: '" + birthDate +

"', work: '" + work +

"', rating: '" + std::to\_string(rating) +

"' }";

}

## 4.6 FileReader.h

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include "Elections.h"

class FileReader {

private:

FileReader()= default;

public:

static std::vector<Elections> read();

};

## 4.7 FileReader.cpp

#include "FileReader.h"

std::vector<Elections> FileReader::read() {

std::vector <Elections> elections;

std::ifstream in("in.txt");

std::string line;

if (!in.is\_open()){

in.close();

throw std::exception("Can not open the file!");

} else {

std::string name1;

std::string name2;

std::string name3;

std::string date;

std::string work;

std::string rating;

while(getline(in, line)){

int nameEnd1 = line.find(' ',0);

name1 = line.substr(0,nameEnd1);

int nameEnd2 = line.find(' ',nameEnd1+1);

name2 = line.substr(nameEnd1+1,nameEnd2-nameEnd1-1);

int nameEnd = line.find(' ',nameEnd2+1);

name3 = line.substr(nameEnd2+1,nameEnd-nameEnd2-1);

int dateEnd = line.find(" \"",nameEnd+1);

date = line.substr(nameEnd+1, dateEnd-nameEnd-1);

int workEnd = line.find("\" ",dateEnd+1);

work = line.substr(dateEnd+2, workEnd-dateEnd-2);

int ratingEnd = line.length()-1;

rating = line.substr(workEnd+2, ratingEnd-workEnd-1);

elections.push\_back(Elections(name1.append(" ").append(name2).append(" ").append(name3), date, work, rating));

}

in.close();

return std::move(elections);

}

}