МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Отчет к лабораторной работе № 6

Тема: **Объектно-ориентированный подход. Стандартная библиотека шаблонов**

Дисциплина: **«Информационные технологии и программирование»**

**Выполнил:**

Студент 1 курса группы ПИЖ-б-о-23-1(2) направление подготовки: 09.03.04 «Программная инженерия»

Сабодаш Алексей Игоревич

**Проверила:**

Новикова Е. Н.

Ставрополь 2024

**Вариант 8**

**Задание 1:**

Написать программу учета заявок на обмен квартир и поиска вариантов обмена.

Каждая заявка содержит сведения о двух квартирах: требуемой (искомой) и имеющейся. Сведения о каждой квартире содержат: количество комнат, площадь, этаж,

район.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из

следующих функций:

а ввод заявки на обмен;

□ поиск в картотеке подходящего варианта: при совпадении требований и предложений по количеству комнат и этажности и различии по показателю «площадь» в пределах 10% выводится соответствующая карточка и удаляется

из списка, в противном случае поступившая заявка включается в картотеку;

□ вывод всей картотеки.

Для хранения данных картотеки использовать контейнерный класс list.

#include <iostream>

#include <list>

#include <cmath>

#include <string>

struct Apartment {

int rooms;

double area;

int floor;

std::string district;

bool matches(const Apartment& other) const {

return rooms == other.rooms && floor == other.floor && std::abs(area - other.area) <= 0.1 \* area;

}

};

struct ExchangeRequest {

Apartment wanted;

Apartment offered;

};

class ExchangeRegistry {

private:

std::list<ExchangeRequest> requests;

public:

void addRequest(const ExchangeRequest& request) {

requests.push\_back(request);

}

bool findMatch(const ExchangeRequest& newRequest) {

for (auto it = requests.begin(); it != requests.end(); ++it) {

if (it->wanted.matches(newRequest.offered) && newRequest.wanted.matches(it->offered)) {

std::cout << "Найден подходящий вариант обмена:\n";

std::cout << "Искомая квартира: Комнат: " << it->wanted.rooms << ", Площадь: " << it->wanted.area

<< ", Этаж: " << it->wanted.floor << ", Район: " << it->wanted.district << '\n';

std::cout << "Предлагаемая квартира: Комнат: " << it->offered.rooms << ", Площадь: " << it->offered.area

<< ", Этаж: " << it->offered.floor << ", Район: " << it->offered.district << '\n';

requests.erase(it);

return true;

}

}

return false;

}

void printAllRequests() const {

for (const auto& request : requests) {

std::cout << "Искомая квартира: Комнат: " << request.wanted.rooms << ", Площадь: " << request.wanted.area

<< ", Этаж: " << request.wanted.floor << ", Район: " << request.wanted.district << '\n';

std::cout << "Предлагаемая квартира: Комнат: " << request.offered.rooms << ", Площадь: " << request.offered.area

<< ", Этаж: " << request.offered.floor << ", Район: " << request.offered.district << '\n';

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

}

}

};

void printMenu() {

std::cout << "Меню:\n";

std::cout << "1. Ввод заявки на обмен\n";

std::cout << "2. Поиск подходящего варианта обмена\n";

std::cout << "3. Вывод всей картотеки\n";

std::cout << "4. Выйти\n";

std::cout << "Выберите опцию: ";

}

Apartment inputApartment(const std::string& type) {

Apartment apartment;

std::cout << "Введите данные для " << type << " квартиры:\n";

std::cout << "Количество комнат: ";

std::cin >> apartment.rooms;

std::cout << "Площадь: ";

std::cin >> apartment.area;

std::cout << "Этаж: ";

std::cin >> apartment.floor;

std::cin.ignore(); // Игнорировать оставшийся символ новой строки

std::cout << "Район: ";

std::getline(std::cin, apartment.district);

return apartment;

}

int main()

{

setlocale(0, "rus");

ExchangeRegistry registry;

int choice;

do {

printMenu();

std::cin >> choice;

std::cin.ignore(); // Игнорировать оставшийся символ новой строки

if (choice == 1) {

ExchangeRequest request;

request.wanted = inputApartment("искомой");

request.offered = inputApartment("предлагаемой");

registry.addRequest(request);

}

else if (choice == 2) {

ExchangeRequest newRequest;

newRequest.wanted = inputApartment("искомой");

newRequest.offered = inputApartment("предлагаемой");

if (!registry.findMatch(newRequest)) {

registry.addRequest(newRequest);

std::cout << "Подходящих вариантов не найдено. Заявка добавлена в картотеку.\n";

}

}

else if (choice == 3) {

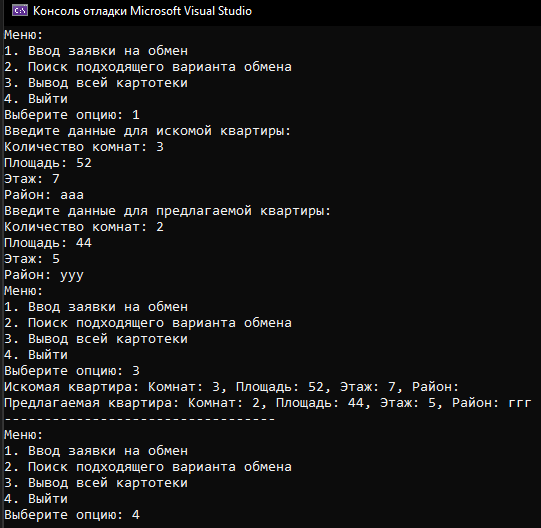
registry.printAllRequests();

}

} while (choice != 4);

return 0;

}



**https://github.com/strit595/Progrm**