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

ОТЧЕТ  
по лабораторной работе № 4  
по учебной дисциплине «Алгоритмические языки»  
на тему: «Изучение возможностей наследования классов в языке Си++»

Вариант 23

Выполнил:   
Студент 1 курса, гр. ИУ8-24  
Спиридонов Олег

**Цель работы:**

Изучить принципы использования объектов своих классов в последовательных контейнерах библиотеки STL.

**Условие задачи:**

Вариант № 23

Создать базовый класс «банковский кредит» (поля: название, сумма кредита, тип валюты, ставка в % годовых); конструкторы, конструктор копирования и перемещения при необходимости (обосновать отсутствие или наличие необходимости), перегруженные операции присваивания с копированием и перемещением при необходимости (обосновать отсутствие или наличие необходимости), перегруженную операцию вставки в поток <<.

Обеспечить копирование одного контейнера (vector) в другой (deque) с помощью алгоритма std::copy и сортировку объектов в исходном контейнере vector с использованием алгоритма std::sort с двумя параметрами: итератор на начало и итератор на конец контейнера.

Исходные данные прочитать из текстового файла input.txt. Вывести в выходной файл output.txt исходный контейнер, контейнер после сортировки, использовать при этом перегруженную операцию вставки в поток, также вывести в выходной файл контейнер, в который скопирован исходный контейнер

Файл **input.txt**:

Oleg 5000 ruble 5

Egor 3000 dollar 3

Vlad 4000 euro 4

Uzbek 2000 ruble

**Текст программы:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <algorithm>

#include <string>

#include <vector>

#include <deque>

using namespace std;

class Credit {

friend ostream& operator<<(ostream& os, Credit& Credit);

friend istream& operator>>(istream& is, Credit& Credit);

friend bool operator<(const Credit&, const Credit&);

private:

string name;

int sum;

string currency;

int annual\_rate;

public:

Credit();

Credit(string name, int sum, string currency, int annual\_rate);

Credit(const Credit& Credit\_copied);

};

Credit::Credit() {

this->name = "";

this->sum = 0;

this->currency = "";

this->annual\_rate = 0;

};

Credit::Credit(string name, int sum, string currency, int annual\_rate) {

this->name = name;

this->sum = sum;

this->currency = currency;

this->annual\_rate = annual\_rate;

};

Credit::Credit(const Credit& Credit\_copied) {

this->name = Credit\_copied.name;

this->sum = Credit\_copied.sum;

this->currency = Credit\_copied.currency;

this->annual\_rate = Credit\_copied.annual\_rate;

};

ostream& operator<<(ostream& os, Credit& Credit) {

fstream file("output.txt", fstream::app);

if (file.is\_open()) {

file << Credit.name << ' ' << Credit.sum << ' ' << Credit.currency << ' ' << Credit.annual\_rate << endl;

}

else {

cout << "Output file didn't open." << endl;

}

os << Credit.name << ' ' << Credit.sum << ' ' << Credit.currency << ' ' << Credit.annual\_rate << '%' << endl;

return os;

};

istream& operator>>(istream& is, Credit& Credit) {

is >> Credit.name;

is >> Credit.sum;

is >> Credit.currency;

is >> Credit.annual\_rate;

return is;

};

bool operator<(const Credit& Credit1, const Credit& Credit2) {

return Credit1.annual\_rate < Credit2.annual\_rate;

};

int main() {

ofstream file\_output("output.txt");

file\_output.close();

ifstream file\_input("input.txt");

vector<Credit> Credits\_vector;

string line;

while (getline(file\_input, line)) {

istringstream ss(line);

Credit to\_vector;

ss >> to\_vector;

Credits\_vector.push\_back(to\_vector);

}

cout << "Vector before sorting: " << endl;

for (int i = 0; i < Credits\_vector.size(); ++i) {

cout << Credits\_vector[i];

}cout << endl;

sort(Credits\_vector.begin(), Credits\_vector.end());

cout << "Vector after sorting: " << endl;

for (int i = 0; i < Credits\_vector.size(); ++i) {

cout << Credits\_vector[i];

}cout << endl;

deque<Credit> Credits\_deque;

copy(Credits\_vector.begin(), Credits\_vector.end(), back\_inserter(Credits\_deque));

cout << "Deque copied from vector: " << endl;

for (int i = 0; i < Credits\_deque.size(); ++i) {

cout << Credits\_deque[i];

}cout << endl;

}

**Результат работы программы:**

Vector before sorting:

Oleg 5000 ruble 5%

Egor 3000 dollar 3%

Vlad 4000 euro 4%

Uzbek 2000 ruble 2%

Vector after sorting:

Uzbek 2000 ruble 2%

Egor 3000 dollar 3%

Vlad 4000 euro 4%

Oleg 5000 ruble 5%

Deque copied from vector:

Uzbek 2000 ruble 2%

Egor 3000 dollar 3%

Vlad 4000 euro 4%

Oleg 5000 ruble 5%

**Вывод:**

В ходе выполнения лабораторной работы были изучены принципы использования объектов своих классов в последовательных контейнерах библиотеки STL.