МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**Лабораторная работа № 4**

по курсу «Объектно-ориентированное программирование»

**«Контейнеры STL»**

**14 ВАРИАНТ**

Выполнили:

студенты гр. КТбо2-7

Миронченко П.Д.

Проверил:

Тарасов С. А.

**Таганрог 2020**

1. **Вариант задания №14**

Класс «Выборы (Elections)» с полями: ФИО кандидата, дата рождения, место работы, рейтинг предварительных опросов; вывести сведения о кандидатах, рейтинг которых превышает вводимое число.

1. **Спецификации классов**

**Спецификация класса Interactor**

Класс Interactor представляет собой фасад. Содержит приватное поле указателя на интерфейс IElectionsContainer и методы для взаимодействия с пользователем и манипуляций с данными

**Спецификация класса Elections**

Содержит в себе приватные поля ФИО, даты рождения, места работы и рейтинга, а также геттеры к ним

**Спецификация класса Date**

Содержит в себе приватные поля дня, месяца и года предназначенные для хранения даты рождения в классе Electtions

**Спецификация класса IElectionsContainer**

Представляет собой интерфейс, требуемый для динамического выбора классов ElectionsMap и ElectionsVector. Содержит виртуальные методы для добавления элемента и поиска элементов, которые рейтингом выше заданного в аргументах

**Спецификация класса ElectionsMap**

Представляет собой реализацию интерфейса IElectionsContainer, содержит приватное поле типа map <float, Elections> и реализацию методов интерфейса

**Спецификация класса ElectionsVector**

Представляет собой реализацию интерфейса IElectionsContainer, содержит приватное поле типа vector <Elections> и реализацию методов интерфейса

**Спецификация класса ContainerFactory**

Представляет собой фабрику, которая выдаёт объекты, реализующие интерфейс IElectionsContainer, по запросу пользователя

1. **Используемые математические зависимости и алгоритмы**

В данной программе использеются алгоритм линейного поиска

1. **Диаграмма класса**

Diagram

Description automatically generated

1. **Листинг программы**

**Lab4.cpp**

#include <iostream>

#include <fstream>

#include "interactor.h"

using namespace std;

int main()

{

Interactor it;

it.Init("in.txt");

return 0;

}

**Interactor.h**

#ifndef INTERACTOR\_H

#define INTERACTOR\_H

#include <iostream>

#include <fstream>

#include <vector>

#include "IElectionsContainer.h"

#include "ContainerFactory.h"

class Interactor

{

public:

Interactor();

~Interactor();

void Init(const std::string& file\_to\_input);

private:

IElectionsContainer\* \_container;

void \_Help() const;

void \_FileParse(const std::string& file\_to\_input);

void \_Interaction(const std::string& file\_to\_input);

void \_ChooseContainer();

};

#endif

**Date.h**

#ifndef DATE\_H

#define DATE\_H

#include <iostream>

class Date

{

public:

Date(int dd, int mm, int yy);

~Date() = default;

int GetDay() const;

int GetMonth() const;

int GetYear() const;

friend std::ostream& operator<<(std::ostream& out, const Date& object);

private:

int \_day;

int \_month;

int \_year;

};

#endif *// !DATE\_H*

**Elections.h**

#ifndef ELECTIONS\_H

#define ELECTIONS\_H

#include <string>

#include "Date.h"

class Elections

{

public:

Elections(const std::string& fullName,

const Date& dateOfBirth,

const std::string& job,

const double& rating);

~Elections() = default;

std::string GetFullName() const;

Date GetDateOfBirth() const;

std::string GetJob() const;

double GetRating() const;

friend std::ostream& operator<<(std::ostream& out, const Elections& object);

private:

std::string \_fullName;

Date \_dateOfBirth;

std::string \_job;

double \_rating;

};

#endif *// !ELECTIONS\_H*

**IElectionsContainer.h**

#ifndef I\_ELECTIONS\_CONTAINER\_H

#define I\_ELECTIONS\_CONTAINER\_H

#include "Elections.h"

#include <vector>

class IElectionsContainer

{

public:

IElectionsContainer() = default;

virtual void AddItem(const Elections& item) = 0;

virtual std::vector<Elections> Search(double rait) const = 0;

virtual ~IElectionsContainer() {};

};

#endif *// !I\_ELECTIONS\_CONTAINER\_H*

**ElectionsVector.h**

#ifndef ELECTIONS\_VECTOR\_H

#define ELECTIONS\_VECTOR\_H

#include <vector>

#include "Elections.h"

#include "IElectionsContainer.h"

class ElectionsVector : public IElectionsContainer

{

public:

void AddItem(const Elections& item) override;

std::vector<Elections> Search(double rait) const override;

~ElectionsVector() override = default;

private:

std::vector<Elections> \_vec;

};

#endif *// !ELECTIONS\_VECTOR\_H*

**ElectionsMap.h**

#ifndef ELECTIONS\_MAP\_H

#define ELECTIONS\_MAP\_H

#include <vector>

#include <map>

#include "Elections.h"

#include "IElectionsContainer.h"

class ElectionsMap : public IElectionsContainer

{

public:

ElectionsMap() = default;

~ElectionsMap() override = default;

void AddItem(const Elections& item) override;

std::vector<Elections> Search(double rait) const override;

private:

std::multimap<double, Elections> \_map;

};

#endif *// !ELECTIONS\_MAP\_H*

**ContainerFactory.h**

#ifndef CONTAINER\_FACTORY\_H

#define CONTAINER\_FACTORY\_H

#include "IElectionsContainer.h"

#include "ElectionsMap.h"

#include "ElectionsVector.h"

class ContainerFactory

{

public:

ContainerFactory() = default;

~ContainerFactory() = default;

IElectionsContainer\* GetContainer(char type);

};

#endif *// !CONTAINER\_FACTORY\_H*

**Interactor.cpp**

#include "Interactor.h"

Interactor::Interactor(): \_container(nullptr)

{

}

Interactor::~Interactor()

{

if (\_container != nullptr)

{

delete \_container;

}

}

void Interactor::\_Help() const

{

std::cout <<"===========================================\n";

std::cout << "Commands: \n";

std::cout << "1) Add new item\n";

std::cout << "2) Search for objects with higher ratingt\n";

std::cout << "3) Change container type\n";

std::cout << "4) Help\n";

std::cout << "0) Exit\n";

std::cout <<"===========================================\n";

}

void Interactor::\_Interaction(const std::string& file\_to\_input)

{

int command = -1;

while (command != 0)

{

std::cout <<"Please, enter your command: ";

std::cin >> command;

switch (command)

{

case 1:

try

{

std::cout << "Please, enter full name: ";

std::string firstName;

std::string secondName;

std::string lastName;

std::cin >> firstName >> secondName >> lastName;

std::string fullName = firstName + " " + secondName + " " + lastName;

int d, m, y;

std::cout << "Please, enter a date (dd.mm.yy): ";

std::cin >> d;

std::cin.get();

std::cin >> m;

std::cin.get();

std::cin >> y;

Date date(d,m,y);

std::cout << "Please, enter a place of job: ";

std::string job;

std::cin >> job;

std::cout << "Please, enter a rating: ";

double rait;

std::cin >> rait;

\_container->AddItem(Elections(fullName, date, job,rait));

}

catch (std::logic\_error e)

{

std::cout << "There is so mistakes in data: " << e.what() << '\n';

}

break;

case 2:

try

{

std::cout << "Enter a rating: ";

double rait;

std::cin >> rait;

std::vector<Elections> tmp = \_container->Search(rait);

if (tmp.size() == 0)

std::cout << "No elements\n";

else

{

for (auto e : tmp)

{

std::cout << e;

}

}

}

catch (std::logic\_error e)

{

std::cout << "There is so mistakes in arg: " << e.what() << '\n';

}

break;

case 3:

delete \_container;

\_ChooseContainer();

\_FileParse(file\_to\_input);

break;

case 4:

\_Help();

break;

case 0:

break;

}

}

}

void Interactor::\_FileParse(const std::string& file\_to\_input)

{

int n;

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

fin >> n;

for (int i = 0; i < n; ++i)

{

try {

std::string firstName;

std::string secondName;

std::string lastName;

fin >> firstName >> secondName >> lastName;

std::string fullName = firstName + " " + secondName + " " + lastName;

int d, m, y;

fin >> d;

fin.get();

fin >> m;

fin.get();

fin >> y;

Date date(d,m,y);

std::string job;

fin >> job;

double rait;

fin >> rait;

\_container->AddItem(Elections(fullName, date, job,rait));

}

catch (std::logic\_error e)

{

std::cout << "There is so mistakes in file: " << e.what() << '\n';

exit(1);

}

}

fin.close();

}

void Interactor::\_ChooseContainer()

{

std::cout << "Choose to type of a container [V: Vector, M: Map]: ";

char type;

std::cin >> type;

ContainerFactory factory;

\_container = factory.GetContainer(tolower(type));

while (\_container == nullptr)

{

std::cout << "The given type doesn't exist. Choose from V and M [V: Vector, M: Map]: ";

std::cin >> type;

\_container = factory.GetContainer(tolower(type));

}

std::cout << "You choose " << (type == 'V' ? "vector\n" : "map\n");

}

void Interactor::Init(const std::string& file\_to\_input)

{

\_ChooseContainer();

\_FileParse(file\_to\_input);

std::cout << "Files parsed from " << file\_to\_input << '\n';

\_Help();

\_Interaction(file\_to\_input);

}

**Date.cpp**

#include "Date.h"

#include <iostream>

Date::Date(int dd, int mm, int yy): \_day(dd), \_month(mm), \_year(yy)

{

if (dd < 1 || dd > 31)

throw std::logic\_error("bad day argument\n");

if (mm < 1 || mm > 12)

throw std::logic\_error("bad mounth argument\n");

if (yy < 0)

throw std::logic\_error("bad year argument\n");

}

int Date::GetDay() const

{

return \_day;

}

int Date::GetMonth() const

{

return \_month;

}

int Date::GetYear() const

{

return \_year;

}

std::ostream& operator<<(std::ostream& out, const Date& object)

{

out << object.\_day << '.' << object.\_month << '.' << object.\_year;

return out;

}

**Elections.cpp**

#include "Elections.h"

#include <iostream>

Elections::Elections(const std::string& fullName,

const Date& dateOfBirth,

const std::string& job,

const double& rating): \_fullName(fullName),

\_dateOfBirth(dateOfBirth),

\_job(job),

\_rating(rating)

{

if (\_rating < 0.0)

throw std::logic\_error("bad rating\n");

}

std::string Elections::GetFullName() const

{

return \_fullName;

}

Date Elections::GetDateOfBirth() const

{

return \_dateOfBirth;

}

std::string Elections::GetJob() const

{

return \_job;

}

double Elections::GetRating() const

{

return \_rating;

}

std::ostream& operator<<(std::ostream& out, const Elections& object)

{

out << "-----------------------------\n";

out << "Full name: " << object.\_fullName <<"\n";

out << "Date of birth: " << object.\_dateOfBirth <<"\n";

out << "Place of job: " << object.\_job <<"\n";

out << "Raiting: " << object.\_rating <<"\n";

out << "-----------------------------\n";

return out;

}

**ElectionsVector.cpp**

#include "ElectionsVector.h"

void ElectionsVector::AddItem(const Elections& item)

{

\_vec.push\_back(item);

}

std::vector<Elections> ElectionsVector::Search(double rait) const

{

if (rait < 0.0 || rait > 100.0)

throw std::logic\_error("bad rating\n");

std::vector<Elections> candidates;

for (int i =0; i < \_vec.size(); ++i)

{

if (\_vec[i].GetRating() > rait)

{

candidates.push\_back(\_vec[i]);

}

}

return candidates;

}

**ElectionsMap.cpp**

#include "ElectionsMap.h"

void ElectionsMap::AddItem(const Elections& item)

{

\_map.insert(std::pair<double, Elections>(item.GetRating(),item));

}

std::vector<Elections> ElectionsMap::Search(double rait) const

{

if (rait < 0.0 || rait > 100.0)

throw std::logic\_error("bad rating\n");

std::vector<Elections> candidates;

for (auto it = \_map.begin(); it != \_map.end(); ++it)

{

if (it->first > rait)

{

candidates.push\_back(it->second);

}

}

return candidates;

}

**ContainerFactory.cpp**

#include "ContainerFactory.h"

IElectionsContainer\* ContainerFactory::GetContainer(char type)

{

if (type == 'v')

{

return new ElectionsVector;

}

else if (type == 'm')

{

return new ElectionsMap;

}

else

{

return nullptr;

}

}