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 6.

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**Постановка задачи**

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.

Условия варианта:

1. Контейнер - multiset;

2. Тип элементов - double.

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.

Условие варианта:

Тип элементов Pait (см. лабораторную работу №3).

Задача 3:

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

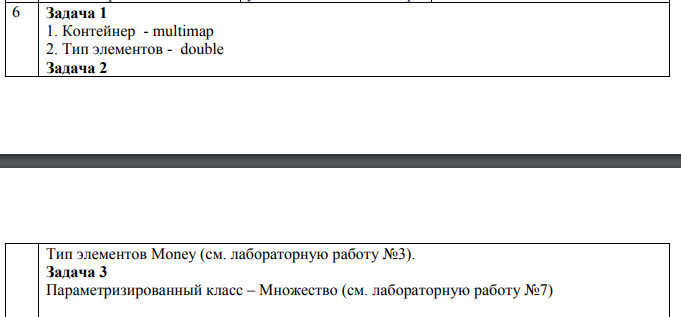
2. Заполнить его элементами.

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.



**Код программы**

**Задача 1**

#include <iostream>

#include <set>

using namespace std;

typedef set<double> TSet;

set<double>::iterator it;

TSet make\_set(int n)

{

TSet m;

double a;

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

{

cout << "Number: ";

cin >> a;

m.insert(a);

}

return m;

}

void print\_set(TSet m)

{

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

cout << \*it << " ";

}

TSet erase\_set(TSet m, double el)

{

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

{

double n = \*it;

if (el == n)

{

cout << \*it << endl;

m.erase(it++);

}

else

++it;

}

return m;

}

double Min\_set(TSet m)

{

double min, temp;

it = m.begin();

min = \*it;

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

{

temp = \*it;

if (min > temp)

min = temp;

}

return min;

}

double Max\_set(TSet m)

{

double max, temp;

it = m.begin();

max = \*it;

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

{

temp = \*it;

if (max < temp)

max = temp;

}

return max;

}

TSet zadanie(TSet m, double min, double max)

{

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

{

double h = \*it + min - max;

cout << h << " ";

}

return m;

}

void main()

{

int n;

cout << "Number: ";

cin >> n;

TSet m = make\_set(n);

print\_set(m);

double d;

cout << "Whats to remove: ";

cin >> d;

TSet j = erase\_set(m, d);

print\_set(j);

double h = Min\_set(j);

double u;

cout << "Add: " << endl;

cin >> u;

j.insert(u);

print\_set(j);

cout << endl;

double g = Max\_set(j);

cout << "Add max and min: " << endl;

zadanie(j, h, g);

}

**Задача 2**

**#include** <iostream>

#include <map>

#include "Money.h"

using namespace std;

typedef map<int, Money> TMap;

typedef TMap::iterator it;

TMap make\_map(int n)

{

TMap m;

Money a;

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

{

cin >> a;

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

}

return m;

}

void print\_map(TMap m)

{

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

cout << i << " - " << m[i] << " " << endl;

}

Money middle(TMap m)

{

Money s = m[0];

for (int i = 1; i < m.size(); i++)

s = s + m[i];

return s / m.size();

}

int Max(TMap v)

{

it i = v.begin(); int nom = 0, k = 0;

Money m = (\*i).second;

while (i != v.end())

{

if (m < (\*i).second)

{

m = (\*i).second;

nom = k;

}

i++;

k++;

}

return nom;

}

int Min(TMap v)

{

it i = v.begin();

int nom = 0, k = 0;

Money m = (\*i).second;

while (i != v.end())

{

if (m > (\*i).second)

{

m = (\*i).second;

nom = k;

}

i++;

k++;

}

return nom;

}

void delenie(TMap& v)

{

Money m = v[Min(v)];

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

v[i] = v[i] / m;

}

void main()

{

int n;

cout << "Number: ";

cin >> n;

map<int, Money> m = make\_map(n);

print\_map(m);

Money el = middle(m);

cout << "Middle: " << el << endl;

m.insert(make\_pair(n, el));

print\_map(m);

int max = Max(m);

cout << "max=" << m[max] << " nom=" << max << endl;

m.erase(max);

print\_map(m);

int min = Min(m);

cout << "min=" << m[min] << " nom=" << min << endl;

delenie(m);

print\_map(m);

}

**Задача 3**

**#include** <iostream>

#include "Many.h"

#include "Money.h"

using namespace std;

void main()

{

int n;

cout << "Number: ";

cin >> n;

Many <Money> v(n);

v.Print();

Money t = v.Srednee();

cout << "Middle: " << t << endl;

cout << "Adding Middle" << endl;

cout << "Pos: ";

int pos;

cin >> pos;

v.Add(pos, t);

v.Print();

cout << "Delete max: " << endl;

v.Del();

v.Print();

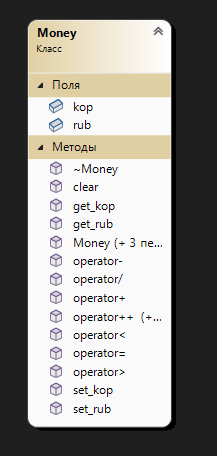
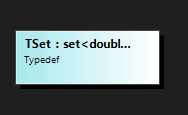
cout << "Division on min: " << endl;

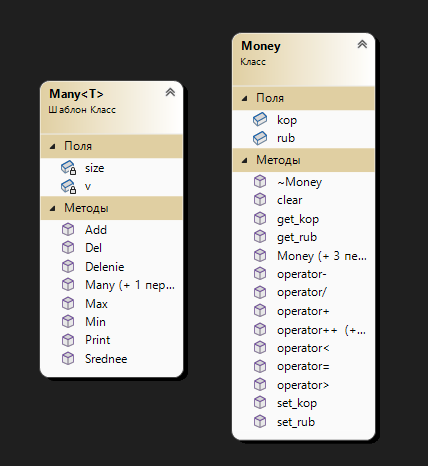
v.Delenie();

v.Print();

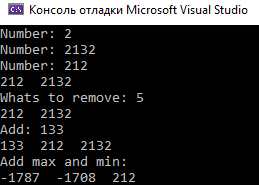
}

**UML диаграммы**

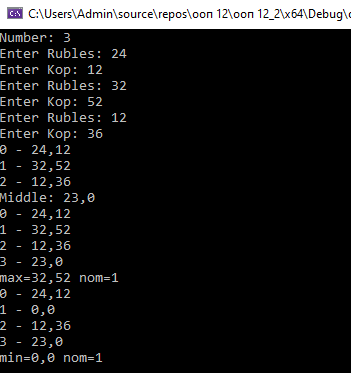
****



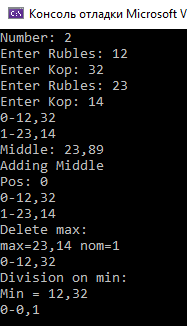
**Вывод программ**



вывод программы в задании под номером 1.



вывод программы в задании под номером 2.



вывод программы в задании под номером 3.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.

# 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.