Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

за 4 семестр

По дисциплине: «ООТПиСП»

Тема: «Стандартная библиотека шаблонов»

Выполнил:

Студент 2 курса

Группы ПО-6(1)

Мартынович Д. М.

Проверил:

Булей Е. В.

2022

Лабораторная работа №5

Стандартная библиотека шаблонов

Цель работы: получить практические навыки работы со стандартной библиотекой шаблонов.

Вариант

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | multiset | queue | char |

**В программе № 1** выполнить следующее:

1. Создать объект-контейнер в соответствии с вариантом задания и заполнить его данными, тип которых определяется вариантом задания.

2. Просмотреть контейнер.

3. Изменить контейнер, удалив из него одни элементы и заменив другие.

4. Просмотреть контейнер, используя для доступа к его элементам итераторы.

5. Создать второй контейнер этого же класса и заполнить его данными того же типа, что и первый контейнер.

6. Изменить первый контейнер, удалив из него n элементов после заданного и добавив затем в него все элементы из второго контейнера.

7. Просмотреть первый и второй контейнеры.

#include <iostream>

#include <ctime>

#include <set>

using namespace std;

void print(multiset<char> obj,char name[]) {

//2 просмотр контейнера

cout <<name<<" printed with itterator are : ";

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

cout << \*it << " ";

cout << endl;

}

int main(){

srand(time(0));

//1

char name1[] = "Multiset1";

multiset<char> obj;

multiset<char>::iterator it;

for (size\_t i = 0; i < 5; i++)

{

char a = 'a' + rand() % 26;

obj.insert(a);

}

//2 Просмотр контейнера

print(obj,name1);

//3

//Добавление и удаление элементов в контейнер

it = obj.begin();

it++;//второй элемент

obj.erase(it);

print(obj,name1);

//5

multiset<char>obj2;

char name2[] = "Multiset2";

for (size\_t i = 0; i < 7; i++)

{

char a = 'a' + rand() % 26;

obj2.insert(a);

}

print(obj2, name2);

//6

it = obj.begin();

it++;

obj.erase(it,obj.end());

cout << name1 << " после удаления: " << endl;

print(obj, name1);

obj.insert(obj2.begin(), obj2.end());

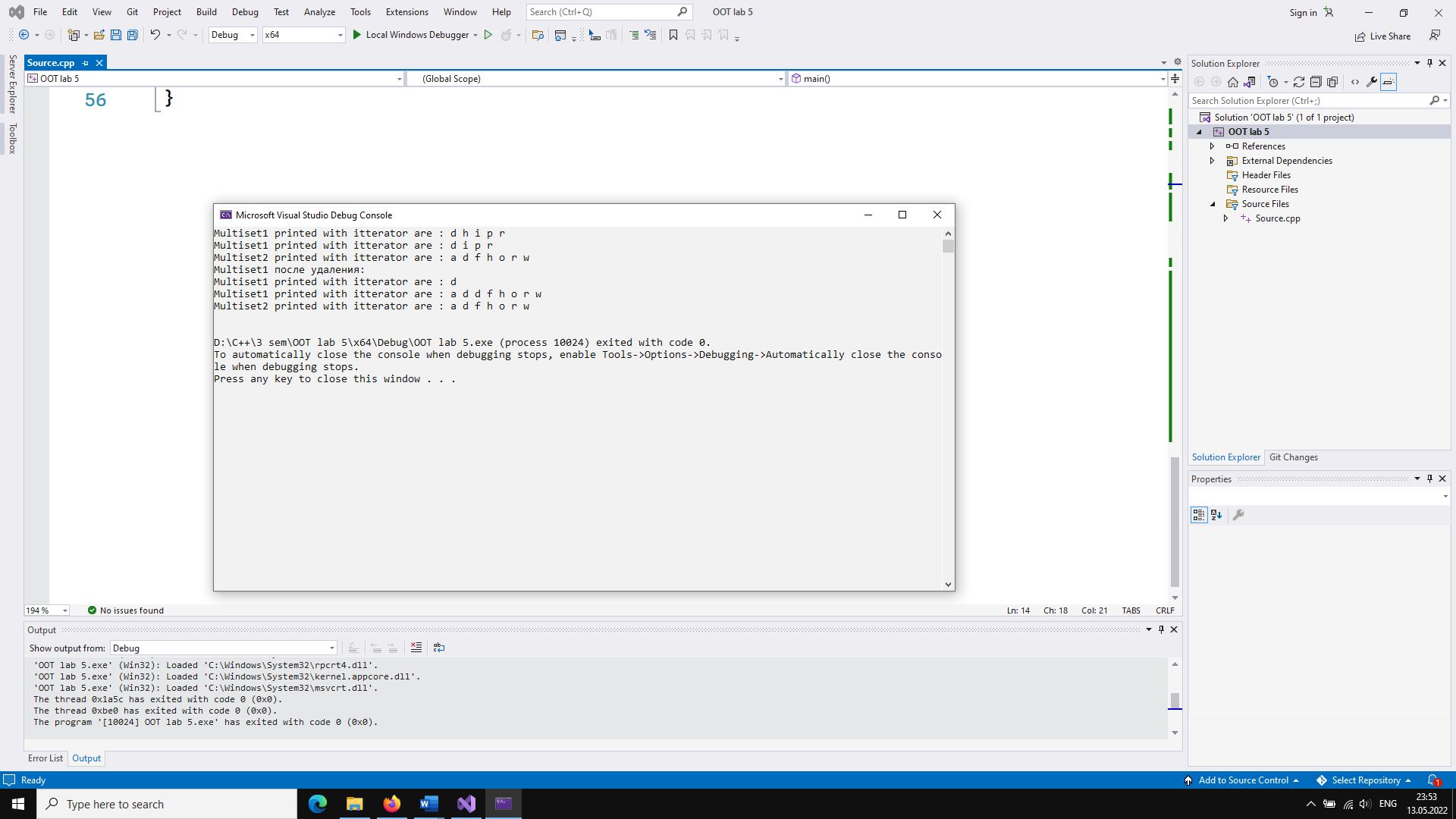
print(obj, name1);

print(obj2, name2);

cout << endl;

return 0;

}



**В программе № 2** выполнить то же самое, но для данных пользовательского типа.

#include <iostream>

#include <ctime>

#include <set>

#include <Windows.h>

using namespace std;

class Hero {

private:

int health;

int attack;

public:

Hero(int health = 0, int attack = 0) {

this->health = health;

this->attack = attack;

}

~Hero() {}

friend ostream& operator<<(ostream& out, const Hero& hero) {

out <<"Hero with health " << hero.health << ", attack " << hero.attack<<endl;

return out;

}

friend istream& operator>>(istream& in, Hero& hero) {

in >> hero.health;

in >> hero.attack;

return in;

}

bool operator!=(const Hero& hero) {

return (this->health != hero.health && this->attack != hero.attack);

}

bool operator< (const Hero& hero)const {

return this->health < hero.health;

}

};

void print(multiset<Hero> obj) {

//2 просмотр контейнераs

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

cout << \*it;

}

int main(){

system("color f0");

srand(time(0));

//1

multiset<Hero> obj;

multiset<Hero>::iterator it;

Hero captain(100, 20);

Hero hulk(350, 54);

Hero iron\_man(120, 15);

obj.insert(captain);

obj.insert(hulk);

obj.insert(iron\_man);

//2 Просмотр контейнера

cout << "Multiset 1" << endl;

print(obj);

//3

//Добавление и удаление элементов в контейнер

it = obj.begin();

obj.erase(it);

Hero dan(235, 40);

Hero dima(215, 135);

obj.insert(dan);

obj.insert(dima);

cout << endl << "After editing" << endl;

print(obj);

//5

multiset<Hero>obj2;

Hero a(80, 20);

Hero b(70, 65);

Hero c(50, 35);

obj2.insert(a);

obj2.insert(b);

obj2.insert(c);

cout << "Multiset 2" << endl;

print(obj2);

//6

it = obj.begin();

++it;

obj.erase(it, obj.end());

obj.insert(obj2.begin(), obj2.end());

cout << "Multiset 1 After editings" << endl;

print(obj);

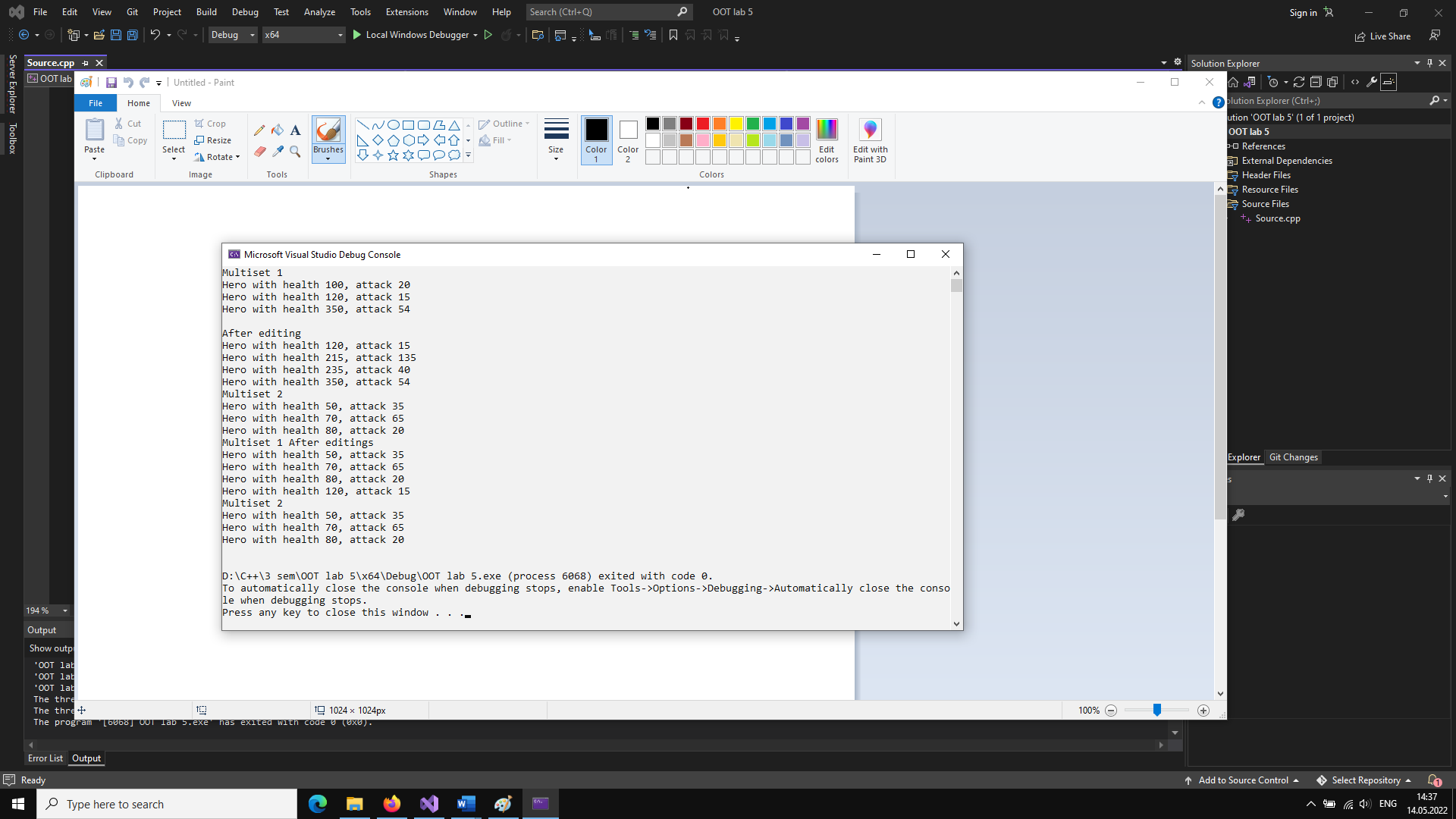
cout << "Multiset 2" << endl;

print(obj2);

cout << endl;

return 0;

}



**В программе № 3** выполнить следующее:

1. Создать контейнер, содержащий объекты пользовательского типа. Тип контейнера выбирается в соответствии с вариантом задания.

2. Отсортировать его по убыванию элементов.

3. Просмотреть контейнер.

4. Используя подходящий алгоритм, найти в контейнере элемент, удовлетворяющий заданному условию.

5. Переместить элементы, удовлетворяющие заданному условию в другой (предварительно пустой) контейнер. Тип второго контейнера определяется вариантом задания.

6. Просмотреть второй контейнер.

7. Отсортировать первый и второй контейнеры по возрастанию элементов.

8. Просмотреть их.

9. Получить третий контейнер путем слияния первых двух.

10. Просмотреть третий контейнер.

11. Подсчитать, сколько элементов, удовлетворяющих заданному условию, содержит третий контейнер.

12.Определить, есть ли в третьем контейнере элемент, удовлетворяющий заданному условию.

#include <iostream>

#include <ctime>

#include <set>

#include <Windows.h>

#include<algorithm>

using namespace std;

class Hero {

private:

int health;

int attack;

public:

int getHealth() {

return this->health;

}

int getAttack() {

return this->attack;

}

Hero(int health = 0, int attack = 0) {

this->health = health;

this->attack = attack;

}

~Hero() {}

friend ostream& operator<<(ostream& out, const Hero& hero) {

out <<"Hero with health " << hero.health << ", attack " << hero.attack<<endl;

return out;

}

friend istream& operator>>(istream& in, Hero& hero) {

in >> hero.health;

in >> hero.attack;

return in;

}

bool operator!=(const Hero& hero) {

return (this->health != hero.health && this->attack != hero.attack);

}

bool operator< (const Hero& hero)const {

return this->health < hero.health;

}

};

void print(multiset<Hero> obj) {

//2 просмотр контейнераs

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

cout << \*it;

}

int main(){

system("color f0");

srand(time(0));

//1

multiset<Hero> obj;

multiset<Hero>::iterator it;

Hero captain(100, 20);

Hero dan(80, 23);

Hero hulk(350, 54);

Hero iron\_man(120, 15);

obj.insert(captain);

obj.insert(dan);

obj.insert(hulk);

obj.insert(iron\_man);

//2 Просмотр контейнера

cout << "Multiset 1" << endl;

print(obj);

//3

//Добавление и удаление элементов в контейнер

multiset<Hero> obj2;

it = find\_if(obj.begin(), obj.end(), [](Hero hero) {return hero.getHealth() == 100; });//Ищем указатель в контейнере

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

std::cout << "Found " << (\*it) << '\n';

obj2.insert(\*it);

obj.erase(\*it);

}

else {

std::cout << "Not found\n";

}

it = find\_if(obj.begin(), obj.end(), [](Hero hero) {return hero.getHealth() == 120; });//Ищем указатель в контейнере

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

std::cout << "Found " << (\*it) << '\n';

obj2.insert(\*it);

obj.erase(\*it);

}

else {

std::cout << "Not found\n";

}

cout << "Multiset 1" << endl;

print(obj);

cout << "Multiset 2" << endl;

print(obj2);

multiset<Hero> obj3;

obj3.insert(obj.begin(), obj.end());

obj3.insert(obj2.begin(), obj2.end());

cout << "Multiset 3" << endl;

print(obj3);

cout << "Numbers of heroes health that bigger than 80: " << endl;

int countCond = count\_if(obj3.begin(), obj3.end(),

[](Hero hero) { return hero.getHealth() > 80;});

if (countCond != 0) {

cout << "Есть, кол-во: " << countCond << endl;

}

else {

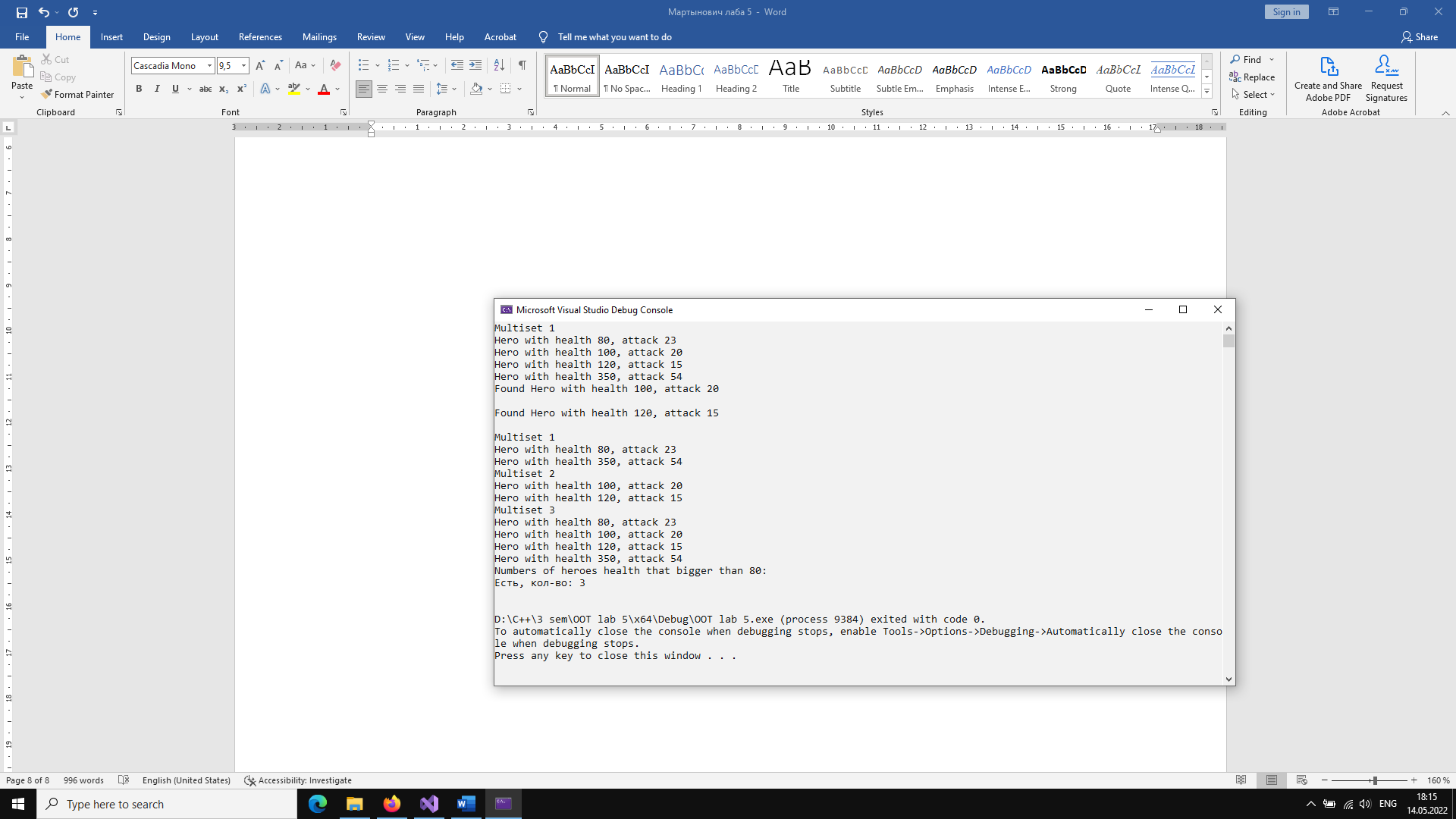
cout << "Нет" << endl;

}

cout << endl;

return 0;

}



Вывод: получил практические навыки работы со стандартной библиотекой шаблонов.