Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Ассоциативные контейнеры библиотеки STL

Вариант 5.

Выполнил работу:

Студент группы РИС-22-1Б

Батин В.В.

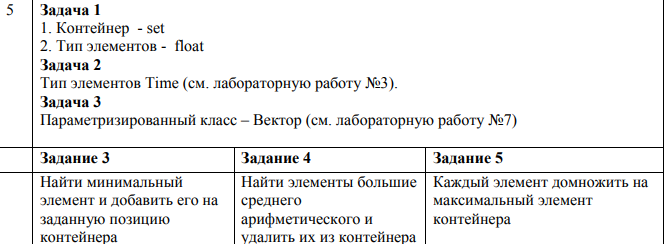
Проверил:

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Постановка задачи**

**

**Код программы**

**Задача 1**

#include <iostream>

#include <set>

#include <vector>

using namespace std;

set<float> mySet;

// Функция добавления элемента в заданную позицию

void addElement(float element, int position) {

auto it = mySet.begin();

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

it++;

}

mySet.insert(it, element);

}

// Функция нахождения среднего арифметического

float findAverage() {

float sum = 0;

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

sum += \*it;

}

return sum / mySet.size();

}

// Функция домножения каждого элемента на максимальный элемент контейнера

vector<float> multiplyElementsByMax() {

float maxElement = \*mySet.rbegin();

vector<float> result;

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

result.push\_back(\*it \* maxElement);

}

return result;

}

int main() {

// Добавление элементов в контейнер

mySet.insert(1.2);

mySet.insert(2.3);

mySet.insert(3.4);

mySet.insert(4.5);

// Вывод элементов контейнера до изменений

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

cout << \*it << " ";

}

cout << endl;

// Добавление минимального элемента в заданную позицию

float minElement = \*mySet.begin();

addElement(minElement, 2);

// Вывод элементов контейнера после добавления минимального элемента

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

cout << \*it << " ";

}

cout << endl;

// Удаление элементов, больших среднего арифметического

float average = findAverage();

auto it = mySet.begin();

while (it != mySet.end()) {

if (\*it > average) {

it = mySet.erase(it);

}

else {

it++;

}

}

// Вывод элементов контейнера после удаления элементов, больших среднего арифметического

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

cout << \*it << " ";

}

cout << endl;

// Домножение каждого элемента на максимальный элемент контейнера

vector<float> newSet = multiplyElementsByMax();

// Вывод элементов нового контейнера после домножения каждого элемента на максимальный элемент контейнера

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

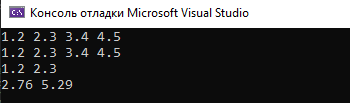
cout << \*it << " ";

}

cout << endl;

return 0;

}



**Задача 2**

#include <iostream>

#include <map>

#include <algorithm>

#include <numeric>

using namespace std;

class Time {

public:

Time() {}

Time(int h, int m) : hours(h), minutes(m) {}

int hours;

int minutes;

};

bool operator<(const Time& t1, const Time& t2) {

if (t1.hours < t2.hours) {

return true;

}

else if (t1.hours == t2.hours && t1.minutes < t2.minutes) {

return true;

}

return false;

}

void insertMinElement(map<Time, int>& m, Time minTime, Time posTime) {

auto it = m.find(minTime);

if (it != m.end()) {

m.erase(it);

m.insert(pair<Time, int>(posTime, 1));

}

}

void removeGreaterAverage(map<Time, int>& m) {

int totalMinutes = 0;

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

totalMinutes += (it->first.hours \* 60) + it->first.minutes;

}

double average = (double)totalMinutes / m.size();

auto it = m.begin();

while (it != m.end()) {

if (((it->first.hours \* 60) + it->first.minutes) > average) {

it = m.erase(it);

}

else {

it++;

}

}

}

int main() {

map<Time, int> m;

m.insert(pair<Time, int>(Time(10, 30), 1));

m.insert(pair<Time, int>(Time(9, 15), 2));

m.insert(pair<Time, int>(Time(14, 0), 3));

m.insert(pair<Time, int>(Time(11, 45), 4));

m.insert(pair<Time, int>(Time(8, 0), 5));

// Insert min element at position

Time minTime = (\*m.begin()).first;

Time posTime(13, 45);

insertMinElement(m, minTime, posTime);

// Remove elements greater than average

removeGreaterAverage(m);

// Print elements

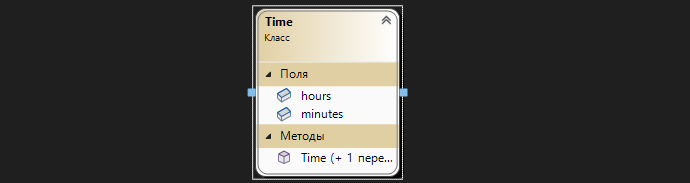
for (auto it = m.begin(); it != m.end(); it++) {

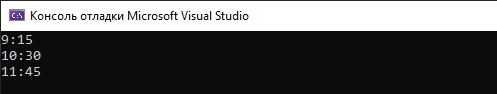
cout << it->first.hours << ":" << it->first.minutes << endl;

}

return 0;

}





**Задача 3**

#include <iostream>

#include <map>

#include <algorithm>

template <typename T, typename Container = std::map<int, T>>

class Vector {

private:

Container data\_;

public:

void addElement(int pos, const T& element) {

data\_[pos] = element;

}

void removeElement(int pos) {

auto it = data\_.find(pos);

if (it != data\_.end()) {

data\_.erase(it);

}

}

T getMinimumElement() const {

auto it = std::min\_element(data\_.begin(), data\_.end(),

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

return a.second < b.second;

});

if (it != data\_.end()) {

return it->second;

}

// Return a default-constructed value if the container is empty

return T();

}

double getArithmeticMean() const {

double sum = 0.0;

for (const auto& pair : data\_) {

sum += pair.second;

}

return sum / data\_.size();

}

void addMinimumToPosition(int pos) {

T minElement = getMinimumElement();

data\_[pos] = minElement;

}

void removeLargerThanMean() {

double mean = getArithmeticMean();

auto it = data\_.begin();

while (it != data\_.end()) {

if (it->second > mean) {

it = data\_.erase(it);

}

else {

++it;

}

}

}

void multiplyByMaximum() {

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

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

return a.second < b.second;

});

if (maxIt != data\_.end()) {

T maxElement = maxIt->second;

for (auto& pair : data\_) {

pair.second \*= maxElement;

}

}

}

void print() const {

for (const auto& pair : data\_) {

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

}

}

};

int main() {

Vector<int> vec;

vec.addElement(0, 5);

vec.addElement(1, 2);

vec.addElement(2, 7);

vec.addElement(3, 3);

vec.addElement(4, 1);

std::cout << "Initial vector:" << std::endl;

vec.print();

int minimum = vec.getMinimumElement();

std::cout << "Minimum element: " << minimum << std::endl;

vec.addMinimumToPosition(5);

std::cout << "Vector after adding minimum to position 5:" << std::endl;

vec.print();

vec.removeLargerThanMean();

std::cout << "Vector after removing elements larger than the mean:" << std::endl;

vec.print();

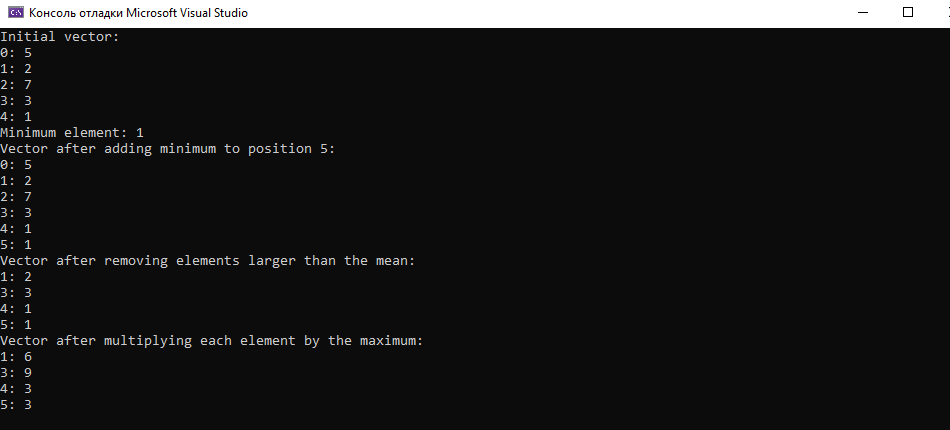
vec.multiplyByMaximum();

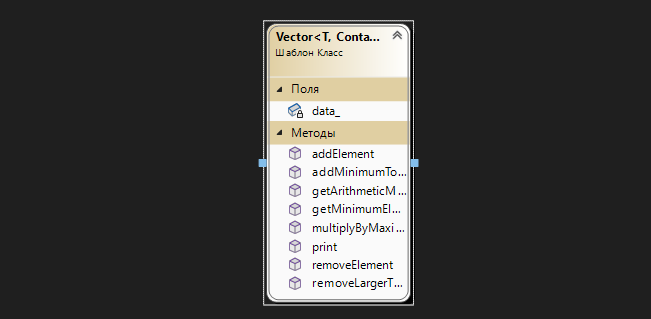
std::cout << "Vector after multiplying each element by the maximum:" << std::endl;

vec.print();

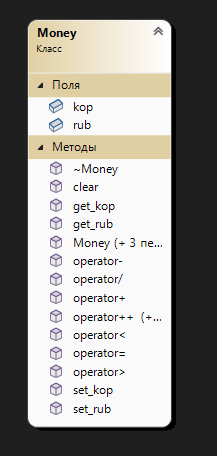
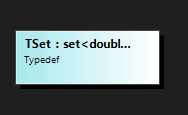
return 0;

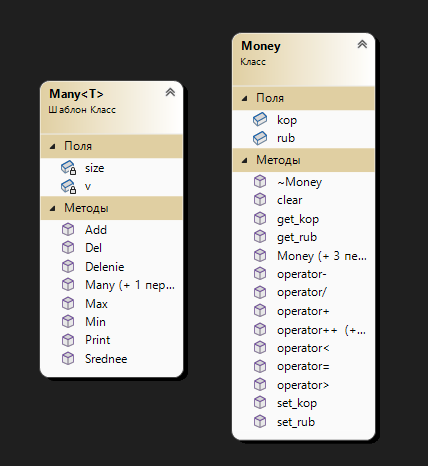
}

****



**UML диаграммы**

****



# Ответы на контрольные вопросы

1. Что представляет собой ассоциативный контейнер?

Ассоциативный массив содержит пары значений. Зная одно значение,

называемое ключом (key), мы можем получить доступ к другому, называемому отображаемым значением (mapped value).

Ассоциативный массив можно представить как массив, для которого индекс не обязательно должен иметь целочисленный тип.

2. Перечислить ассоциативные контейнеры библиотеки STL.

map - ассоциативный массив, по ключу в контейнере хранится одно значение

multimap - ассоциативный массив с повторяющимися ключами

set - массив уникальных ключей без значений

multiset - массив с повторяющимися ключами без значений.

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

V& operator[](const K&) возвращает ссылку на элемент V, соответствующий значению K.

4. Привести примеры методов, используемых в ассоциативных контейнерах.

bool empty() const, size\_type size() const, size\_type max\_size(), insert(), erase(), clear(), swap(), key\_comp(), value\_comp(), find(), count(), lower\_bound(), upper\_bound().

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

#include <iostream>

#include <map>

using namespace std;

int main() {

map<int, float> m;//словарь\

int n;//количество элементов

cout << "n: ";

cin >> n;

float a;

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

cout <<" ?";

cin >> a; //создаем пару и добавляем ее в словарь

m.insert(make\_pair(i, a)); }

return 0;

}

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

Ассоциативный контейнер map требует, чтобы для типов ключа существовала операция “<”, то есть, элементы хранятся в порядке возрастания. Он хранит свои элементы отсортированными по ключу так, что перебор происходит по порядку.

Спецификация шаблона для класса map:

template <class Key, class T, class Comp = less <Key>, class Allocator = allocator <pair> >

class Comp = less <Key> - параметр, определяющий критерий

упорядочения, по умолчанию less (по возрастанию ключа).

Чтобы изменить критерий упорядочивания, нужно изменить параметр Comp.

7. Какие операции определены для контейнера map?

Определена операция присваивания: map& operator=(const map&);

Определены следующие операции: ==, <, <=, !=, >, >= и операция индексации ([]).

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

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

cout <<" ?";

cin >> a;

m.insert(make\_pair(i, a));

}

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

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

cout <<" ?";

cin >> a;

m[i] = a;

}

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

for (map<int, float>::iterator i = m.begin(); i != m.end(); ++i) {

cout << (\*i).first << " " << (\*i).second << endl;

}

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

for (int i = 0; i < m.size(); ++i) {

cout << m[i] << endl;

}

12. Чем отличаются контейнеры map и multimap?

Словари с дубликатами (multimap) допускают хранение элементов с одинаковыми ключами. Поэтому для них не определена операция доступа по индексу. Элементы с одинаковыми ключами хранятся в словаре в порядке их занесения. При удалении по ключу функция erase возвращает количество удаленных элементов. В остальном они аналогичны обычным словарям.

13. Что представляет собой контейнер set?

set — это контейнер, который автоматически сортирует добавляемые элементы в порядке возрастания. Но при добавлении одинаковых значений, set будет хранить только один его экземпляр. По другому его еще называют множеством.

14. Чем отличаются контейнеры map и set?

Тип set позволяет хранить уникальные объекты различных типов, эффективно добавлять, удалять объекты и выполнять поиск. Тип map позволяет хранить пары ключ-значение, причем ключи должны быть уникальными.

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

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

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

Множество, как и словарь, требует, чтобы для типа T существовала операция “меньше” (<). Оно хранит свои элементы отсортированными, так что перебор происходит по порядку.

В множестве хранятся объекты, упорядоченные по некоторому ключу, являющемуся атрибутом самого объекта. Например, множество может хранить объекты класса Person, упорядоченные в алфавитном порядке по значению ключевого поля name. Если в множестве хранятся значения одного из встроенных типов, например int, то ключом является сам элемент.

set<int, greater<int>> set - изменение критерия упорядочения путем спецификации параметра компаратора

17. Какие операции определены для контейнера set?

Для вставки элементов в множество можно использовать метод insert(), для

удаления — метод erase(). Также к множествам применимы общие для всех контейнеров методы. Во всех ассоциативных контейнерах есть метод count(), возвращающий количество объектов с заданным ключом.

18. Написать функцию для добавления элементов в контейнер set.

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

cout <<" ? ";

cin >> a;

s.insert(a);

}

19. Написать функцию для печати контейнера set.

for(auto it = s.begin(); it != s.end(); ++i) cout<<\*i<<” ”;

20. Чем отличаются контейнеры set и multiset?

В множествах с дубликатами ключи могут повторяться. Элементы с одинаковыми ключами хранятся в множестве в порядке их занесения. Функция find() возвращает значение первого найденного элемента или end(), если ни одного элемента с заданным ключом не найдено.