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 №12

# По теме **«Ассоциативные контейнеры библиотеки STL»**

Вариант №10

Выполнял:

студент группы РИС-24-1б

Морозова Н.С.

Проверял:

Доцент кафедры ИТАС

Полякова О.А.

Постановка задач:

### 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.

Задания:

Задача 1

1. Контейнер - multimap
2. Тип элементов - float

Задача 2

Тип элементов Money (см. лаб. работу №3).

Задача 3

Параметризированный класс – Вектор (см. лаб. работу №7)

|  |  |  |
| --- | --- | --- |
| Задание 3 | Задание 4 | Задание 5 |
| Найти минимальный элемент и добавить его на заданную позицию  контейнера | Найти элементы большие среднего арифметического и  удалить их из контейнера | Каждый элемент домножить на максимальный элемент контейнера |

Анализ задач:

1. Задача 1, файл Лаб\_11.1 (да, у меня всё ещё 11 л/р, потому что перепутала), похожа на лабораторную работу №11
2. Во 2, файл Лаб\_11.2, задаче были перегружены операции сложения, умножения вывода и был реализован перевод денежной суммы в целое число для удобства реализации некоторых операции.
3. В 3 задаче, файл Лаба\_12.3 (и вот тут я вспомнила, что я делаю 12 л/р), реализован параметризированный класс My\_cont.

Решение

Код. Задача 1

Файл Лаб\_11.1

#include <iostream>

#include <map>

#include <algorithm>

#include <numeric>

#include <Windows.h>

using namespace std;

typedef multimap<int, float> Multi;

Multi make\_multimap(int n) {

Multi mm;

srand(time(0));

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

int key = i;

float value = rand() % 100 / 10.0f;

mm.insert(make\_pair(key, value));

}

return mm;

}

void print(const Multi& mm) {

for (const auto& pair : mm) {

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

}

}

// Задание 3: Вставка минимума на заданную позицию

void insert\_min\_at(Multi& mm, int pos) {

if (mm.empty()) throw runtime\_error("Словарь пустой!");

// Находим минимальный элемент

auto min\_it = min\_element(mm.begin(), mm.end(),

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

float min\_val = min\_it->second;

cout << "Минимальный элемент: " << min\_val << endl;

// Создаем новый ключ для вставки

auto it = mm.begin();

advance(it, pos);

int new\_key = (pos > 0) ? prev(it)->first + 0.1f : it->first - 1.0f;

// Вставляем копию элемента

mm.insert({ new\_key, min\_it->second });

}

// Задание 4: Удаление элементов > среднего

void remove\_above\_avg(Multi& mm) {

if (mm.empty()) return;

// Вычисляем среднее значение

float sum = 0;

for (const auto& pair : mm) sum += pair.second;

float avg = sum / mm.size();

cout << "\nСреднее значение: " << avg << endl;

// Удаляем элементы > среднего

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

if (it->second > avg)

it = mm.erase(it);

else ++it;

}

}

// Задание 5: Умножение на максимум

void mult\_max(Multi& mm) {

if (mm.empty()) return;

// Находим максимальный элемент

auto max\_it = max\_element(mm.begin(), mm.end(),

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

float max\_val = max\_it->second;

cout << "\nМаксимальный элемент: " << max\_val << endl;

// Умножаем все элементы на максимум

for (auto& pair : mm) {

pair.second \*= max\_val;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try {

// 1. Создание и заполнение multimap

int n;

cout << "Введите количество элементов: ";

cin >> n;

Multi mm = make\_multimap(n);

cout << "Исходный словарь:" << endl;

print(mm);

// Задание 3: Вставка минимума

int pos;

cout << "\nВведите позицию для вставки (1-" << mm.size() << "): ";

cin >> pos;

insert\_min\_at(mm, pos - 1);

cout << "После вставки минимума:" << endl;

print(mm);

// Задание 4: Удаление > среднего

remove\_above\_avg(mm);

cout << "После удаления больше среднего:" << endl;

print(mm);

// Задание 5: Умножение на максимум

mult\_max(mm);

cout << "После умножения на максимум : " << endl;

print(mm);

}

catch (const exception& e) {

cerr << "Ошибка: " << e.what() << endl;

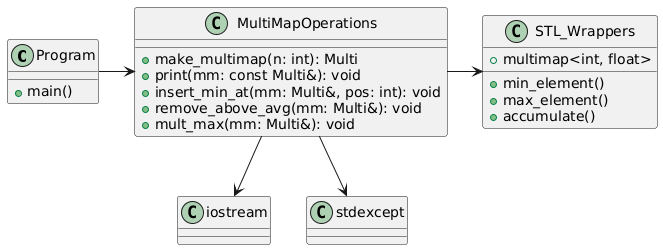
return 1;

}

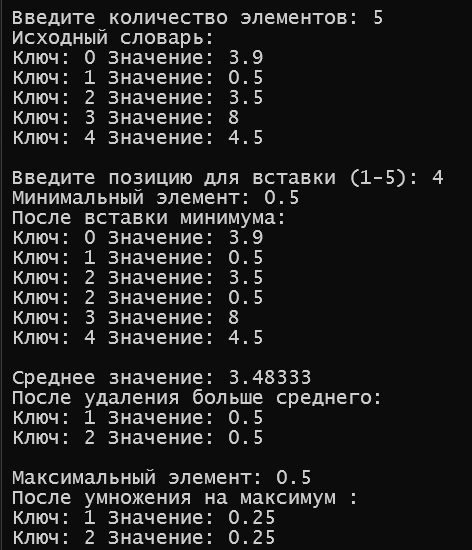
return 0;

}

UML-диаграмма



Результат работы программы:



Код. Задача 2

Файл Лаб\_11.2

#include <iostream>

#include <map>

#include <vector>

#include <algorithm>

#include <numeric>

#include <Windows.h>

#include <iomanip>

using namespace std;

class Money {

private:

int rubles;

int kopecks;

void normalize() {

rubles += kopecks / 100;

kopecks %= 100;

}

public:

Money(int r = 0, int k = 0) : rubles(r), kopecks(k) { normalize(); }

// Для сравнения

bool operator<(const Money& other) const {

return (rubles < other.rubles) ||

(rubles == other.rubles && kopecks < other.kopecks);

}

// Для умножения (задание 5)

Money operator\*(float multiplier) const {

float total = rubles + kopecks / 100.0f;

total \*= multiplier;

return Money(static\_cast<int>(total),

static\_cast<int>((total - static\_cast<int>(total)) \* 100));

}

// Для вывода

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

os << m.rubles << "," << setw(2) << setfill('0') << m.kopecks << " руб.";

return os;

}

// Для удобства, делает рубли и копейки целыми числами

float toFloat() const { return rubles + kopecks / 100.0f; }

};

typedef multimap<int, Money> MoneyMap;

MoneyMap make(int n) {

MoneyMap m;

srand(time(0));

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

int key = i;

Money value(rand() % 100, rand() % 100);

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

}

return m;

}

void print(const MoneyMap& m) {

for (const auto& pair : m)

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

}

// Задание 3: Вставка минимума на заданную позицию

void inMin(MoneyMap& m, int pos) {

if (m.empty()) throw runtime\_error("Словарь пустой!");

// Находим минимальный элемент

auto min\_it = min\_element(m.begin(), m.end(),

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

float min\_val = min\_it->second.toFloat();

cout << "Минимальный элемент: " << min\_val << endl;

//Создаем новый ключ для вставки

int new\_key;

auto it = m.begin();

advance(it, pos);

new\_key = (pos > 0) ? prev(it)->first + 0.1f : it->first - 1.0f;

// Вставляем копию элемента

m.insert({ new\_key, min\_it->second });

}

// Задание 4: Удаление элементов > среднего

void remove\_above\_avg(MoneyMap& m) {

if (m.empty()) return;

// Вычисляем среднее значение

float sum = 0;

for (const auto& pair : m) sum += pair.second.toFloat();

float avg = sum / m.size();

cout << "\nСреднее значение: " << fixed << setprecision(2) << avg << " руб." << endl;

// Удаляем элементы > среднего

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

if (it->second.toFloat() > avg)

it = m.erase(it);

else

++it;

}

}

// Задание 5: Умножение на максимум

void mult\_max(MoneyMap& m) {

if (m.empty()) return;

// Находим максимальный элемент

auto max\_it = max\_element(m.begin(), m.end(),

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

float max\_val = max\_it->second.toFloat();

cout << "\nМаксимальный элемент: " << max\_it->second << endl;

// Умножаем все элементы на максимум

for (auto& pair : m)

pair.second = pair.second \* max\_val;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try {

// 1. Создание и заполнение

int n;

cout << "Введите количество элементов: ";

cin >> n;

MoneyMap m = make(n);

cout << "Исходный словарь:" << endl;

print(m);

// Задание 3: Вставка минимума

int pos;

cout << "\nВведите позицию для вставки (1-" << m.size() << "): ";

cin >> pos;

inMin(m, pos - 1);

cout << "После вставки минимума:" << endl;

print(m);

// Задание 4: Удаление > среднего

remove\_above\_avg(m);

cout << "После удаления больше среднего:" << endl;

print(m);

// Задание 5: Умножение на максимум

mult\_max(m);

cout << "После умножения на максимум:" << endl;

print(m);

}

catch (const exception& e) {

cerr << "Ошибка: " << e.what() << endl;

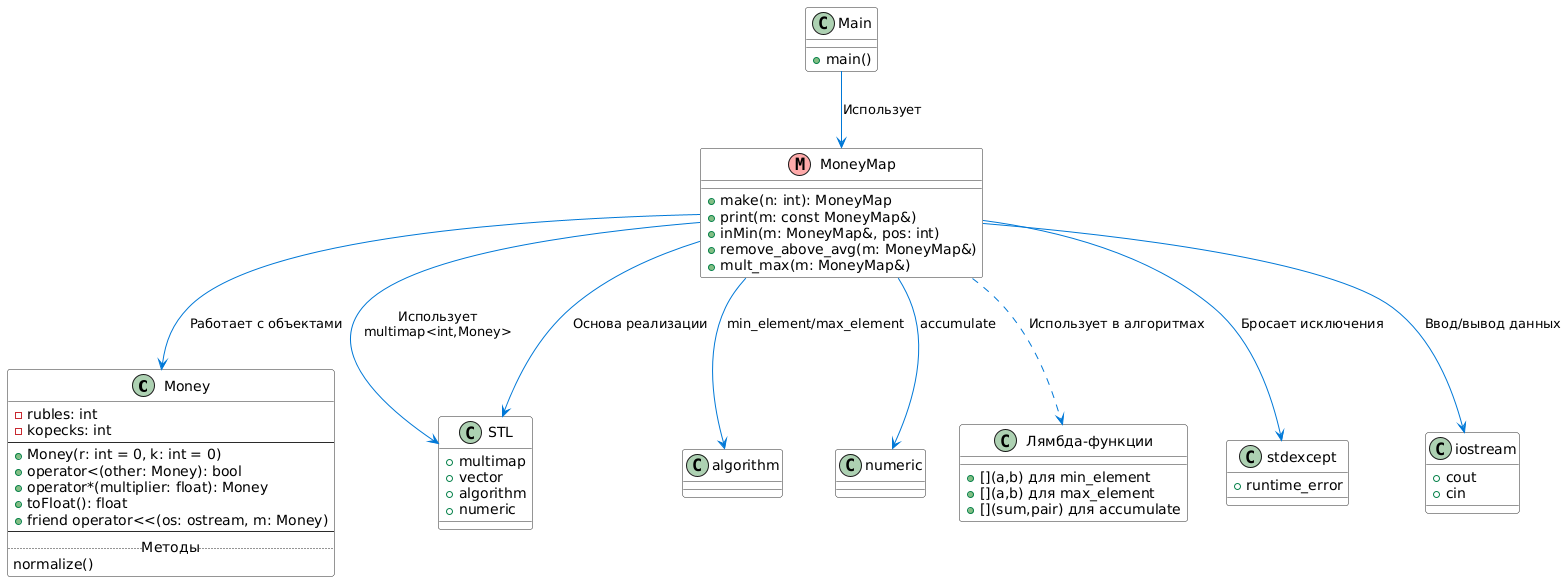
return 1;

}

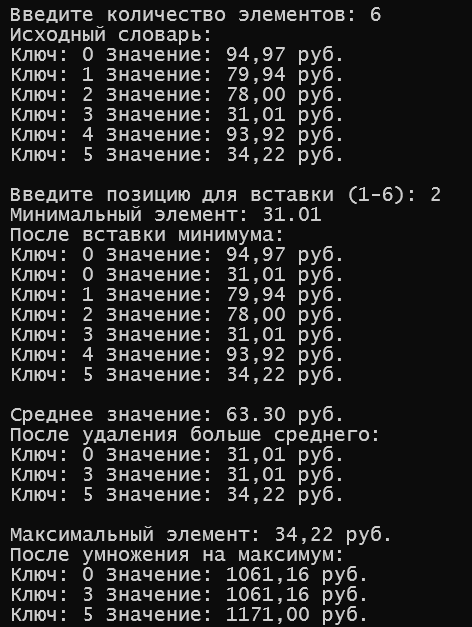
return 0;

}

UML-диаграмма



Результат работы программы:



Код. Задача 3

Файл Лаба\_12.3

#include <iostream>

#include <map>

#include <algorithm>

#include <numeric>

#include <Windows.h>

#include <iomanip>

using namespace std;

// Класс Money из 2 задачи

class Money {

private:

int rubles;

int kopecks;

void normalize() {

rubles += kopecks / 100;

kopecks %= 100;

}

public:

Money(int r = 0, int k = 0) : rubles(r), kopecks(k) { normalize(); }

bool operator<(const Money& other) const {

return (rubles < other.rubles) ||

(rubles == other.rubles && kopecks < other.kopecks);

}

Money operator\*(float multiplier) const {

float total = rubles + kopecks / 100.0f;

total \*= multiplier;

return Money(static\_cast<int>(total),

static\_cast<int>((total - static\_cast<int>(total)) \* 100));

}

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

os << m.rubles << "," << setw(2) << setfill('0') << m.kopecks << " руб.";

return os;

}

float toFloat() const { return rubles + kopecks / 100.0f; }

};

// Параметризированный класс с multimap

template <typename Key, typename Value>

class My\_cont {

private:

multimap<Key, Value> data;

public:

// Добавление элемента (дубли ключей разрешены)

void add(const Key& key, const Value& value) { data.insert(make\_pair(key, value)); }

// Удаление элемента по ключу

void remove(const Key& key) { data.erase(key); }

// Печать контейнера

void print() const {

for (const auto& pair : data)

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

}

// Задание 3: Вставка минимума на позицию

void in\_min(int pos) {

if (data.empty()) throw runtime\_error("Контейнер пустой!");

// Находим минимальный элемент по значению

auto min\_it = min\_element(data.begin(), data.end(),

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

float min\_val = min\_it->second.toFloat();

cout << "Минимальный элемент: " << min\_val << endl;

// Создаем новый ключ для вставки

auto it = data.begin();

advance(it, pos);

Key new\_key = (pos > 0) ? prev(it)->first + 0.1f : it->first - 1.0f;

// Вставляем копию элемента

data.insert({ new\_key, min\_it->second });

}

// Задание 4: Удаление элементов > среднего

void rem\_average() {

if (data.empty()) return;

// Вычисляем среднее значение

float sum = 0;

for (const auto& pair : data) sum += pair.second.toFloat();

float avg = sum / data.size();

cout << "\nСреднее значение: " << avg << endl;

// Удаляем элементы > среднего

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

if (it->second.toFloat() > avg)

it = data.erase(it);

else

++it;

}

}

// Задание 5: Умножение на максимум

void mult\_max() {

if (data.empty()) return;

// Находим максимальный элемент

auto max\_it = max\_element(data.begin(), data.end(),

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

float max\_val = max\_it->second.toFloat();

cout << "\nМаксимальный элемент: " << max\_it->second << endl;

// Создаем временный контейнер для модифицированных значений

multimap<Key, Value> new\_data;

for (auto& pair : data)

new\_data.insert({ pair.first, pair.second \* max\_val });

data = move(new\_data);

}

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try {

// Демонстрация для Money

My\_cont<int, Money> money\_cont;

int n;

cout << "Введите количество элементов: ";

cin >> n;

for (int i = 0; i < n; i++)

money\_cont.add(i, Money(rand() % 100, rand() % 100));

cout << "Исходный словарь:" << endl;

money\_cont.print();

// Задание 3: Вставка минимума

int pos;

cout << "\nВведите позицию для вставки: ";

cin >> pos;

money\_cont.in\_min(pos - 1);

cout << "После вставки минимума:" << endl;

money\_cont.print();

// Задание 4: Удаление > среднего

money\_cont.rem\_average();

cout << "После удаления больше среднего:" << endl;

money\_cont.print();

// Задание 5: Умножение на максимум

money\_cont.mult\_max();

cout << "После умножения на максимум:" << endl;

money\_cont.print();

}

catch (const exception& e) {

cerr << "Ошибка: " << e.what() << endl;

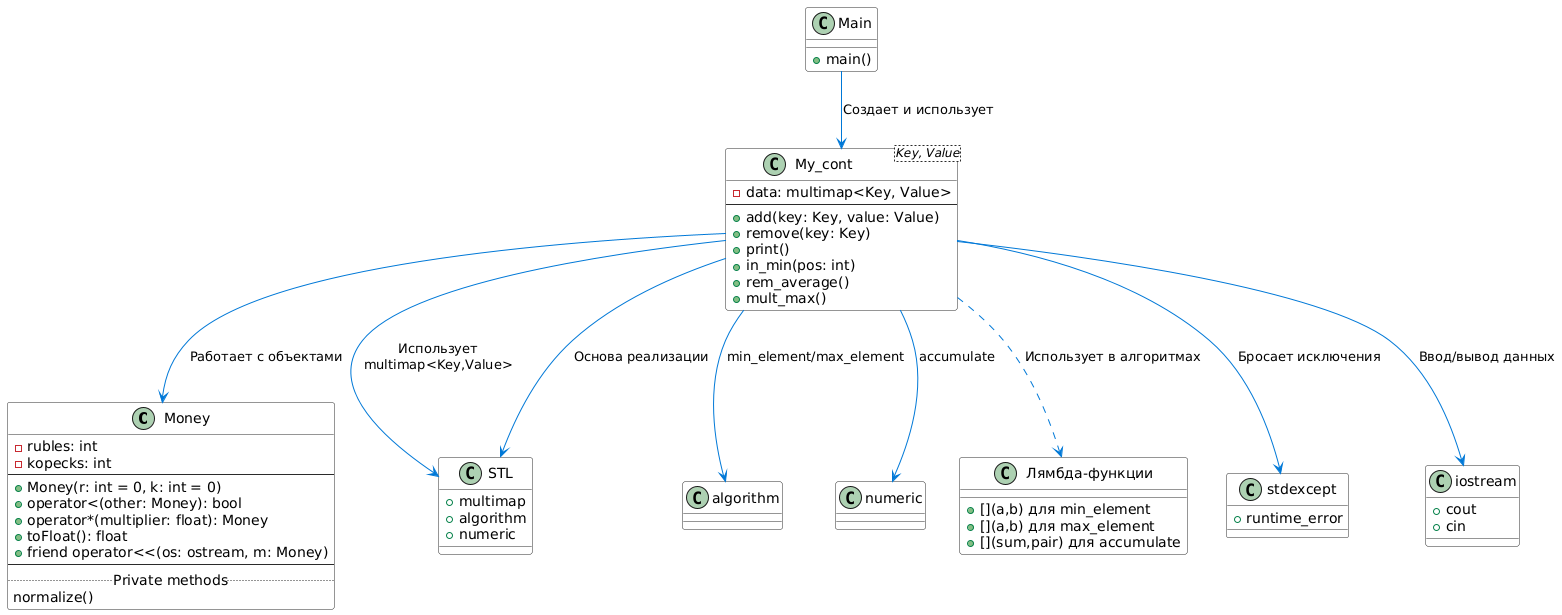
return 1;

}

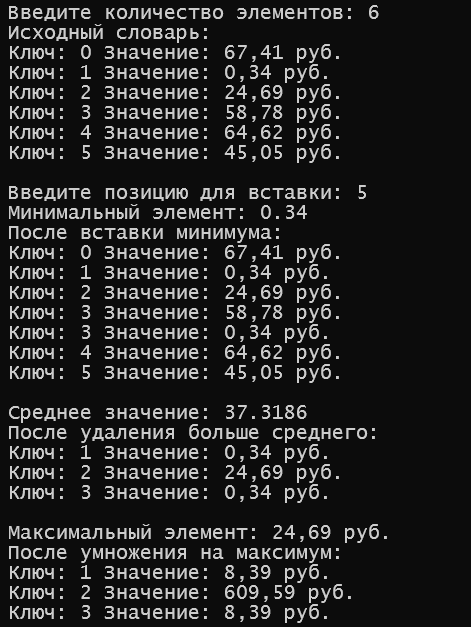
return 0;

}

UML-диаграмма



Результат работы программы:



Контрольные вопросы:

1. Что представляет собой ассоциативный контейнер?

Ассоциативный контейнер в C++ — это контейнер, который хранит элементы в виде пар "ключ-значение". Он обеспечивает быстрый доступ к значениям по их ключам. Ключи уникальны (или могут повторяться в multimap/multiset), а значения располагаются в определенном порядке (или без него в неупорядоченных контейнерах).

1. Перечислить ассоциативные контейнеры библиотеки STL.

Упорядоченные: (словарь) map, (множество) set, (словарь с дубликатами) multimap, (множество с дубликатами) multiset.

Неупорядоченные (хеш-таблицы, C++11 и выше): unordered\_map, unordered\_set, unordered\_multimap, unordered\_multiset.

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

Через итераторы (begin(), end(), find()).

Через оператор прямого доступа [] (только для map и unordered\_map).

Через метод at() (аналогично [], но с проверкой границ).

1. Привести примеры методов, используемых в ассоциативных контейнерах.

insert() – добавление элемента или диапазон элементов.

erase() – удаление элемента или диапазон элементов.

find() – поиск элемента по ключу.

count() – подсчет элементов с заданным ключом.

size() – количество элементов.

empty() – проверка на пустоту.

clear() – очистка контейнера.

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

#include <iostream>

#include <map>

#include <string>

int main() {

// Создание map с ключами типа string и значениями типа int

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

// Добавление элементов

ageMap["Alice"] = 25;

ageMap["Bob"] = 30;

// Или через insert

ageMap.insert(std::make\_pair("Charlie", 35));

for (const auto& pair : ageMap) { // Вывод

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

}

return 0;

}

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

По умолчанию map сортирует элементы по возрастанию ключей (используя std::less<Key>). Чтобы изменить порядок на обратный:

#include <map>

#include <string>

#include <functional> // для greater

int main() {

std::map<std::string, int, std::greater<std::string>> reverseMap;

reverseMap["Alice"] = 25;

reverseMap["Bob"] = 30; //Будет отсортировано в порядке убывания ключей

return 0;

}

1. Какие операции определены для контейнера map?

Основные операции:

Вставка: insert(), emplace(), operator[].

Удаление: erase(), clear().

Поиск: find(), count(), contains() (C++20).

Доступ: at(), operator[].

Обход: итераторы (begin(), end()).

Размер: size(), empty().

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

#include <map>

#include <string>

#include <utility> // для make\_pair

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

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

}

int main() {

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

addToMap(myMap, "Dave", 40);

return 0;

}

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

#include <map>

#include <string>

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

m[key] = value;

}

int main() {

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

addToMap(myMap, "Eve", 45);

return 0;

}

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

#include <iostream>

#include <map>

#include <string>

void printMap(const std::map<std::string, int>& m) {

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

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

}

}

int main() {

std::map<std::string, int> myMap = {{"Alice", 25}, {"Bob", 30}};

printMap(myMap);

return 0;

}

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

Примечание: оператор [] не подходит для безопасного обхода, так как он создает элементы при отсутствии ключа. Лучше использовать at() или итераторы.

#include <iostream>

#include <map>

#include <string>

void printMap(const std::map<std::string, int>& m) {

for (const auto& key : {"Alice", "Bob"}) { // Только если ключи известны заранее

std::cout << key << ": " << m.at(key) << std::endl;

}

}

int main() {

std::map<std::string, int> myMap = {{"Alice", 25}, {"Bob", 30}};

printMap(myMap);

return 0;

}

Если нужно печатать все элементы, лучше использовать итераторы (как в вопросе 10).

1. Чем отличаются контейнеры map и multimap?

map:

1. Хранит уникальные ключи.
2. Каждому ключу соответствует ровно одно значение.
3. Пример: std::map<std::string, int> — у каждого имени (string) только одно значение (int).

multimap:

1. Допускает дублирование ключей.
2. Один ключ может быть связан с несколькими значениями.
3. Пример: std::multimap<std::string, int> — имя (string) может встречаться несколько раз с разными значениями (int).
4. Что представляет собой контейнер set?

set — это ассоциативный контейнер, хранящий уникальные элементы (ключи), автоматически упорядоченные по возрастанию. Не хранит пары "ключ-значение", только ключи. Используется, когда важно проверять наличие элементов и поддерживать их в отсортированном порядке.

1. Чем отличаются контейнеры map и set?

| Характеристика | map | set |
| --- | --- | --- |
| Хранение | Пары "ключ-значение" (pair) | Только ключи |
| Доступ к значению | Да (через ключ) | Нет (только проверка наличия ключа) |
| Пример | std::map<std::string, int> | std::set<int> |
| Использование | Словари, ассоциативные массивы | Уникальные коллекции |

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

Имеется три способа определить объект типа set:

set<int> set1; // создается пустое множество

int а[5] = { 1. 2. 3. 4, 5};

set<int> set2(a, а + 5); // инициализация копированием

set<int> set3(set2); // инициализация другим множеством

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

По умолчанию: по возрастанию (используется компаратор std::less<Key>).

Для обратного порядка: использовать std::greater<Key>.

#include <set>

#include <functional> // для greater

int main() {

std::set<int, std::greater<int>> reverseSet = {5, 2, 8}; // Упорядочится: 8 5 2

return 0;

}

1. Какие операции определены для контейнера set?

Основные методы:

Вставка: insert(), emplace().

Удаление: erase(), clear().

Поиск: find(), count(), contains() (C++20).

Обход: итераторы (begin(), end()).

Размер: size(), empty().

1. Написать функцию для добавления элементов в контейнер set.

#include <set>

void addToSet(std::set<int>& s, int value) {

s.insert(value);

}

int main() {

std::set<int> mySet = {1, 2, 3};

addToSet(mySet, 4); // Добавит 4

addToSet(mySet, 2); // Не добавит (дубликат)

return 0;

}

1. Написать функцию для печати контейнера set.

#include <iostream>

#include <set>

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

for (const auto& elem : s) {

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

}

std::cout << std::endl;

}

int main() {

std::set<int> mySet = {5, 1, 3};

printSet(mySet); // Вывод: 1 3 5

return 0;

}

1. Чем отличаются контейнеры set и multiset?

set:

1. Хранит уникальные элементы.
2. Попытка добавить дубликат игнорируется.

multiset:

1. Допускает дублирование элементов.
2. Полезен, когда нужно учитывать количество повторений (например, частотный анализ).