МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Факультет: Электротехнический

Кафедра: «Информационные технологии и автоматизированные системы» (ИТАС)

Направление: Разработка информационных систем (РИС)

**О Т Ч Ё Т**   
**о лабораторной работе №12**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Ассоциативные контейнеры библиотеки STL»

Выполнил:

Студент группы РИС-24-3б

Суяргулов А.О.

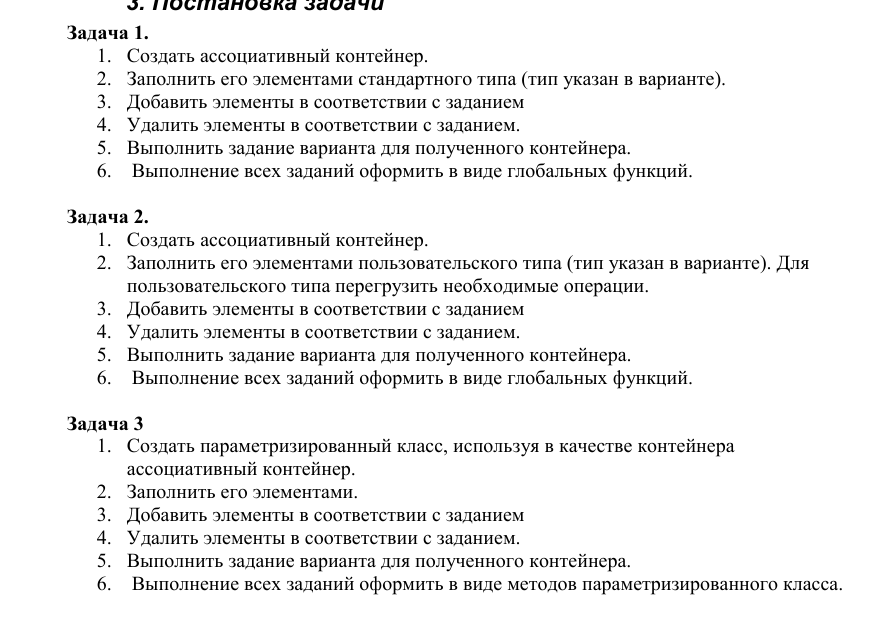
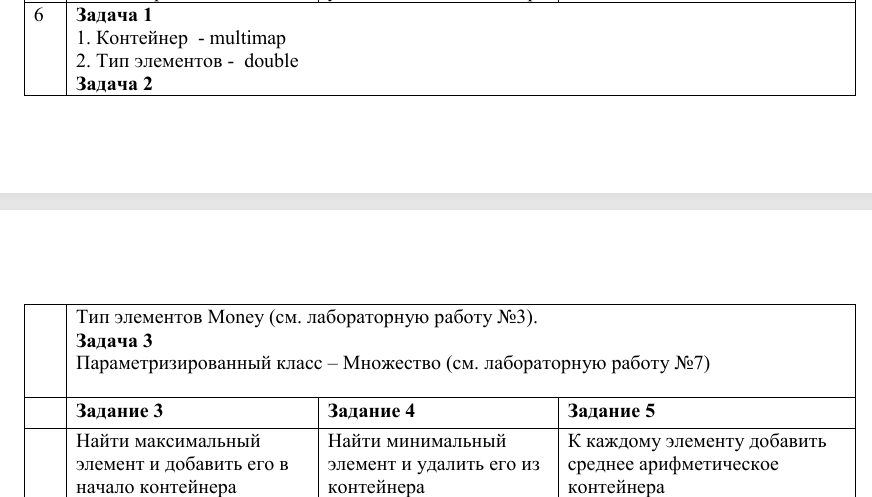
Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2025

ВВЕДЕНИЕ



ОСНОВНАЯ ЧАСТЬ

**Диаграмма классов:**

**Код:**

**Main1:**

#include <iostream>

#include <map>

using namespace std;

// Функция для добавления максимального элемента в начало

void addMaxToBegin(multimap<double, double>& mm) {

if (mm.empty()) return;

// Находим максимальный ключ (последний элемент в multimap)

auto max\_it = --mm.end();

double max\_key = max\_it->first;

double max\_value = max\_it->second;

// Создаем новый ключ меньше минимального, чтобы элемент стал первым

double new\_key = mm.begin()->first - 1.0;

mm.insert(make\_pair(new\_key, max\_value));

}

// Функция для удаления минимального элемента

void removeMin(multimap<double, double>& mm) {

if (!mm.empty()) {

mm.erase(mm.begin());

}

}

// Функция для добавления среднего арифметического к значениям

void addAverage(multimap<double, double>& mm) {

if (mm.empty()) return;

// Считаем сумму всех значений

double sum = 0.0;

for (auto& pair : mm) {

sum += pair.second;

}

// Вычисляем среднее

double avg = sum / mm.size();

// Обновляем значения

multimap<double, double> temp;

for (auto& pair : mm) {

temp.insert(make\_pair(pair.first, pair.second + avg));

}

mm = temp;

}

int main() {

setlocale(LC\_ALL, "RU");

multimap<double, double> myMap;

// Заполняем контейнер

myMap.insert(make\_pair(1.1, 2.5));

myMap.insert(make\_pair(3.3, 4.0));

myMap.insert(make\_pair(5.5, 6.5));

// Выполняем задания

addMaxToBegin(myMap);

removeMin(myMap);

addAverage(myMap);

// Выводим результат

cout << "Результат:\n";

for (auto& pair : myMap) {

cout << "Ключ: " << pair.first << ", Значение: " << pair.second << endl;

}

return 0;

}

**Main2:**

#include <iostream>

#include <map>

class Money {

private:

double amount;

public:

Money(double amt) : amount(amt) {}

bool operator<(const Money& other) const { return amount < other.amount; }

Money operator+(const Money& other) const { return Money(amount + other.amount); }

double getAmount() const { return amount; }

};

// Исправлено: добавлено std::

std::ostream& operator<<(std::ostream& os, const Money& m) {

os << m.getAmount();

return os;

}

using MoneyMultiMap = std::multimap<double, Money>;

void addMaxToBegin(MoneyMultiMap& mm) {

if (mm.empty()) return;

auto max\_it = --mm.end();

double new\_key = mm.begin()->first - 1.0;

mm.insert(std::make\_pair(new\_key, max\_it->second));

}

void removeMin(MoneyMultiMap& mm) {

if (!mm.empty()) mm.erase(mm.begin());

}

void addAverage(MoneyMultiMap& mm) {

if (mm.empty()) return;

double sum = 0.0;

for (const auto& pair : mm) {

sum += pair.second.getAmount();

}

double avg = sum / mm.size();

MoneyMultiMap temp;

for (const auto& pair : mm) {

temp.insert(std::make\_pair(pair.first, Money(pair.second.getAmount() + avg)));

}

mm = temp;

}

int main() {

setlocale(LC\_ALL, "RU");

MoneyMultiMap moneyMap;

moneyMap.insert(std::make\_pair(1.1, Money(2.5)));

moneyMap.insert(std::make\_pair(3.3, Money(4.0)));

moneyMap.insert(std::make\_pair(5.5, Money(6.5)));

addMaxToBegin(moneyMap);

removeMin(moneyMap);

addAverage(moneyMap);

std::cout << "Результат:\n";

for (const auto& pair : moneyMap) {

std::cout << "Ключ: " << pair.first << ", Значение: " << pair.second << std::endl;

}

return 0;

}

**Main3:**

#include <iostream>

#include <set>

template<typename T>

class MySet {

private:

std::multiset<T> data;

public:

void add(const T& elem) {

data.insert(elem);

}

void remove(const T& elem) {

// Удаляет все вхождения элемента

data.erase(elem);

}

// Добавить копию максимального элемента

void addMaxToBegin() {

if (data.empty()) return;

T maxElem = \*data.rbegin(); // Получаем максимальный элемент

data.insert(maxElem);

}

// Удалить минимальный элемент

void removeMin() {

if (!data.empty()) {

data.erase(data.begin());

}

}

// Добавить среднее арифметическое к каждому элементу

void addAverageToEach() {

if (data.empty()) return;

// Считаем сумму

T sum = T();

for (const T& elem : data) {

sum = sum + elem; // Требуется перегрузка operator+ для типа T

}

// Вычисляем среднее

double avg = static\_cast<double>(sum) / data.size();

// Обновляем элементы

std::multiset<T> temp;

for (const T& elem : data) {

temp.insert(elem + avg); // Требуется перегрузка operator+ для T + double

}

data.swap(temp);

}

// Вывод элементов

void print() const {

for (const T& elem : data) {

std::cout << elem << " ";

}

std::cout << std::endl;

}

};

int main() {

MySet<double> s;

s.add(2.5);

s.add(4.0);

s.add(6.5);

s.addMaxToBegin();

s.removeMin();

s.addAverageToEach();

std::cout << "Result: ";

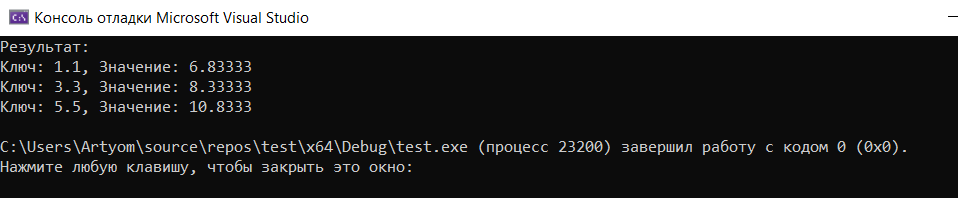
s.print();

return 0;

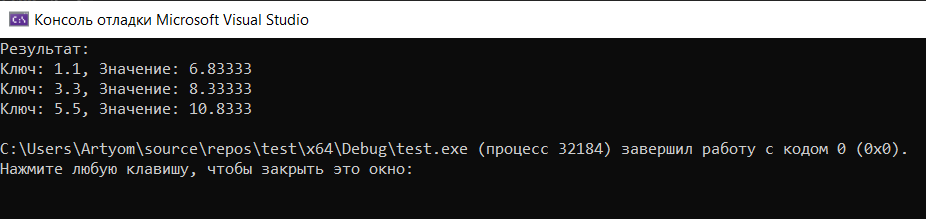
}

**Результат работы кода:**

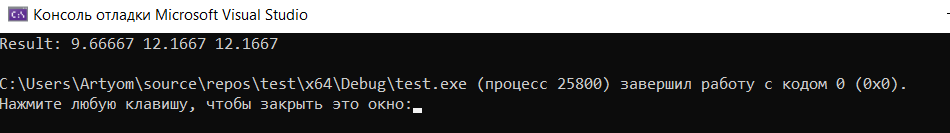
**1.**

****

**2.**

****

**3.**

****

**Контрольные вопросы:**

1. Что представляет собой ассоциативный контейнер?

\* Ассоциативный контейнер — это тип контейнера, в котором элементы хранятся в соответствии с определенным ключом, что обеспечивает быстрый доступ к ним по этому ключу. В отличие от последовательных контейнеров (таких как vector или list), элементы не обязательно хранятся в порядке их добавления.

2. Перечислите ассоциативные контейнеры библиотеки STL.

\* map

\* multimap

\* set

\* multiset

3. Каким образом можно получить доступ к элементам ассоциативного контейнера?

\* С помощью ключа (для map и multimap).

\* Итераторов.

\* Методов find, count, lower\_bound, upper\_bound и equal\_range.

4. Привести примеры методов, используемых в ассоциативных контейнерах.

\* insert(): добавляет элемент.

\* erase(): удаляет элемент.

\* find(): находит элемент по ключу.

\* count(): возвращает количество элементов с заданным ключом.

\* lower\_bound(): возвращает итератор на первый элемент, не меньший заданного ключа.

\* upper\_bound(): возвращает итератор на первый элемент, больший заданного ключа.

\* equal\_range(): возвращает пару итераторов, определяющих диапазон элементов с заданным ключом.

\* begin(), end(): возвращают итераторы на начало и конец контейнера.

\* size(): возвращает количество элементов в контейнере.

\* empty(): проверяет, пуст ли контейнер.

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

\* std::map<int, std::string> myMap; // Создает пустой map, где ключ - int, значение - string

\* std::map<std::string, double> myMap = {{"apple", 1.5}, {"banana", 0.75}}; // Создает map с начальными значениями

\* std::map<int, std::string> myMap(anotherMap.begin(), anotherMap.end()); // Создает map, копируя элементы из другого map

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

\* По умолчанию элементы в map упорядочены по возрастанию ключа, используя оператор <.

\* Чтобы изменить порядок на обратный, можно использовать компаратор std::greater:

\* std::map<int, std::string, std::greater<int>> myMap;

7. Какие операции определены для контейнера map?

\* Вставка элемента (insert)

\* Удаление элемента (erase)

\* Поиск элемента (find)

\* Доступ к элементу по ключу (operator[])

\* Подсчет количества элементов с заданным ключом (count)

\* Получение итераторов на начало и конец (begin, end)

\* Проверка на пустоту (empty)

\* Определение размера (size)

\* Получение диапазона элементов с заданным ключом (equal\_range, lower\_bound, upper\_bound)

8. Написать функцию для добавления элементов в контейнер map с помощью функции make\_pair().

#include <iostream>

#include <map>

void addElementUsingMakePair(std::map<int, std::string>& myMap, int key, const std::string& value) {

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

}

int main() {

std::map<int, std::string> myMap;

addElementUsingMakePair(myMap, 1, "apple");

addElementUsingMakePair(myMap, 2, "banana");

for (const auto& pair : myMap) {

std::cout << "Key: " << pair.first << ", Value: " << pair.second << std::endl;

}

return 0;

}

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

#include <iostream>

#include <map>

void addElementUsingBracketOperator(std::map<int, std::string>& myMap, int key, const std::string& value) {

myMap[key] = value;

}

int main() {

std::map<int, std::string> myMap;

addElementUsingBracketOperator(myMap, 1, "apple");

addElementUsingBracketOperator(myMap, 2, "banana");

for (const auto& pair : myMap) {

std::cout << "Key: " << pair.first << ", Value: " << pair.second << std::endl;

}

return 0;

}

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

#include <iostream>

#include <map>

void printMapUsingIterator(const std::map<int, std::string>& myMap) {

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

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

}

}

int main() {

std::map<int, std::string> myMap = {{1, "apple"}, {2, "banana"}};

printMapUsingIterator(myMap);

return 0;

}

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

#include <iostream>

#include <map>

void printMapUsingBracketOperator(const std::map<int, std::string>& myMap) {

for (const auto& pair : myMap) {

std::cout << "Key: " << pair.first << ", Value: " << myMap.at(pair.first) << std::endl;

}

}

int main() {

std::map<int, std::string> myMap = {{1, "apple"}, {2, "banana"}};

printMapUsingBracketOperator(myMap);

return 0;

}

12. Чем отличаются контейнеры map и multimap?

\* map: Каждый ключ может встречаться только один раз.

\* multimap: Один ключ может встречаться несколько раз.

13. Что представляет собой контейнер set?

\* set — это ассоциативный контейнер, содержащий уникальные элементы, упорядоченные в соответствии с определенным критерием. В set нет ключей и значений, только значения.

14. Чем отличаются контейнеры map и set?

\* map: Хранит пары "ключ-значение". Ключи должны быть уникальными.

\* set: Хранит только уникальные значения.

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

\* std::set<int> mySet; // Создает пустой set целых чисел.

\* std::set<std::string> mySet = {"apple", "banana", "cherry"}; // Создает set строк с начальными значениями

\* std::set<int> mySet(anotherSet.begin(), anotherSet.end()); // Создает set, копируя элементы из другого set

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

\* По умолчанию элементы в set упорядочены по возрастанию, используя оператор <.

\* Чтобы изменить порядок на обратный, можно использовать компаратор std::greater:

\* std::set<int, std::greater<int>> mySet;

17. Какие операции определены для контейнера set?

\* Вставка элемента (insert)

\* Удаление элемента (erase)

\* Поиск элемента (find)

\* Подсчет количества элементов, равных заданному значению (count)

\* Получение итераторов на начало и конец (begin, end)

\* Проверка на пустоту (empty)

\* Определение размера (size)

\* Получение итераторов на элементы, удовлетворяющие определенным критериям (equal\_range, lower\_bound, upper\_bound)

18. Написать функцию для добавления элементов в контейнер set.

#include <iostream>

#include <set>

void addElementToSet(std::set<int>& mySet, int value) {

mySet.insert(value);

}

int main() {

std::set<int> mySet;

addElementToSet(mySet, 10);

addElementToSet(mySet, 20);

addElementToSet(mySet, 10); // Попытка добавления дубликата - не будет добавлен

for (int value : mySet) {

std::cout << value << " ";

}

std::cout << std::endl;

return 0;

}

19. Написать функцию для печати контейнера set.

ChatGPT4 | Midjourney, [07.05.2025 3:25]

#include <iostream>

#include <set>

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

for (int value : mySet) {

std::cout << value << " ";

}

std::cout << std::endl;

}

int main() {

std::set<int> mySet = {10, 20, 30};

printSet(mySet);

return 0;

}

20. Чем отличаются контейнеры set и multiset?

• set: Содержит только уникальные элементы.

• multiset: Может содержать повторяющиеся элементы.

**Ссылка на GitHub:**

[Drazda1488/oop\_12lab](https://github.com/Drazda1488/oop_12lab)