Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

# Лабораторная работа " Ассоциативные контейнеры библиотеки STL."

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь, 2024

Вариант 8.

Постановка задачи:

Общая:

Задача 1.

1. Создать ассоциативный контейнер.

2. Заполнить его элементами стандартного типа (тип указан в варианте).

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

5. Выполнить задание варианта для полученного контейнера.

6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 2.

1. Создать ассоциативный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

5. Выполнить задание варианта для полученного контейнера.

6. Выполнение всех заданий оформить в виде глобальных функций.

Задача 3

1. Создать параметризированный класс, используя в качестве контейнера ассоциативный контейнер.

2. Заполнить его элементами.

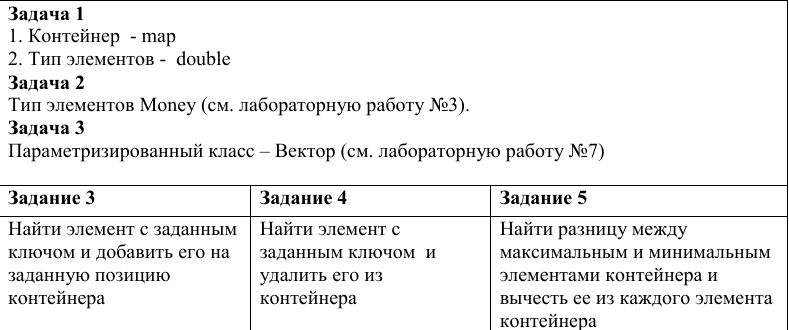
3. Добавить элементы в соответствии с заданием

4. Удалить элементы в соответствии с заданием.

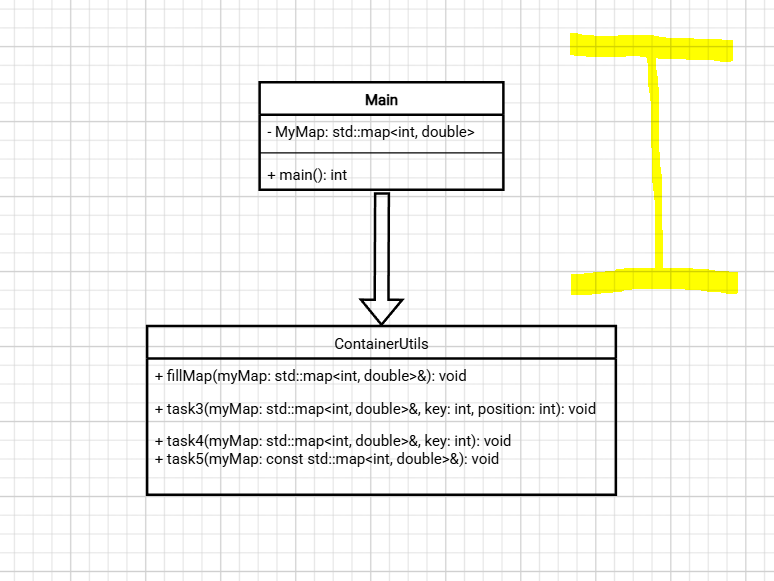
5. Выполнить задание варианта для полученного контейнера.

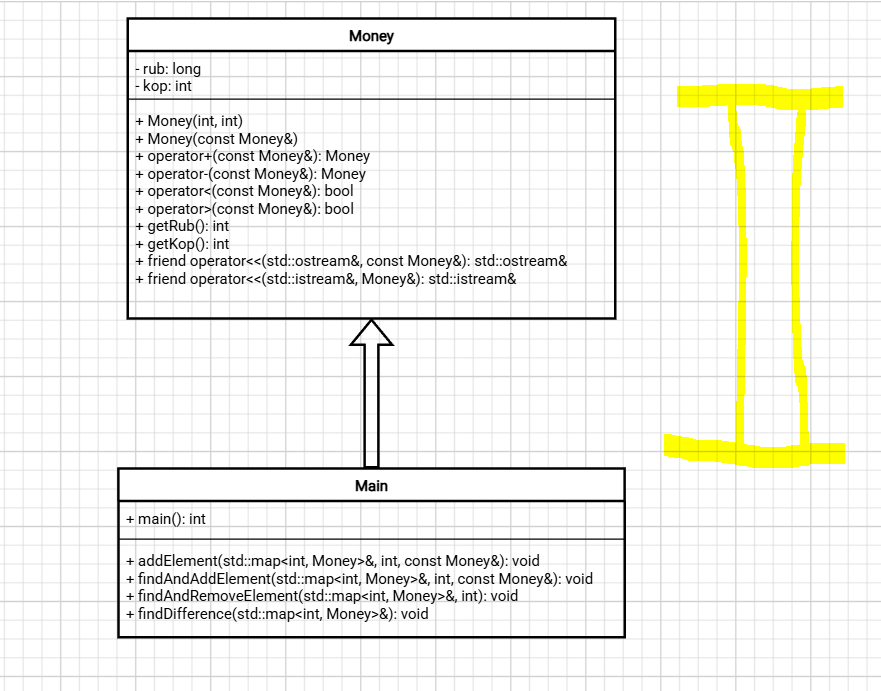
6. Выполнение всех заданий оформить в виде методов параметризированного класса.

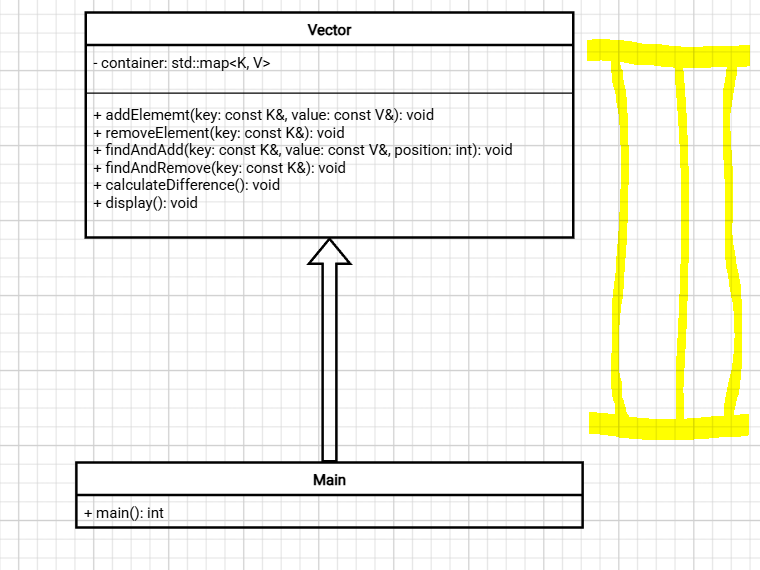
Персональная:



UML – диаграмма:







Программная реализация

**Заголовочный файлы**

**I  
container\_utils.h:**#pragma once

#include <map>

// Заполнить контейнер элементами

void fillMap(std::map<int, double>& myMap);

// Задание 3

void task3(std::map<int, double>& myMap, int key, int position);

// Задание 4

void task4(std::map<int, double>& myMap, int key);

// Задание 5

void task5(const std::map<int, double>& myMap); **II  
money.h:**#pragma once

#include <iostream>

class Money

{

public:

Money(int rub = 0, int kop = 0);

Money(const Money& other);

// Перегрузка операторов

Money operator+(const Money& other) const;

Money operator-(const Money& other) const;

bool operator<(const Money& other) const;

bool operator>(const Money& other) const;

// Получение значений

int getRub() const;

int getKop() const;

// Вывод

friend std::ostream& operator<<(std::ostream& out, const Money& money);

friend std::istream& operator>>(std::istream& in, Money& money);

private:

long rub; // Рубли

int kop; // Копейки

}; **III**

**vector.h:**#pragma once

#include <map>

#include <iostream>

#include <algorithm>

template <typename K, typename V>

class Vector

{

private:

std::map<K, V> container;

public:

void addElement(const K& key, const V& value);

void removeElement(const K& key);

void findAndAdd(const K& key, const V& value, int position);

void findAndRemove(const K& key);

void calculateDifference();

void display() const;

};

**Реализация методов**

**I  
container\_utils.cpp:**

#include <iostream>

#include <map>

#include <iterator>

#include "container\_utils.h"

void fillMap(std::map<int, double>& myMap)

{

myMap[1] = 1.1;

myMap[2] = 2.2;

myMap[3] = 3.3;

myMap[4] = 4.4;

myMap[5] = 5.5;

}

void task3(std::map<int, double>& myMap, int key, int position)

{

auto it = myMap.find(key);

if (it != myMap.end())

{

// Добавление элемента на заданную позицию

myMap.insert({ position, it->second });

}

else

{

std::cout << "Элемент с указанным ключом не найден.\n";

}

}

void task4(std::map<int, double>& myMap, int key)

{

auto it = myMap.find(key);

if (it != myMap.end())

{

// Удаление элемента

myMap.erase(it);

}

else

{

std::cout << "Элемент с указанным ключом не найден.\n";

}

}

void task5(const std::map<int, double>& myMap)

{

if (myMap.empty())

{

std::cout << "Контейнер пуст.\n";

return;

}

double maxVal = myMap.begin()->second;

double minVal = myMap.begin()->second;

for (const auto& pair : myMap)

{

if (pair.second > maxVal)

{

maxVal = pair.second;

}

if (pair.second < minVal)

{

minVal = pair.second;

}

}

double difference = maxVal - minVal;

for (const auto& pair : myMap)

{

std::cout << "Элемент: " << pair.first << " -> " << (pair.second - difference) << "\n";

}

} **II  
money.cpp:**#include "money.h"

// Конструктор

Money::Money(int rub, int kop) : rub(rub), kop(kop)

{

if (kop >= 100)

{

rub += kop / 100;

this->kop = kop % 100;

}

else

{

this->kop = kop;

}

}

// Конструктор копирования

Money::Money(const Money& other) : rub(other.rub), kop(other.kop) {}

// Перегрузка операторов

Money Money::operator+(const Money& other) const

{

return Money(rub + other.rub, kop + other.kop);

}

Money Money::operator-(const Money& other) const

{

return Money(rub - other.rub, kop - other.kop);

}

bool Money::operator<(const Money& other) const

{

return rub < other.rub || (rub == other.rub && kop < other.kop);

}

bool Money::operator>(const Money& other) const

{

return rub > other.rub || (rub == other.rub && kop > other.kop);

}

int Money::getRub() const

{

return rub;

}

int Money::getKop() const

{

return kop;

}

std::ostream& operator<<(std::ostream& out, const Money& money)

{

out << money.rub << " руб. " << money.kop << " коп.";

return out;

}

std::istream& operator>>(std::istream& in, Money& money)

{

in >> money.rub >> money.kop;

return in;

} **III**

**vector.cpp:**#include "vector.h"

// Метод для добавления элемента в контейнер

template <typename K, typename V>

void Vector<K, V>::addElement(const K& key, const V& value)

{

container[key] = value;

}

// Метод для удаления элемента из контейнера

template <typename K, typename V>

void Vector<K, V>::removeElement(const K& key)

{

container.erase(key);

}

// Поиск элемента с заданным ключом и добавление его на заданную позицию

template <typename K, typename V>

void Vector<K, V>::findAndAdd(const K& key, const V& value, int position)

{

auto it = container.find(key);

if (it != container.end())

{

// Добавляем на заданную позицию

auto insertIt = container.begin();

std::advance(insertIt, position);

container.insert(insertIt, \*it);

}

}

// Поиск элемента с заданным ключом и удаление его из контейнера

template <typename K, typename V>

void Vector<K, V>::findAndRemove(const K& key)

{

auto it = container.find(key);

if (it != container.end())

{

container.erase(it);

}

}

// Метод для нахождения разницы между максимальным и минимальным элементами

template <typename K, typename V>

void Vector<K, V>::calculateDifference()

{

if (container.empty()) return;

auto maxIt = std::max\_element(container.begin(), container.end(),

[](const auto& a, const auto& b) { return a.second < b.second; });

auto minIt = std::min\_element(container.begin(), container.end(),

[](const auto& a, const auto& b) { return a.second < b.second; });

V difference = maxIt->second - minIt->second;

for (auto& elem : container)

{

elem.second -= difference;

}

}

// Метод для отображения содержимого контейнера

template <typename K, typename V>

void Vector<K, V>::display() const

{

for (const auto& elem : container)

{

std::cout << "Ключ: " << elem.first << ", Ценa: " << elem.second << std::endl;

}

}

// Не забудьте явно инстанцировать шаблонный класс для необходимых типов

template class Vector<int, double>; // например, для int и double

**Функция main**

**I**#include <iostream>

#include "container\_utils.h"

int main()

{

std::map<int, double> myMap;

setlocale(LC\_ALL, "Russian");

// Заполнение карты элементами

fillMap(myMap);

// Задание 3

int keyToFind, positionToAdd;

std::cout << "Введите ключ для задания 3: ";

std::cin >> keyToFind;

std::cout << "Введите позицию для добавления: ";

std::cin >> positionToAdd;

task3(myMap, keyToFind, positionToAdd);

// Задание 4

std::cout << "Введите ключ для задания 4: ";

std::cin >> keyToFind;

task4(myMap, keyToFind);

// Задание 5

task5(myMap);

return 0;

} **II**#include <iostream>

#include <map>

#include <algorithm> // для std::max\_element и std::min\_element

#include "money.h"

void addElement(std::map<int, Money>& container, int key, const Money& value)

{

container[key] = value; // добавляет или обновляет элемент

}

void findAndAddElement(std::map<int, Money>& container, int key,

const Money& value)

{

// Если ключ существует, обновляем значение

if (container.find(key) != container.end())

{

container[key] = value; // обновляем значение

}

}

void findAndRemoveElement(std::map<int, Money>& container, int key)

{

container.erase(key); // удаляем элемент по ключу

}

void findDifference(std::map<int, Money>& container)

{

if (container.empty()) return; // Проверка на пустой контейнер

// max и min элементы

auto max\_it = std::max\_element(container.begin(), container.end(),

[](const auto& a, const auto& b)

{

return a.second < b.second;

});

auto min\_it = std::min\_element(container.begin(), container.end(),

[](const auto& a, const auto& b)

{

return a.second < b.second;

});

Money difference = max\_it->second - min\_it->second;

// Вычитание разности из каждого элемента

for (auto& pair : container)

{

pair.second = pair.second - difference;

}

}

int main()

{

setlocale(LC\_ALL, "RU");

std::map<int, Money> container;

// Пример заполнения контейнера

addElement(container, 1, Money(100, 50));

addElement(container, 2, Money(200, 75));

// Вывод элементов контейнера

for (const auto& pair : container)

{

std::cout << pair.first << ": " << pair.second << std::endl;

}

std::cout << std::endl;

// Выполнение заданий

findAndAddElement(container, 1, Money(50, 25)); // Задание 3

findAndRemoveElement(container, 1); // Задание 4

findDifference(container); // Задание 5

// Вывод элементов контейнера

for (const auto& pair : container)

{

std::cout << pair.first << ": " << pair.second << std::endl;

}

return 0;

} **III**

#include "vector.h"

int main()

{

Vector<int, double> vec;

setlocale(LC\_ALL, "Russian");

// Заполняем контейнер

vec.addElement(1, 2.5);

vec.addElement(2, 3.0);

vec.addElement(3, 1.5);

vec.addElement(4, 4.25);

std::cout << "Начальный контейнер:" << std::endl;

vec.display();

// Задание 3: Найти элемент с заданным ключом и добавить его на заданную позицию

vec.findAndAdd(2, 5.0, 1);

std::cout << "После нахождения и добавления:" << std::endl;

vec.display();

// Задание 4: Найти элемент с заданным ключом и удалить его

vec.findAndRemove(3);

std::cout << "После обнаружения и удаления:" << std::endl;

vec.display();

// Задание 5: Найти разницу и вычесть ее из каждого элемента

vec.calculateDifference();

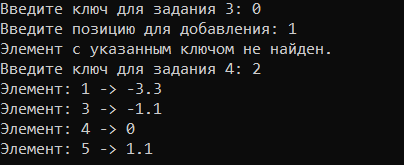
std::cout << "После вычисления разницы:" << std::endl;

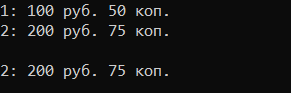
vec.display();

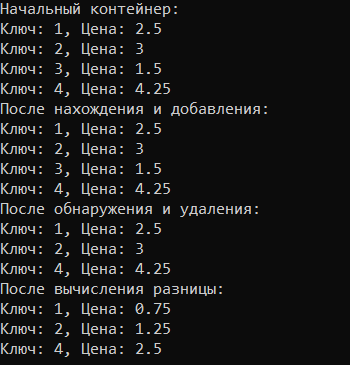
return 0;

}

Результат работы:

**I** **II**

 **III**



Контрольные вопросы:

1. Что представляет собой ассоциативный контейнер?

2. Перечислить ассоциативные контейнеры библиотеки STL.

3. Каким образом можно получить доступ к элементам ассоциативного

контейнера?

4. Привести примеры методов, используемых в ассоциативных контейнерах.

5. Каким образом можно создать контейнер map? Привести примеры.

6. Каким образом упорядочены элементы в контейнере тар по умолчанию? Как изменить порядок на обратный?

7. Какие операции определены для контейнера mаp?

8. Написать функцию для добавления элементов в контейнер mар с помощью функции make pair().

9. Написать функцию для добавления элементов в контейнер mаp с помощью функции операции прямого доступа [].

10. Написать функцию для печати контейнера mар с помощью итератора.

11. Написать функцию для печати контейнера mар с помощью функции операции прямого доступа [].

12. Чем отличаются контейнеры map и multimap?

13. Что представляет собой контейнер set?

14. Чем отличаются контейнеры map и set?

15. Каким образом можно создать контейнер set? Привести примеры.

16. Каким образом упорядочены элементы в контейнере set по умолчанию? Как изменить порядок на обратный?

17. Какие операции определены для контейнера set?

18. Написать функцию для добавления элементов в контейнер set.

19. Написать функцию для печати контейнера set.

20. Чем отличаются контейнеры set и multiset?

Ответы:

**1.Что представляет собой ассоциативный контейнер?**  
Ассоциативный контейнер представляет собой структуру данных, которая содержит пары значений (ключ и отображенное значение), позволяя получать доступ к одному значению по известному ключу.

**2.Перечислить ассоциативные контейнеры библиотеки STL.**  
Ассоциативные контейнеры библиотеки STL включают: map, multimap, set, и multiset.

**3.Каким образом можно получить доступ к элементам ассоциативного контейнера?**  
Доступ к элементам ассоциативного контейнера можно получить с помощью итераторов или операции индексации (operator[]).

**4.Привести примеры методов, используемых в ассоциативных контейнерах.**  
Примеры методов включают: insert(), erase(), count(), и операции сравнения (==, <, >, и т.д.).

**5.Каким образом можно создать контейнер map? Привести примеры.**  
Контейнер map можно создать следующим образом:  
#include <map>

typedef std::map<int, float> tmap; // определение типа tmap

tmap ml; // создание пустого контейнера map

**6.Каким образом упорядочены элементы в контейнере map по умолчанию? Как изменить порядок на обратный?**  
Элементы в контейнере map по умолчанию упорядочены по возрастанию ключа. Чтобы изменить порядок на обратный, можно использовать компаратор greater<Key> при создании контейнера.

**7.Какие операции определены для контейнера map?**  
Для контейнера map определены операции: ==, <, <=, !=, >, и >=.

**8.Написать функцию для добавления элементов в контейнер map с помощью функции make\_pair().**void addElementToMap(tmap& m, int key, float value)

{

m.insert(std::make\_pair(key, value));

}

**9.Написать функцию для добавления элементов в контейнер map с помощью функции операции прямого доступа [ ].**void addElementToMap(tmap& m, int key, float value)

{

m[key] = value; // использование операции прямого доступа

}

**10.Написать функцию для печати контейнера map с помощью итератора.**

void printMap(const tmap& m)

{

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

{

std::cout << it->first << ": " << it->second << std::endl;

}

}

**11.Написать функцию для печати контейнера map с помощью функции операции прямого доступа [].**

void printMap(const tmap& m)

{

for (const auto& pair : m)

{

std::cout << pair.first << ": " << m[pair.first] << std::endl;

}

}

**12.Чем отличаются контейнеры map и multimap?**  
Контейнер map хранит уникальные ключи, тогда как multimap допускает хранение элементов с одинаковыми ключами.

**13.Что представляет собой контейнер set?**  
Контейнер set представляет собой ассоциативный массив, где хранятся уникальные значения (ключи), и порядок их хранения отсортирован.

**14.Чем отличаются контейнеры map и set?**  
Контейнер map хранит пары ключ/значение, тогда как set хранит только уникальные ключи без связанных значений.

**15.Каким образом можно создать контейнер set? Привести примеры.**  
Контейнер set можно создать следующим образом:

**16.Каким образом упорядочены элементы в контейнере set по умолчанию? Как изменить порядок на обратный?**  
Элементы в контейнере set по умолчанию упорядочены по возрастанию. Чтобы изменить порядок на обратный, можно использовать компаратор greater<T> при создании контейнера.

**17.Какие операции определены для контейнера set?**  
Контейнер set позволяет выполнять следующие операции:

* Вставка элементов с помощью метода insert().
* Удаление элементов с помощью метода erase().
* Подсчет количества объектов с заданным ключом с помощью метода count().

**18.Написать функцию для добавления элементов в контейнер set.**  
Пример функции для добавления элементов в контейнер set:

#include <set>

void addElement(std::set<int>& mySet, int element)

{

mySet.insert(element);

}

**19.Написать функцию для печати контейнера set.**  
Пример функции для печати элементов контейнера set:

#include <iostream>

#include <set>

void printSet(const std::set<int>& mySet)

{

for (const auto& element : mySet)

{

std::cout << element << " ";

}

std::cout << std::endl;

}

**20.Чем отличаются контейнеры set и multiset?**  
Основные отличия между set и multiset заключаются в следующем:

* В контейнере set все ключи уникальны, то есть не может быть двух одинаковых элементов. В multiset, наоборот, допускаются дубликаты, и элементы с одинаковыми ключами могут храниться.
* В set доступ по индексу не поддерживается, так как элементы уникальны, в то время как multiset также не поддерживает доступ по индексу, но позволяет хранить несколько элементов с одинаковыми ключами.

Ссылка:

**https://github.com/LeonidZhir/-**