Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №4  
по курсу «Объектно-ориентированное программирование»  
на тему «Шаблоны в языке C++»

Выполнили студенты группы 22ВВП1:  
Беляев Д.

Захаров А.

Приняли:  
Евсеева Ю.И.  
Гудков А. А.

Пенза 2024

**Название**

Шаблоны в языке C++

**Цель работы**

Изучить шаблоны в языке C++

**Лабораторное задание**

Класс − множество set. Дополнительно перегрузить следующие

операции:

+ − добавить элемент в множество (типа set+item);

\* − пересечение множеств;

()− мощность множества.

**Метод решения задачи**

Разработали классы согласно заданию, перегрузили операции и выполнили задания, связанные с ними

**Листинг**

**main.cpp**

#include <iostream>

#include <set>

#include <algorithm>

#include <iterator>

template<typename T>

class OverloadSet

{

public:

OverloadSet() { }

OverloadSet(std::set<T> from) { selfSet = from; }

virtual ~OverloadSet() { };

void Insert(T value) { selfSet.insert(value); }

void Erase(T value) { selfSet.erase(value); }

auto Begin() { return selfSet.begin(); }

auto End() { return selfSet.end(); }

bool Contains(T value) { return selfSet.count(value); }

void printD()

{

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

std::cout << \*it << " ";

std::cout << std::endl;

}

// Overload

OverloadSet operator +(T value)

{

std::set<T> tmp;

std::copy(selfSet.begin(), selfSet.end(), std::inserter(tmp, tmp.begin()));

tmp.insert(value);

return OverloadSet(tmp);

}

OverloadSet operator -(T value)

{

std::set<T> tmp;

std::copy(selfSet.begin(), selfSet.end(), std::inserter(tmp, tmp.begin()));

tmp.erase(value);

return OverloadSet(tmp);

}

OverloadSet operator \*(OverloadSet<T> secondSet)

{

std::set<T> tmp;

std::set\_intersection(selfSet.begin(), selfSet.end(),

secondSet.Begin(), secondSet.End(),

std::inserter(tmp, tmp.begin()));

return OverloadSet(tmp);

}

int operator ()()

{

return selfSet.size();

}

private:

std::set<T> selfSet;

};

int main()

{

OverloadSet<int> set1;

set1.Insert(1);

set1.Insert(2);

set1.Insert(3);

set1.Insert(4);

OverloadSet<int> set2;

set2.Insert(4);

set2.Insert(2);

set2.Insert(6);

set2.Insert(7);

std::cout << "Set1: ";

set1.printD();

std::cout << "Set2: ";

set2.printD();

OverloadSet<int> set3 = set1 - 1;

std::cout << "Set3 = set1 - 1: ";

set3.printD();

OverloadSet<int> set4 = set2 + 10;

std::cout << "Set4 = set2 + 10: ";

set4.printD();

OverloadSet<int> set5 = set3 \* set4;

std::cout << "Set5 = set3 \* set4: ";

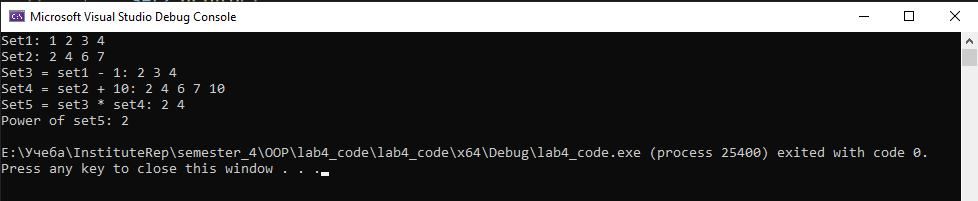
set5.printD();

std::cout << "Power of set: " << set5() << std::endl;

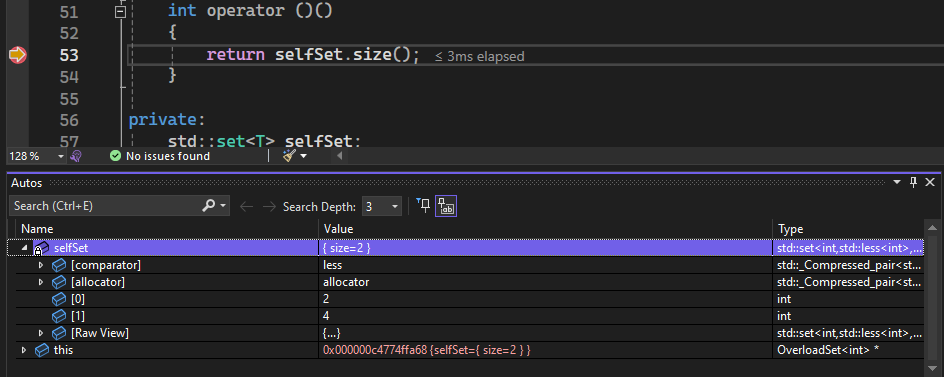
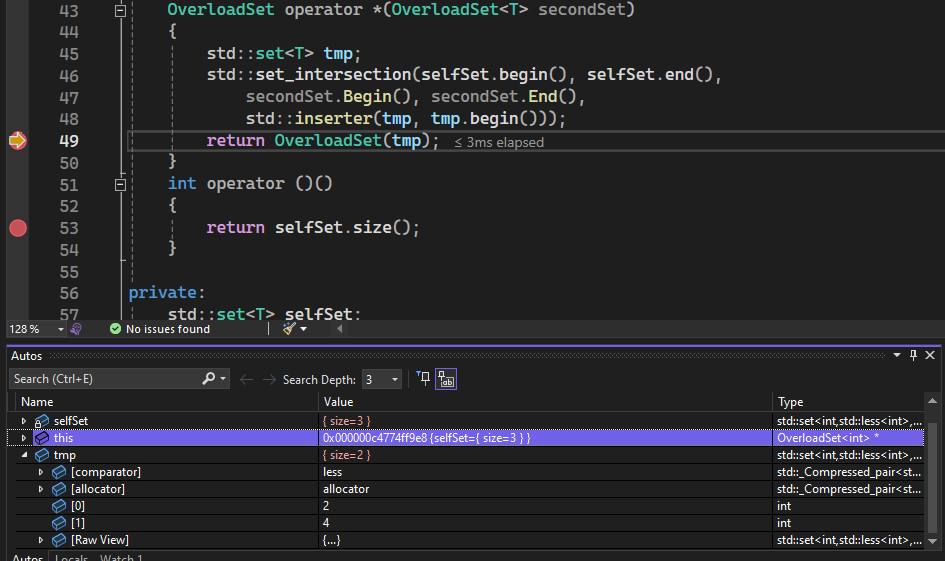
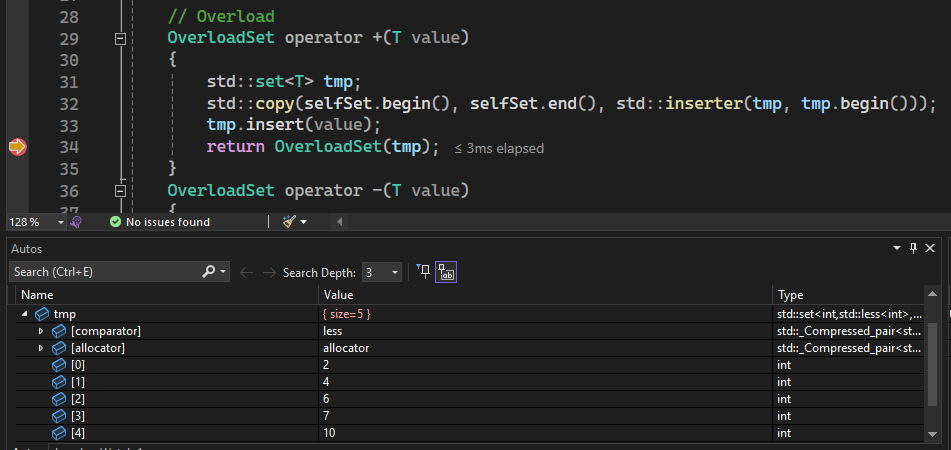
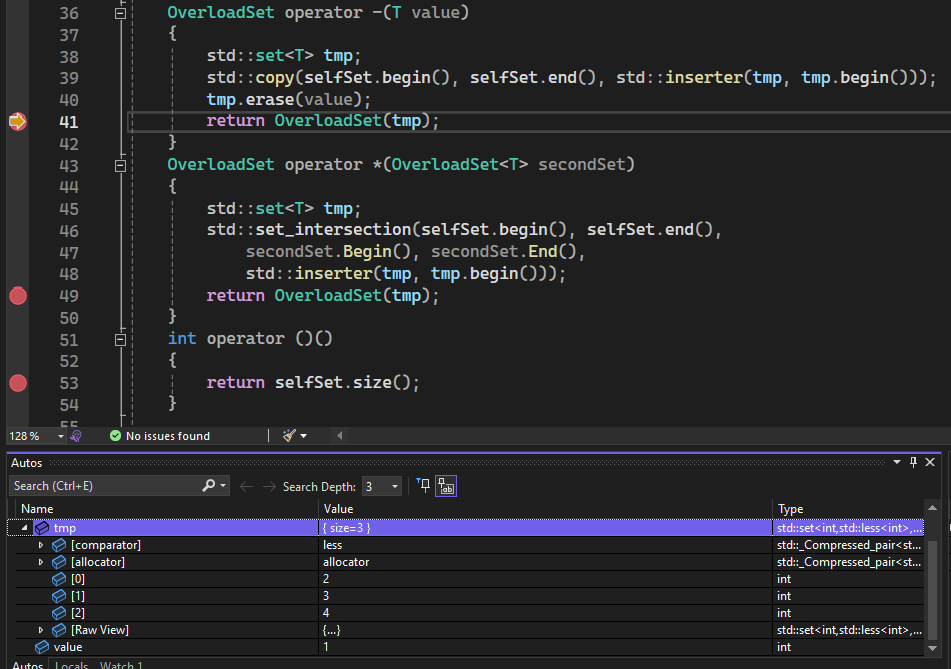
return 0;

}

**Результат работы программы**



**Протокол трассировки программы**



**Вывод**

Мы изучили механизмы наследования в языке C++