Министерство науки и высшего образования РФ

 Пензенский государственный университет Кафедра "Вычислительная техника"

Отчет по лабораторной работе №4

Работу выполнили студенты группы 23ВВВ3 бригады №9:

Полиневский Вадим

Кизым Иван

Проверили:

Кандидат т.н. Гудков А.А

Кандидат т.н. Евсеева И. Е

Пенза 2025

**Цель работы**: Изучить механизмы наследования в языке C++

**Задаие :** 9. Класс − множество set. Дополнительно перегрузить следующие

операции:

() − конструктор множества;

+ − объединение множеств;

<= − сравнение множеств.

**Описание практической части**:

Для выполнения программы было использовано два модуля: **lab3.cpp** с функцией main и файл заголовка **header.h.**

В файле заголовка прописано 4 класса: **Product**, **Toy**, **Produce**, **Dairy\_Produce**.

Список ирерхии для логичности выполняемой лабораторной работы:

Product -> Toy

Product -> Produce -> Dairy\_Produce.

Дочерние классы на каждом уровне имеют дополнительные поля и методы, помимио методов и полей родительского класса.

Листинг:

**lab3.cpp:**

#include <iostream>

#include <vector>

#include "header.h"

using namespace std;

int main (){

cout << "lab4" << endl;

vector <int> plenty\_1;

vector <int> plenty\_2;

vector <int> plenty\_3;

Enter\_Plenty(plenty\_1, 1);

Enter\_Plenty(plenty\_2, 2);

Enter\_Plenty(plenty\_3, 3);

Operation\_Planty(plenty\_1, plenty\_2, plenty\_3);

}

**header.h:**

#pragma once

#include <string>

#include <algorithm>

#include <unordered\_set>

using namespace std;

template <typename Type>

class Set{

private:

vector <Type> plenty;

void Remove\_Duplicates(vector <Type>& this\_vector) {

unordered\_set <Type> s;

vector <Type> result;

for (Type value : this\_vector){

if(s.find(value) == s.end()){

s.insert(value);

result.push\_back(value);

}

}

this\_vector.clear();

this\_vector = result;

}

public:

void Print\_Vector(string Out\_Info) {

cout << Out\_Info<< endl;

for (int num : plenty) {

cout << num << "\t";

}

cout << endl;

}

void operator () (const vector <Type>& temp\_vector) {

this->plenty = temp\_vector;

}

void operator + (const Set& \_Set) {

this->plenty.insert(plenty.end(), \_Set.plenty.begin(), \_Set.plenty.end());

this->Remove\_Duplicates(this->plenty);

}

bool operator <= (const Set& \_Set) {

if (this->plenty.size() > \_Set.plenty.size()) return false;

else return true;

}

};

void Print\_Plenty (Set<int> set\_1, Set<int> set\_2, Set<int> set\_3) {

set\_1.Print\_Vector("SET1 :");

set\_2.Print\_Vector("SET2 :");

set\_3.Print\_Vector("SET3 :");

}

void Enter\_Plenty(vector <int>& temp\_vector, int number) {

const int count = 5;

int temp\_value;

cout << " ENTER " << number << " PLANTY:" << endl;

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

cout << i << ": ";

cin >> temp\_value;

temp\_vector.push\_back(temp\_value);

}

}

void Operation\_Planty(vector <int> temp\_1, vector<int> temp\_2, vector <int> temp\_3){

Set <int> Set\_1, Set\_2, Set\_3;

Set\_1(temp\_1);

Set\_2(temp\_2);

Set\_3(temp\_3);

Print\_Plenty(Set\_1, Set\_2, Set\_3);

cout << "\n" << endl;

Set\_1 + Set\_2;

if (Set\_1 <= Set\_2) cout << "Set\_1 <= Set\_2"<<endl;

else cout << "Set\_1 > Set\_2";

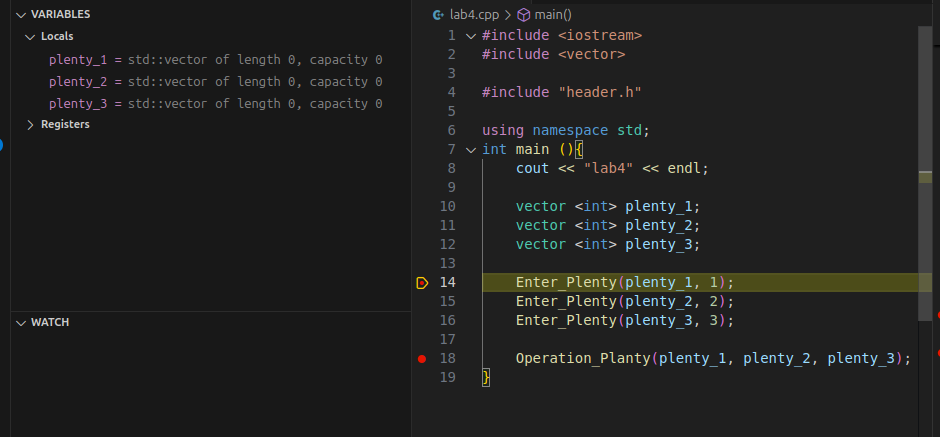
cout << "\n" << endl;

Print\_Plenty(Set\_1, Set\_2, Set\_3);

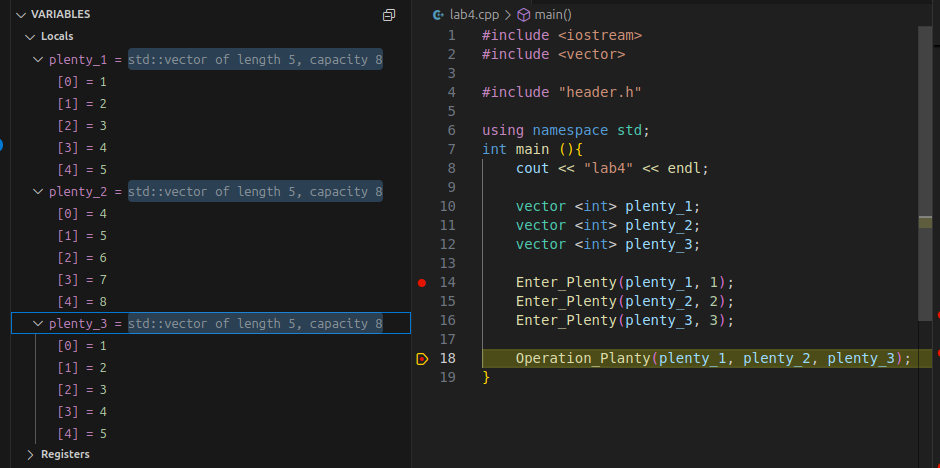
}

**Отладка:**

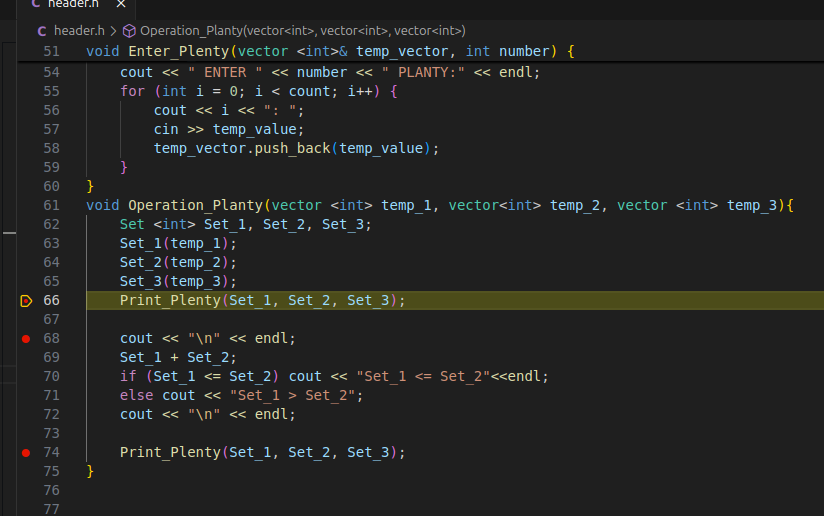
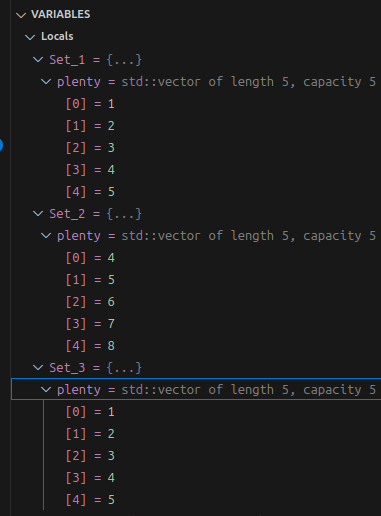
Первая точка остановки: векторы с переменными для множеств до заполнения



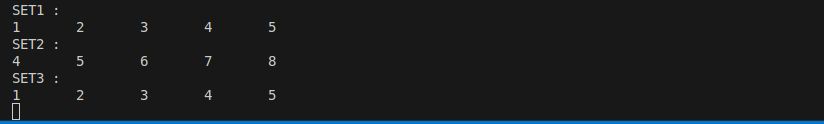
Вторая точка остановки: векторы с переменными для множеств после заполнения:



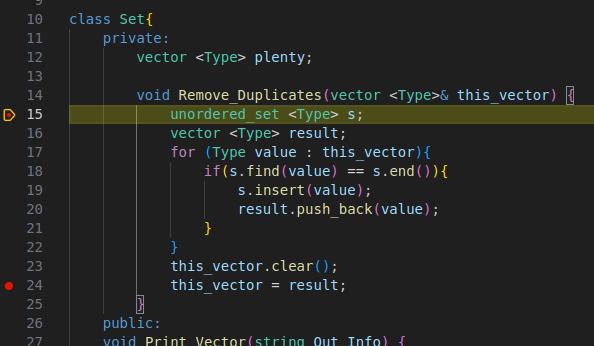
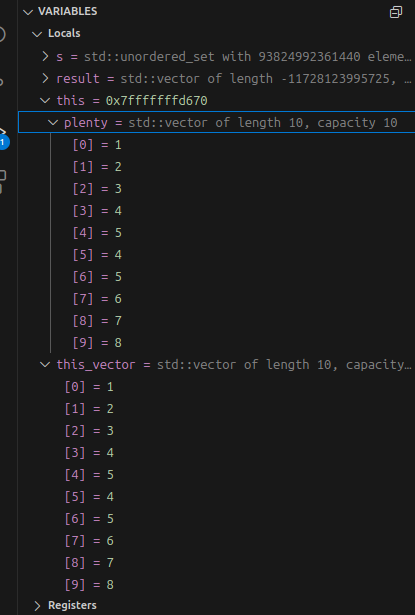
Третья точка остановки: Объекты типа вектор после присвоения полю plenty значений векторов:

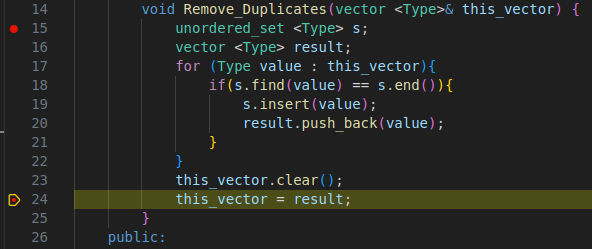
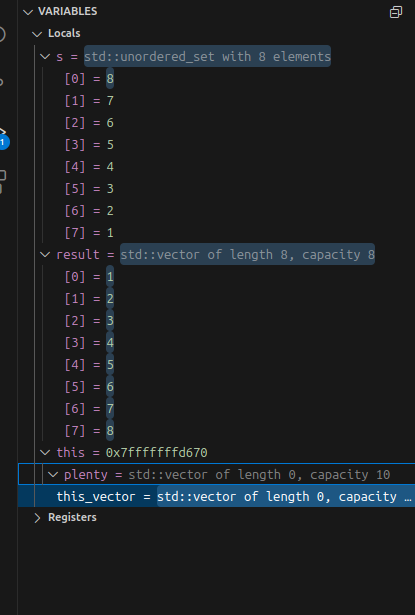


Четвертая точка остановки. Вывод множеств из полей plenty объектов Set\_1, Set\_2, Set\_3:

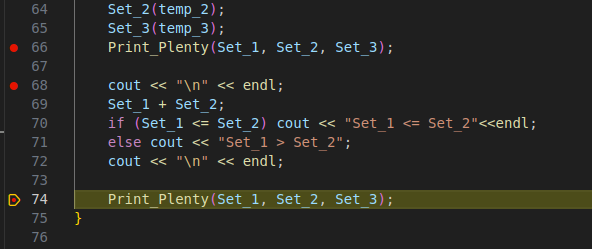
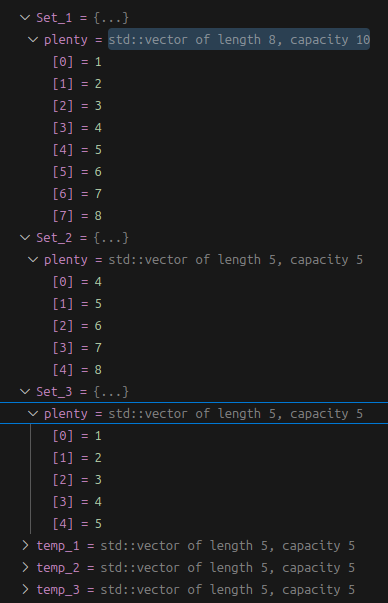


Пятая, 6 точки остановки: шаги алгаритма по объединению множеств (удаление дубликатов):





Седьмая точка остановки: значения полей класса перед вторым выводом, результат сравнения:



**Terminal-Log:**

lab4

ENTER 1 PLANTY:

0: 1

1: 2

2: 3

3: 4

4: 5

ENTER 2 PLANTY:

0: 4

1: 5

2: 6

3: 7

4: 8

ENTER 3 PLANTY:

0: 1

1: 2

2: 3

3: 4

4: 5

SET1 :

1 2 3 4 5

SET2 :

4 5 6 7 8

SET3 :

1 2 3 4 5

Set\_1 > Set\_2

SET1 :

1 2 3 4 5 6 7 8

SET2 :

4 5 6 7 8

SET3 :

1 2 3 4 5

[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-tdxbc2yn.mgc" 1>"/tmp/Microsoft-MIEngine-Out-jo3egdek.m1s"