Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)

**Лабораторная работа № 1  
Использование пользовательских классов в контейнерах STL в языке Си++ (часть 2)**

Работу выполнил:

Конов Михаил, ИУ8-24

**ЦЕЛЬ**

Цель работы состоит в изучении использования пользовательских классов в контейнерах set и unordered\_set.

**УСЛОВИЕ ЗАДАЧИ**

Для класса, разработанного в ЛР4, обеспечить возможность добавления объектов в контейнер set (сортировка как указано в задании на ЛР4) и в контейнер unordered\_set. Исходные данные как в ЛР4 читать из файла, вывести на печать для контроля объекты контейнеров.

**ВАРИАНТ 28 (4)**

|  |  |  |
| --- | --- | --- |
| **Параметры приложений** | | **Исходный контейнер list, копируем в vector** |
| Объект- студент (поля: ФИО, группа, номер зачетной книжки, массив 4-х оценок за сессию) | Сортировка по ФИО | 10 |

**ТЕКСТ ПРОГРАММЫ**

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include <fstream>

#include <set>

#include <unordered\_set>

#include <sstream>

#include <algorithm>

using namespace std;

//Объект - студент (поля: ФИО, группа, номер зачетной книжки, массив 4-х оценок за сессию)

//Сортировка по ФИО

//Исходный контейнер - list, копируем в vector

class Student{

public:

string name;

string group;

string zachetka;

int session[4];

Student(){}

Student(string name, string group, string zachetka, int session[4]){

this->name = name;

this->group = group;

this->zachetka = zachetka;

for (int i = 0; i < 4; ++i){

this->session[i] = session[i];

}

}

//конструктор копирования

Student(const Student& S){

this->name = S.name;

this->group = S.group;

this->zachetka = S.zachetka;

for (int i = 0; i < 4; ++i){

this->session[i] = S.session[i];

}

}

//конструктор перемещения

Student(Student&& chel){

swap(this->name, chel.name);

swap(this->group, chel.group);

swap(this->zachetka, chel.zachetka);

for (int i = 0; i < 4; ++i){

swap(this->session[i], chel.session[i]);

}

}

print(){

cout << "\nФИО: " << name << "\nГруппа: " << group << "\nЗачетка: " << zachetka << endl;

for (auto pos: session){

cout << session[pos];

}

}

Student& operator=(const Student& s1) {

this->name = s1.name;

this->group = s1.group;

this->zachetka = s1.zachetka;

for (int i = 0; i < 4; ++i){

this->session[i] = s1.session[i];

}

return \*this;

}

//вспомогательные операторы

bool operator==(const Student& s1) const{

return this->name == s1.name;

}

bool operator <(const Student& s) const{

return (this->name < s.name);

}

friend ostream& operator << (ostream& osm, const Student& St){

osm << "\nФИО: " << St.name << "\nГруппа: " << St.group << "\nЗачетка: " << St.zachetka << endl;

for (int i = 0; i < 4; ++i){

osm << St.session[i];

}

osm << endl;

return osm;

}

};

struct hashworker {

size\_t operator()(const Student& s) const{

return hash<string>()(s.name);

}

};

int main(){

ifstream infile("input.txt");

fstream outfile("output.txt");

setlocale(LC\_ALL, "RUS");

//создаем list, set, unordered\_set

list <Student> origin;

set<Student> collection;

unordered\_set<Student, hashworker> un\_collection;

string name, group, zachetka;

int session[4];

//считываем из файла

while(!infile.eof()){

getline(infile, name);

infile >> group;

infile.ignore();

infile >> zachetka;

infile.ignore();

for (int i = 0; i < 4; ++i){

infile >> session[i];

cout << session[i];

}

cout << "\t";

infile.ignore();

origin.push\_back(Student(name, group, zachetka, session));

collection.insert(Student(name, group, zachetka, session));

un\_collection.insert(Student(name, group, zachetka, session));

}

//сортируем с помощью sort без параметров

cout << "Исходный список до сортировки:\n";

outfile << "Исходный список до сортировки:\n";

for (auto iter = origin.begin(); iter != origin.end(); iter++)

{

outfile << \*iter << "\t";

cout << \*iter << "\t";

}

vector <Student> legacy;

legacy.resize(origin.size());

copy(origin.begin(), origin.end(), legacy.begin());

cout << "Результат копирования в другой контейнер:\n";

outfile << "Результат копирования в другой контейнер:\n";

for (auto iter = legacy.begin(); iter != legacy.end(); iter++)

{

outfile << \*iter << "\t";

cout << \*iter << "\t";

}

cout << "Исходный список после сортировки:\n";

origin.sort();

outfile << "Исходный список после сортировки:\n";

for (auto iter = origin.begin(); iter != origin.end(); iter++)

{

outfile << \*iter << "\t";

cout << \*iter << "\t";

}

cout << "Содержимое set:\n";

outfile << "Содержимое set:\n";

for (const auto& x : collection) {

cout << x << endl;

outfile << x << endl;

}

cout << "Содержимое unordered set:\n";

outfile << "Содержимое unordered set:\n";

for (const auto& x : un\_collection) {

cout << x << endl;

outfile << x << endl;

}

return 0;

}

**ВЫВОД**

В данной лабораторной работе мы научились создавать контейнеры на основе пользовательских классов и управлять ими. Данные приемы помогают в упрощении разработки программ при решении задач на языке C++.