ФАКУЛЬТЕТ «Информатика и системы управления»  
КАФЕДРА «Информационная безопасность»

ОТЧЕТ  
по лабораторной работе № 5  
по учебной дисциплине «Алгоритмические языки»  
на тему: « Изучение использования объектов своих классов в упорядоченных и неупорядоченных контейнерах библиотеки STL в языке Си++»

Вариант 23

Выполнил:   
Студент 1 курса, гр. ИУ8-24  
Терехин Данила

**Цель работы:**

Практическое применение контейнеров стандартной библиотеки шаблонов (STL) для хранения объектов пользовательского класса, а также использование методов и алгоритмов STL для работы с этими контейнерами.

**Задачи работы:**

* Изучить необходимые учебные материалы, посвященные упорядоченным и неупорядоченным контейнерам стандартной библиотеки шаблонов в языке Си++
* Разработать программу на языке Си++ для решения заданного варианта задания
* Отладить программу
* Представить результаты работы программы
* Подготовить отчет по лабораторной работе

**Выполнение работы:**

**Main.cpp**

#include <iostream>

#include <string>

#include <vector>

#include <set>

#include <unordered\_set>

#include <algorithm>

#include <fstream>

#include <functional>

class BankCredit {

public:

    // Конструктор для инициализации объекта BankCredit с заданными параметрами

    BankCredit(const std::string& name, double amount, const std::string& currency, double rate)

        : name(name), amount(amount), currency(currency), rate(rate) {}

    // Конструктор копирования для создания копии объекта BankCredit

    BankCredit(const BankCredit& other)

        : name(other.name), amount(other.amount), currency(other.currency), rate(other.rate) {}

    // Оператор присваивания копированием для копирования значений из другого объекта BankCredit

    BankCredit& operator=(const BankCredit& other) {

        if (this != &other) {

            name = other.name;

            amount = other.amount;

            currency = other.currency;

            rate = other.rate;

        }

        return \*this;

    }

    // Перегруженный оператор < для сравнения объектов по ставке кредита

    bool operator<(const BankCredit& other) const {

        return rate < other.rate;

    }

    // Методы доступа

    const std::string& getName() const { return name; }

    double getAmount() const { return amount; }

    const std::string& getCurrency() const { return currency; }

    double getRate() const { return rate; }

    // Объявление оператора << как друга класса

    friend std::ostream& operator<<(std::ostream& os, const BankCredit& credit);

private:

    std::string name; // Название кредита

    double amount; // Сумма кредита

    std::string currency; // Тип валюты

    double rate; // Ставка в процентах годовых

};

// Определение перегруженного оператора <<

std::ostream& operator<<(std::ostream& os, const BankCredit& credit) {

    os << credit.getName() << " " << credit.getAmount() << " " << credit.getCurrency() << " " << credit.getRate();

    return os;

}

// Хеш-функция для unordered\_set

struct BankCreditHash {

    std::size\_t operator()(const BankCredit& bc) const {

        using std::size\_t;

        using std::hash;

        using std::string;

        size\_t res = 17;

        res = res \* 31 + hash<string>()(bc.getName());

        res = res \* 31 + hash<double>()(bc.getAmount());

        res = res \* 31 + hash<string>()(bc.getCurrency());

        res = res \* 31 + hash<double>()(bc.getRate());

        return res;

    }

};

// Оператор равенства для unordered\_set

struct BankCreditEqual {

    bool operator()(const BankCredit& lhs, const BankCredit& rhs) const {

        return lhs.getName() == rhs.getName() && lhs.getAmount() == rhs.getAmount() && lhs.getCurrency() == rhs.getCurrency() && lhs.getRate() == rhs.getRate();

    }

};

int main() {

    std::ifstream input("input.txt"); // Открытие файла для чтения

    std::ofstream output("output.txt"); // Открытие файла для записи

    if (!input.is\_open() || !output.is\_open()) {

        std::cerr << "Error opening file!" << std::endl;

        return 1;

    }

    std::vector<BankCredit> creditsVector; // Вектор для хранения объектов BankCredit

    std::set<BankCredit> creditsSet; // Множество для хранения объектов BankCredit (сортировка по ставке)

    std::unordered\_set<BankCredit, BankCreditHash, BankCreditEqual> creditsUnorderedSet; // Неупорядоченное множество для хранения объектов BankCredit

    std::string name;

    double amount, rate;

    std::string currency;

    // Чтение данных из файла и заполнение контейнеров

    while (input >> name >> amount >> currency >> rate) {

        BankCredit credit(name, amount, currency, rate);

        creditsVector.push\_back(credit);

        creditsSet.insert(credit);

        creditsUnorderedSet.insert(credit);

    }

    if (input.bad()) {

        std::cerr << "Error reading file!" << std::endl;

        return 1;

    }

    // Вывод исходного вектора в файл

    output << "Original Vector:\n";

    for (const auto& credit : creditsVector) {

        output << credit << "\n";

    }

    // Вывод set в файл

    output << "Set:\n";

    for (const auto& credit : creditsSet) {

        output << credit << "\n";

    }

    // Вывод unordered\_set в файл

    output << "Unordered Set:\n";

    for (const auto& credit : creditsUnorderedSet) {

        output << credit << "\n";

    }

    if (!output) {

        std::cerr << "Error writing to file!" << std::endl;

        return 1;

    }

    input.close(); // Закрытие файла для чтения

    output.close(); // Закрытие файла для записи

    return 0;

}

Файл со входными данными– **input.txt:**Credit1 1000 USD 2.0

Credit2 4200 EUR 3.5

Credit4 3000 JPY 2.1

Credit3 1500 GBP 4.2

Файл с выходными данными **– output.txt:**

Original Vector:

Credit1 1000 USD 2

Credit2 4200 EUR 3.5

Credit4 3000 JPY 2.1

Credit3 1500 GBP 4.2

Set:

Credit1 1000 USD 2

Credit4 3000 JPY 2.1

Credit2 4200 EUR 3.5

Credit3 1500 GBP 4.2

Unordered Set:

Credit3 1500 GBP 4.2

Credit2 4200 EUR 3.5

Credit4 3000 JPY 2.1

Credit1 1000 USD 2

**Вывод:**

В ходе выполнения лабораторной работы было успешно реализовано использование объектов собственных классов в упорядоченных и неупорядоченных контейнерах библиотеки STL с применением различных методов и алгоритмов для работы с ними. Это позволило познакомиться с применением стандартной библиотеки шаблонов в практических задачах программирования.