МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

**Лабораторная работа №6**

по дисциплине

«Объектно-ориентированное программирование»

**Выполнил:**

Ермолаев Кирилл Александрович

Студент 2 курса группы \_ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

Ставрополь, 2023 г.

**Тема:** Основы объектно-ориентированного программирования на ЯП С++.

**Цель работы:** Стандартная библиотека шаблонов.

**Ход работы:**

**Вариант 7**

Написать программу «Моя записная книжка». Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям.

Хранение данных организовать с помощью класса map или multimap.

// main.cpp

#include <iostream>

#include "address\_book.h"

int main() {

AddressBook addressBook;

// Добавление нескольких контактов

addressBook.addContact(Contact("John Doe", "123456789", "01/01/1990"));

addressBook.addContact(Contact("Alex Smith", "987654321", "02/15/1985"));

addressBook.addContact(Contact("Bob Johnson", "555111222", "05/20/1978"));

addressBook.addContact(Contact("Alice Smith", "555333444", "03/10/1995"));

std::cout << "Sorted by name:\n";

addressBook.sortByname();

std::cout << "\nSorted by phone number:\n";

addressBook.sortByPhoneNumber();

std::cout << "\nSorted by birthdate:\n";

addressBook.sortByBirthdate();

std::cout << "\nSearch by name (Alice Smith):\n";

addressBook.searchByName("Alice Smith");

std::cout << "\nRemove contact (John Doe):\n";

addressBook.removeContact("John Doe");

addressBook.sortByname();

return 0;

}

// address\_book.h

#pragma once

#ifndef ADDRESS\_BOOK\_H

#define ADDRESS\_BOOK\_H

#include <map>

#include <string>

#include <vector>

#include "contact.h"

class AddressBook {

public:

// Добавление контакта

void addContact(const Contact& contact);

// Удаление контакта по имени

void removeContact(const std::string& name);

// Поиск контакта по имени

void searchByName(const std::string& name) const;

// Сортировка по имени

void sortByname() const;

// Сортировка по номеру телефона

void sortByPhoneNumber() const;

// Сортировка по дате рождения

void sortByBirthdate() const;

const std::multimap<std::string, Contact>& getContacts() const {

return contacts;

}

private:

std::multimap<std::string, Contact> contacts;

};

#endif

// address\_book.h

#include "address\_book.h"

void AddressBook::addContact(const Contact& contact) {

contacts.insert(std::make\_pair(contact.getName(), contact));

}

void AddressBook::removeContact(const std::string& name) {

auto it = contacts.find(name);

if (it != contacts.end()) {

contacts.erase(it);

}

}

void AddressBook::searchByName(const std::string& name) const {

auto range = contacts.equal\_range(name);

for (auto it = range.first; it != range.second; ++it) {

std::cout << it->second;

}

}

void AddressBook::sortByname() const {

for (const auto& entry : contacts) {

std::cout << entry.second;

}

}

void AddressBook::sortByPhoneNumber() const {

std::multimap<std::string, Contact, std::less<>> sortedContacts;

for (const auto& entry : contacts) {

sortedContacts.insert(std::make\_pair(entry.second.getPhoneNumber(), entry.second));

}

for (const auto& entry : sortedContacts) {

std::cout << entry.second;

}

}

void AddressBook::sortByBirthdate() const {

std::multimap<std::string, Contact, std::less<>> sortedContacts;

for (const auto& entry : contacts) {

sortedContacts.insert(std::make\_pair(entry.second.getBirthdate(), entry.second));

}

for (const auto& entry : sortedContacts) {

std::cout << entry.second;

}

}

// contact.h

#pragma once

#ifndef CONTACT\_H

#define CONTACT\_H

#include <iostream>

#include <string>

class Contact {

public:

Contact(const std::string& name, const std::string& phoneNumber, const std::string& birthdate);

// Перегрузка оператора вывода для удобного вывода контакта

friend std::ostream& operator<<(std::ostream& os, const Contact& contact);

// Дополнительные методы для получения значений полей

const std::string& getName() const;

const std::string& getPhoneNumber() const;

const std::string& getBirthdate() const;

const std::string getAllInfo() const;

private:

std::string name;

std::string phoneNumber;

std::string birthdate;

};

#endif

// contact.cpp

#include "contact.h"

Contact::Contact(const std::string& name, const std::string& phoneNumber, const std::string& birthdate)

: name(name), phoneNumber(phoneNumber), birthdate(birthdate) {}

std::ostream& operator<<(std::ostream& os, const Contact& contact) {

os << "Name: " << contact.name << "\nPhone Number: " << contact.phoneNumber << "\nBirthdate: " << contact.birthdate << "\n";

return os;

}

const std::string& Contact::getName() const {

return name;

}

const std::string& Contact::getPhoneNumber() const {

return phoneNumber;

}

const std::string& Contact::getBirthdate() const {

return birthdate;

}

const std::string Contact::getAllInfo() const {

return "Name: " + name + "\nPhone Number: " + phoneNumber + "\nBirthdate: " + birthdate + "\n";

}

// UnitTest6.cpp

#include "pch.h"

#include "CppUnitTest.h"

#include <string>

#include <vector>

#include <iostream>

#include "..\ConsoleApplication1\contact.h"

#include "..\ConsoleApplication1\contact.cpp"

#include "..\ConsoleApplication1\address\_book.h"

#include "..\ConsoleApplication1\address\_book.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

TEST\_CLASS(AddressBookTests)

{

public:

TEST\_METHOD(TestAddContact)

{

AddressBook addressBook;

Contact contact("John Doe", "123456789", "01/01/1990");

addressBook.addContact(contact);

Assert::AreEqual(static\_cast<size\_t>(1), addressBook.getContacts().size());

}

TEST\_METHOD(TestRemoveContact)

{

AddressBook addressBook;

Contact contact("John Doe", "123456789", "01/01/1990");

addressBook.addContact(contact);

addressBook.removeContact("John Doe");

Assert::IsTrue(addressBook.getContacts().empty());

}

TEST\_METHOD(TestSearchByName)

{

AddressBook addressBook;

Contact contact1("John Doe", "123456789", "01/01/1990");

Contact contact2("Alice Smith", "987654321", "02/15/1985");

addressBook.addContact(contact1);

addressBook.addContact(contact2);

std::vector<Contact> searchResults;

for (const auto& entry : addressBook.getContacts()) {

if (entry.second.getName() == "Alice Smith") {

searchResults.push\_back(entry.second);

}

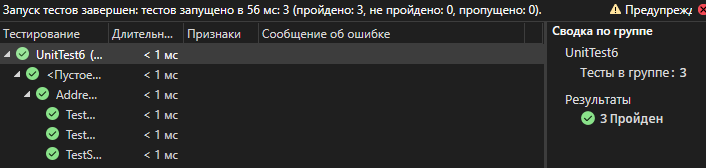
}

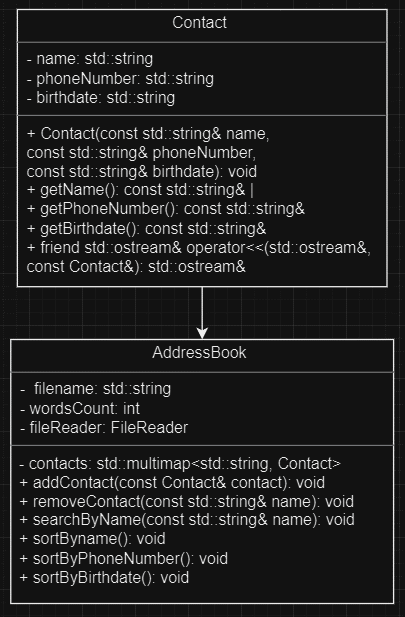
Assert::AreEqual(static\_cast<size\_t>(1), searchResults.size());

Assert::AreEqual("Name: Alice Smith\nPhone Number: 987654321\nBirthdate: 02/15/1985\n", searchResults[0].getAllInfo().c\_str());

}

};





<https://github.com/MoeTomatoki/OOP>

**Вывод:** изучил базовые понятия стандартной библиотеки шаблонов.