**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №5

«Наследование. Множества»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-22Б |  | Преподаватель каф. ИУ5 |
| Поляков Л. С. |  | Бурмистрова М.В. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

# Постановка задачи

Разработать **шаблонный класс** множество (MySet) на базе **шаблонного класса** вектор (MyVector) для выполнения операций над множествами (+, -, \*, +=, -=, \*=, ==) и функцию main() для его тестирования.

Необходимо реализовать **специализацию шаблонов** для работы со скалярными типами (char\*, double, int и др.). Это необходимо сделать в виде **специализации конкретных методов шаблонов**, которые зависят от параметров шаблона. Иными словами, вместо реализации специализации шаблона полностью, необходимо специализировать конкретные методы шаблона, функционирование которых зависит от типов аргументов (например, деструкторы, сортировка и т.д.).

* Класс вектор должен быть динамическим массивом, размер которого может автоматически изменяться (увеличиваться или уменьшаться) в процессе выполнения программы. Добавление элементов производится в конец вектора.
* Для ускорения выполнения операций над множествами вектор, используемый классом множество, должен быть отсортирован (сортировку достаточно делать только при добавлении элемента в множество). Для поиска элементов множества следует использовать метод половинного деления.
* Методы add\_element() и delete\_element() производного класса MySet переопределяют одноименные методы базового класса MyVector, а остальные элементы класса MyVector наследуются классом MySet.

При выполнении лабораторной работы использовать файлы с описанием классов MyVector, MySet и функцию main() главного проекта, приведенные в [Приложении 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab5/Instructions/SupplementMaterial1/).

**Требования:**

* Класс MySet и MyVector должны быть упакованы в отдельную статическую библиотеку. Пример того, как это может быть сделано, представлен в [Приложении 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab5/Instructions/SupplementMaterial1/).
* Обратите внимание, что заготовка **не содержит шаблонов классов** и работает только со строками (с элементами char\*). **Необходимо самостоятельно** шаблонизировать описание классов MyVector и MySet и реализовать различные специализации классов.

# Разработка алгоритма

## Структура проекта

.

├── CMakeLists.txt

├── MyContainers

│   ├── CMakeLists.txt

│   ├── MySet.cpp

│   ├── MySet.hpp

│   ├── MyVector.cpp

│   └── MyVector.hpp

└── main.cpp

## Текст программы

**MyVector.hpp**

#ifndef LAB\_5\_2\_MYVECTOR\_HPP  
#define LAB\_5\_2\_MYVECTOR\_HPP  
  
  
#include <iostream>  
#include <iomanip>  
#include <cstring>  
  
const int MAX\_SIZE = 5;  
  
template <class INF>  
  
class MyVector  
{  
 typedef class MyVector<INF> Vector;  
protected:  
 INF \*inf;  
 int maxSize;  
 int currentSize;  
 void resize();  
public:  
 MyVector() : inf(nullptr), maxSize(MAX\_SIZE), currentSize(0) {};  
 MyVector(const MyVector&);  
 ~MyVector() { delete[] inf; }  
  
 virtual void addElement(INF);  
 void deleteElement(int);  
 INF& operator [] (int);  
 void sort();  
 bool isElement(INF);  
 int getSize() { return currentSize; }  
 int getMaxSize() { return maxSize; }  
 int find(INF);  
 MyVector<INF>& operator = (const MyVector<INF>&);  
 template<class U>  
 friend std::ostream& operator <<(std::ostream&, const MyVector<U>&);  
};  
  
template<class INF>  
inline void MyVector<INF>::resize() {  
 bool flag{};  
 if (currentSize > maxSize) {  
 maxSize \*= 2;  
 flag = true;  
 }  
 else if (maxSize / 4 > currentSize) {  
 maxSize /= 2;  
 flag = true;  
 }  
 if (flag) {  
 auto temp = new INF[maxSize];  
 for (int i{}; i < currentSize - 1; ++i)  
 temp[i] = this->inf[i];  
 delete[] inf;  
 inf = temp;  
 }  
  
}  
  
template<class INF>  
inline MyVector<INF>::MyVector(const MyVector& other) {  
 maxSize = MAX\_SIZE;  
 currentSize = other.currentSize;  
 inf = new INF[currentSize];  
 for (int i{}; i < other.currentSize; ++i) {  
 inf[i] = other.inf[i];  
 }  
}  
  
template<class INF>  
inline void MyVector<INF>::addElement(INF elem) {  
 currentSize += 1;  
 resize();  
 inf[currentSize - 1] = elem;  
}  
  
template<class INF>  
inline void MyVector<INF>::deleteElement(int idx) {  
 if (idx < 0 or idx > currentSize)  
 return;  
 for (int i = idx; i < currentSize - 1; ++i)  
 inf[i] = inf[i + 1];  
 resize();  
 currentSize--;  
}  
  
template<class INF>  
inline INF& MyVector<INF>::operator[](int idx) {  
 return \*(inf + idx);  
}  
  
template<class INF>  
inline void MyVector<INF>::sort() {  
 for (int i{}; i < currentSize - 1; ++i) {  
 for (int j = i + 1; j < currentSize; ++j) {  
 if (inf[i] > inf[j]) {  
 std::swap(inf[i], inf[j]);  
 }  
 }  
 }  
}  
  
template<>  
inline void MyVector<char \*>::sort() {  
 for (int i{}; i < currentSize -1; i++) {  
 for (int j = i + 1; j < currentSize; ++j) {  
 if (strcmp(inf[i], inf[j]) > 0) {  
 std::swap(inf[i], inf[j]);  
 }  
 }  
 }  
}  
  
template<class INF>  
inline int MyVector<INF>::find(INF elem) {  
 for (int i{}; i < currentSize; ++i)  
 if (inf[i] == elem)  
 return i + 1;  
 return -1;  
}  
  
template<>  
inline int MyVector<char\*>::find(char\* elem) {  
 for (int i{}; i < currentSize; ++i) {  
 if (strncmp(inf[i], elem, sizeof(\*elem)) == 0) {  
 return i + 1;  
 }  
 }  
 return -1;  
}  
  
template<class INF>  
inline MyVector<INF>& MyVector<INF>::operator=(const MyVector<INF>& source) {  
 if (this != &source) {  
 this->currentSize = 0;  
 delete[] inf;  
 inf = new INF[currentSize];  
 for (int i{}; i < source.currentSize; ++i) {  
 this->addElement(source.inf[i]);  
 }  
 }  
 return \*this;  
}  
  
template<class U>  
inline std::ostream& operator<<(std::ostream& out, const MyVector<U>& elem) {  
 if (elem.currentSize == 0) {  
 out << "[EMPTY]";  
 return out;  
 }  
 out << '{';  
 for (int i{}; i < elem.currentSize - 1; ++i) {  
 out << elem.inf[i] << ", ";  
 }  
 out << elem.inf[elem.currentSize - 1];  
 out << '}';  
 return out;  
}  
  
template<class INF>  
inline bool MyVector<INF>::isElement(INF elem) {  
 if (this->find(elem) != -1)  
 return true;  
 return false;  
}  
  
#endif //LAB\_5\_2\_MYVECTOR\_HPP

**MySet.hpp**

#ifndef LAB\_5\_2\_MYSET\_HPP  
#define LAB\_5\_2\_MYSET\_HPP  
  
#include "MyVector.hpp"  
  
template <class INF>  
class MySet: public MyVector<INF> {  
public:  
 explicit MySet(INF inf = NULL) : MyVector<INF>(){};  
 bool operator == (MySet&);  
 MySet& operator += (MySet&);  
 MySet& operator -= (MySet&);  
 MySet& operator \*= (MySet&);  
  
 void addElement(INF);  
 void delete\_Element(INF);  
  
 template <class U>  
 friend MySet<U> operator+(MySet<U>&, MySet<U>&);  
 template <class U>  
 friend MySet<U> operator-(MySet<U>&, const MySet<U>&);  
 template <class U>  
 friend MySet<U> operator\*(MySet<U>&, MySet<U>&);  
};  
  
template<class INF>  
inline bool MySet<INF>::operator==(MySet& other) {  
 if (this->currentSize != other.currentSize) {  
 return false;  
 }  
 for (int i{}; i < this->currentSize - 1; ++i) {  
 if (this->find(other.inf[i]) == -1) {  
 return false;  
 }  
 }  
 return true;  
}  
  
template<class U>  
inline MySet<U> operator+(MySet<U>& lhs, MySet<U>& rhs) {  
 MySet<U> result;  
 for (int i{}; i < lhs.currentSize; ++i) {  
 result.addElement(lhs.inf[i]);  
 }  
 for (int j{}; j < rhs.currentSize; ++j) {  
 if (result.find(rhs.inf[j]) == -1) {  
 result.addElement(rhs.inf[j]);  
 }  
 }  
 return result;  
}  
  
template<class INF>  
inline MySet<INF>& MySet<INF>::operator+=(MySet<INF>& source) {  
 \*this = \*this + source;  
 return \*this;  
}  
  
template<class U>  
inline MySet<U> operator-(MySet<U>& lhs, const MySet<U>& rhs) {  
 MySet<U> result = lhs;  
 for (int i{}; i < rhs.currentSize; ++i) {  
 if (result.isElement(rhs.inf[i])) {  
 result.deleteElement(result.find(rhs.inf[i]) - 1);  
 }  
 }  
 return result;  
}  
  
template<class INF>  
inline MySet<INF>& MySet<INF>::operator-=(MySet<INF>& source) {  
 \*this = \*this - source;  
 return \*this;  
}  
  
template<class U>  
inline MySet<U> operator\*(MySet<U>& lhs, MySet<U>& rhs) {  
 MySet<U> result;  
 for (int i{}; i < lhs.currentSize; ++i) {  
 if (lhs.find(lhs.inf[i]) != -1 && rhs.find(lhs.inf[i]) != -1) {  
 result.addElement(lhs.inf[i]);  
 }  
 }  
 return result;  
}  
  
template<class INF>  
inline MySet<INF>& MySet<INF>::operator\*=(MySet<INF>& source) {  
 \*this = \*this \* source;  
 return \*this;  
}  
  
template<class INF>  
inline void MySet<INF>::addElement(INF elem) {  
 if (!this->isElement(elem)) {  
 MyVector<INF>::addElement(elem);  
 }  
}  
  
template<class INF>  
inline void MySet<INF>::delete\_Element(INF elem) {  
 if (this->isElement(elem)) {  
 MyVector<INF>::deleteElement(this->find(elem));  
 }  
}  
  
#endif //LAB\_5\_2\_MYSET\_HPP

**MyVector.cpp**

#include "MyVector.hpp"

**MySet.cpp**

#include "MySet.hpp"

**MyContainers/CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME MyContainers)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
 MyVector.hpp  
 MySet.hpp  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 MyVector.cpp  
 MySet.cpp  
)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_library(${PROJECT\_NAME} STATIC ${${PROJECT\_NAME}\_SOURCE\_LIST})

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME Lab\_5)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 main.cpp  
)  
  
add\_subdirectory(MyContainers)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_executable(${PROJECT\_NAME} ${${PROJECT\_NAME}\_SOURCE\_LIST})  
  
target\_link\_libraries(${PROJECT\_NAME} MyContainers)

**main.cpp**

#include <iostream>  
#include "MyContainers/MySet.hpp"  
  
  
int main() {  
 setlocale(LC\_ALL, "russian");  
 MyVector<int> testVec;  
 MySet<int> testSet;  
 MySet<int> testSet1, testSet2;  
 testSet1.addElement(1);  
 testSet1.addElement(2);  
 testSet1.addElement(3);  
 testSet2.addElement(4);  
 testSet2.addElement(5);  
 testSet2.addElement(6);  
  
 testSet = testSet1 + testSet2;  
 std::cout << testSet << std::endl;  
  
 testVec.addElement(54);  
 testVec.addElement(54);  
 testVec.deleteElement(14);  
 testSet.addElement(54);  
 testSet.addElement(54);  
 testSet.delete\_Element(54);  
 std::cout << testVec << ' ' << testSet << std::endl;  
  
 MySet<int> t1, t2, result;  
 t1.addElement(12);  
 t1.addElement(123);  
 t1.addElement(23);  
 t1.addElement(73);  
  
 t2.addElement(12);  
 t2.addElement(27);  
 t2.addElement(1337);  
 t2.addElement(73);  
  
 MyVector<char\*> v;  
 v.addElement(const\_cast<char \*>("Привет!"));  
 v.addElement(const\_cast<char \*>("Привет!"));  
 v.addElement(const\_cast<char \*>("Привет!"));  
 v.addElement(const\_cast<char \*>("Привет!"));  
 v.addElement(const\_cast<char \*>("Привет!"));  
 std::cout << "Вектор v: " << v << std::endl;  
 v.addElement(const\_cast<char \*>("Привет!"));  
 v.addElement(const\_cast<char \*>("Привет!"));  
 v.addElement(const\_cast<char \*>("Привет!"));  
 std::cout << "Вектор v: " << v << std::endl;  
 MyVector<char\*> v1 = v;  
 std::cout << "Вектор v1: " << v1 << std::endl;  
 for (int i = 0; i < MAX\_SIZE; i++)  
 v1.deleteElement(0);  
 std::cout << "Вектор v1: " << v1 << std::endl;  
 MySet <char\*> s, s1, s2;  
 s.addElement(const\_cast<char \*>("Привет!"));  
 s.addElement(const\_cast<char \*>("No"));  
 char\* str = const\_cast<char \*>("Hello!");  
 s.addElement(str);  
 std::cout << "Множество s: " << s << std::endl;  
 s1.addElement(const\_cast<char \*>("Cat"));  
 s1.addElement(const\_cast<char \*>("No"));  
 s1.addElement(const\_cast<char \*>("Привет!"));  
 std::cout << "Множество s1: " << s1 << std::endl;  
 s2 = s1 - s;  
 std::cout << "Множество s2=s1-s: " << s2 << std::endl;  
 std::cout << "Множество s1: " << s1 << std::endl;  
 std::cout << "Множество s: " << s << std::endl;  
 s2 = s - s1;  
 std::cout << "Множество s2=s-s1: " << s2 << std::endl;  
 std::cout << "Множество s1: " << s1 << std::endl;  
 std::cout << "Множество s: " << s << std::endl;  
 s2 = s1 + s;  
 std::cout << "Множество s2=s1+s: " << s2 << std::endl;  
 std::cout << "Множество s1: " << s1 << std::endl;  
 std::cout << "Множество s: " << s << std::endl;  
 s2 = s1 \* s;  
 std::cout << "Множество s2=s1\*s: " << s2 << std::endl;  
 std::cout << "Множество s1: " << s1 << std::endl;  
 std::cout << "Множество s: " << s << std::endl;  
 MySet<char\*> s3 = s2;  
 std::cout << "Множество s3=s2: " << s3 << std::endl;  
 if (s3 == s2)  
 std::cout << "Множество s3=s2\n";  
 else  
 std::cout << "Множество s3!=s2\n";  
 if (s3 == s1)  
 std::cout << "Множество s3=s1\n";  
 else  
 std::cout << "Множество s3!=s1\n";  
 if (s1 == s3)  
 std::cout << "Множество s1=s3\n";  
 else  
 std::cout << "Множество s1!=s3\n";  
 return 0;  
}

# Анализ Результатов

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

# Вывод

Я научился

* Работать с шаблонами классов в C++
* Использовать специализацию шаблонов
* Использовать механизм наследования классов в C++
* Работать со статическими библиотеками с использованием Cmake

Познакомился с внутренним устройством таких структур данных как динамический массив и сет, понял принцип их работы и научился реализовывать их в коде.