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 шаблоны в языке C++.**

1. Получить индивидуальное задание у преподавателя.
2. Написать программу, выполняющую действия, согласно заданию.
3. Данные в программе должны вводиться с клавиатуры
4. Программу требуется отладить и протестировать на различных наборах данных
5. Продемонстрировать работу программы на экране.
6. Сдать теоретическую часть по отчету преподавателю.

**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: **lab4.cpp** с функцией main и файл заголовка **header.h.**

**lab4.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\_Plenty(plenty\_1, plenty\_2, plenty\_3);

}

**header.h:**

#pragma once

#include <string>

using namespace std;

template <typename Type>

class Set{

private:

Type \*plenty;

unsigned int Size\_Plenty;

public:

Set() : plenty(nullptr), Size\_Plenty(0) {}

Set(const Set& other) {

Size\_Plenty = other.Size\_Plenty;

if (Size\_Plenty > 0) {

plenty = new Type[Size\_Plenty];

for (unsigned int i = 0; i < Size\_Plenty; i++) {

plenty[i] = other.plenty[i];

}

} else {

plenty = nullptr;

}

}

Set& operator=(const Set& other) {

if (this != &other) {

delete[] plenty;

Size\_Plenty = other.Size\_Plenty;

if (Size\_Plenty > 0) {

plenty = new Type[Size\_Plenty];

for (unsigned int i = 0; i < Size\_Plenty; i++) {

plenty[i] = other.plenty[i];

}

} else {

plenty = nullptr;

}

}

return \*this;

}

unsigned int get\_size\_plenty() {

return Size\_Plenty;

}

Type\* get\_Plenty\_elements() {

return plenty;

}

void operator + (const Set& \_Set) {

Set<Type> result;

for (unsigned i = 0; i < Size\_Plenty; ++i) {

bool found = false;

for (unsigned j = 0; j < result.Size\_Plenty; ++j) {

if (plenty[i] == result.plenty[j]) {

found = true;

break;

}

}

if (!found) {

Type\* temp = new Type[result.Size\_Plenty + 1];

for (unsigned k = 0; k < result.Size\_Plenty; ++k) {

temp[k] = result.plenty[k];

}

temp[result.Size\_Plenty] = plenty[i];

delete[] result.plenty;

result.plenty = temp;

result.Size\_Plenty++;

}

}

for (unsigned i = 0; i < \_Set.Size\_Plenty; ++i) {

bool found = false;

for (unsigned j = 0; j < result.Size\_Plenty; ++j) {

if (\_Set.plenty[i] == result.plenty[j]) {

found = true;

break;

}

}

if (!found) {

Type\* temp = new Type[result.Size\_Plenty + 1];

for (unsigned k = 0; k < result.Size\_Plenty; ++k) {

temp[k] = result.plenty[k];

}

temp[result.Size\_Plenty] = \_Set.plenty[i];

delete[] result.plenty;

result.plenty = temp;

result.Size\_Plenty++;

}

}

delete[] plenty;

plenty = result.plenty;

Size\_Plenty = result.Size\_Plenty;

result.plenty = nullptr;

result.Size\_Plenty = 0;

}

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

this->plenty = new Type[temp\_vector.size()];

Size\_Plenty = temp\_vector.size();

for (int i = 0; i < Size\_Plenty; i++) {

plenty[i] = temp\_vector[i];

}

}

~Set () {

delete[] plenty;

plenty = nullptr;

}

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

if (this->Size\_Plenty == \_Set.Size\_Plenty) {

for (int i = 0; i < Size\_Plenty; i++) {

if (this->plenty[i] != \_Set.plenty[i]) {return -1;}

}

return 0;

}

else {return -1;}

}

};

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 Print\_Plenty(Set <int> \_Set , string Name\_Set){

int \*temp\_plenty;

unsigned int temp\_Size;

temp\_plenty = \_Set.get\_Plenty\_elements();

temp\_Size = \_Set.get\_size\_plenty();

cout << Name\_Set<< ": ";

for (int i = 0; i < temp\_Size; i++) {

cout << temp\_plenty[i] << ", ";

}

cout << endl;

}

void Operation\_Plenty(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\_1 operator () ");

Print\_Plenty(Set\_2, "Set\_2 operator () ");

Print\_Plenty(Set\_3, "Set\_3 operator () ");

Set\_1 + Set\_2;

Print\_Plenty (Set\_1, "Set\_1 operator (+) Set\_2");

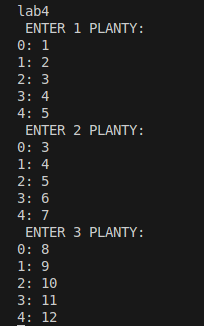
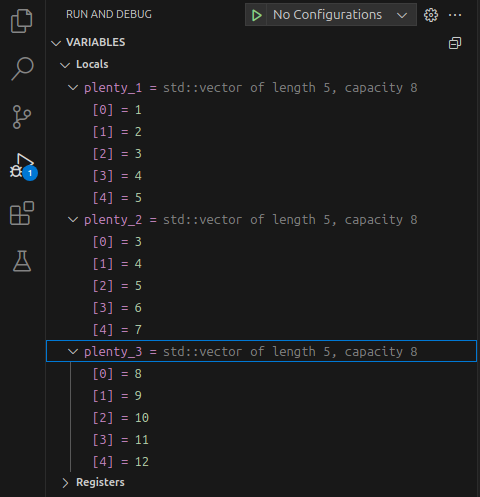
if (Set\_1 <= Set\_2 == 0) {cout << "sets are equal"<< endl;}

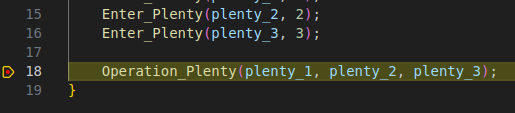
else {cout << "sets are not equal"<< endl;}

}

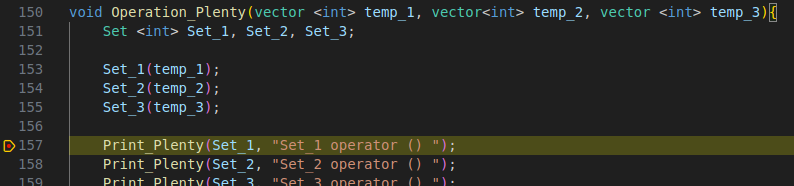
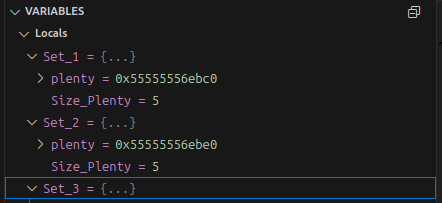
**Отладка:**

Первая точка остановки: векторы с переменными для множеств после заполнения вводными данными:





Вторая точка остановки: инициализация множеств.



Третья точка остановки: выполнение операций над множествами:

