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**

Дисциплина: «Информатика»

ТемаАссоциативные контейнеры библиотеки STL

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Протасов Н.С.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановказадачи(общаяиконкретноговарианта).**

### Задача 1.

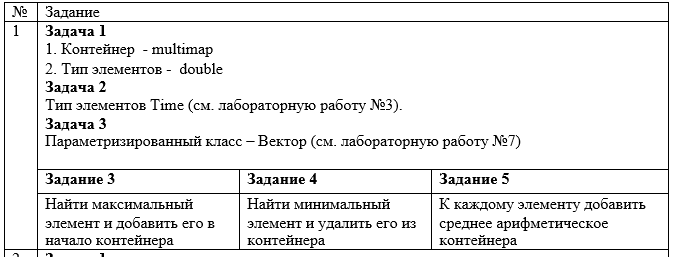
* + 1. Создатьассоциативныйконтейнер.
    2. Заполнитьегоэлементамистандартноготипа(типуказанвварианте).
    3. Добавитьэлементывсоответствиисзаданием
    4. Удалитьэлементывсоответствиисзаданием.
    5. Выполнитьзаданиевариантадляполученногоконтейнера.
    6. Выполнениевсехзаданийоформитьввидеглобальныхфункций.

### Задача 2.

1. Создать ассоциативныйконтейнер.
2. Заполнитьегоэлементамипользовательскоготипа(типуказанвварианте).Дляпользовательскоготипаперегрузитьнеобходимые операции.
3. Добавитьэлементывсоответствиисзаданием
4. Удалитьэлементывсоответствиисзаданием.
5. Выполнитьзаданиевариантадляполученного контейнера.
6. Выполнениевсехзаданийоформитьввидеглобальныхфункций.

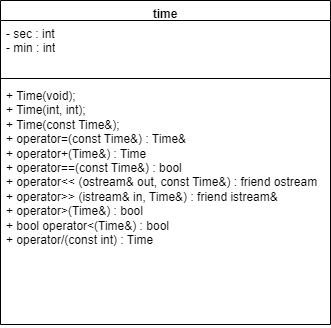
### Задача 3

1. Создать параметризированный класс, используя в качестве контейнераассоциативныйконтейнер.
2. Заполнитьегоэлементами.
3. Добавитьэлементывсоответствиисзаданием
4. Удалитьэлементывсоответствиисзаданием.
5. Выполнитьзаданиевариантадляполученногоконтейнера.
6. Выполнениевсехзаданийоформитьввидеметодовпараметризированногокласса.

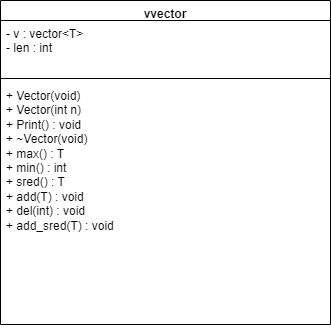


**UML**

**2**

****

**3**

****

**Функции для решения задачи 1**

#pragma once

#include <iostream>

#include <map>

#include <numeric>

using namespace std;

typedef std::multimap<int, double> Tmap;

typedef Tmap::iterator it;

Tmap createAndFillMultimap(int n)

{

Tmap container;

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

{

double a;

cout << i << " : ";

cin >> a;

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

}

return container;

}

double average(Tmap v)

{

double s = 0.0;

int n = v.size();

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

{

s += it->second;

}

return s / n;

}

int Max(Tmap v)

{

it i = v.begin();

int nom = 0;

int k = 0;

double 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;

int k = 0;

double m = i->second;

while (i != v.end())

{

if (m > i->second)

{

m = i->second;

nom = k;

}

++i;

++k;

}

return nom;

}

void del(Tmap& container)

{

int min = Min(container);

auto it = container.begin();

std::advance(it, min);

container.erase(it);

}

void add(Tmap& container)

{

int max = Max(container);

it maxIterator = container.begin();

std::advance(maxIterator, max);

auto maxElement = \*maxIterator;

container.insert(container.begin(), maxElement);

}

void addAver(Tmap& container)

{

double sred = average(container);

for (auto& elem : container)

{

elem.second += sred;

}

}

void printMultimap(const Tmap& container)

{

for (const auto& element : container)

{

cout << "first: " << element.first << ", second: " << element.second << endl;

}

}

void Execute1()

{

int n;

cout << "n = ";

cin >> n;

Tmap container = createAndFillMultimap(n);

printMultimap(container);

cout << endl;

add(container);

printMultimap(container);

cout << endl;

del(container);

printMultimap(container);

cout << endl;

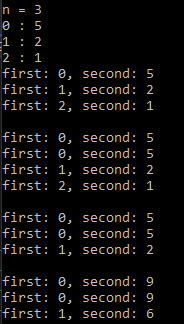
addAver(container);

printMultimap(container);

cout << endl;

}

**Результаты работы программы.**

****

**Описание пользовательского класса для решения задачи 2**

#pragma once

#include <iostream>

using namespace std;

class Time

{

private:

int min, sec;

public:

Time(int m = 0, int s = 0) { min = m; sec = s; }

Time(const Time& t) { min = t.min;sec = t.sec; }

~Time() {};

int get\_min() { return min; }

int get\_sec() { return sec; }

void set\_min(int m) { min = m; }

void set\_sec(int s) { sec = s; }

int toNumber() { return min \* 60 + sec; }

Time operator+(const Time& t)

{

int temp1 = toNumber();

int temp2 = t.min \* 60 + t.sec;

Time p;

p.min = (temp1 + temp2) / 60;

p.sec = (temp1 + temp2) % 60;

return p;

}

bool operator>(const Time& t)

{

if (min > t.min || (min == t.min && sec > t.sec)) return true;

return false;

}

bool operator==(const Time& t)

{

return t.min == min && t.sec == sec;

}

bool operator<(const Time& t)

{

if (\*this > t || \*this == t) return false;

return true;

}

Time operator/(int n)

{

int s = toNumber() / n;

return Time(s / 60, s % 60);

}

friend istream& operator>>(istream& in, Time& t)

{

cout << "minutes = ";

in >> t.min;

cout << "seconds = ";

in >> t.sec;

return in;

}

friend ostream& operator<<(ostream& out, const Time& t)

{

return (out << t.min << ":" << t.sec);

}

};

**Функции для решения задачи 2**

#pragma once;

#include <iostream>

#include <map>

#include "Time.h"

using namespace std;

typedef std::multimap<int, Time> Tmap2;

typedef Tmap2::iterator it2;

Tmap2 createAndFillMultimap2(int n)

{

Tmap2 container;

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

{

Time a;

cin >> a;

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

}

return container;

}

Time Average(Tmap2 v)

{

Time s;

int n = v.size();

for (it2 it = v.begin(); it != v.end(); ++it)

{

s = s + it->second;

}

return s / n;

}

int Max(Tmap2 v)

{

it2 i = v.begin();

int pos = 0;

int k = 0;

Time m = i->second;

while (i != v.end())

{

if (m < i->second)

{

m = i->second;

pos = k;

}

++i;

++k;

}

return pos;

}

int Min(Tmap2 v)

{

it2 i = v.begin();

int pos = 0;

int k = 0;

Time m = i->second;

while (i != v.end())

{

if (m > i->second)

{

m = i->second;

pos = k;

}

++i;

++k;

}

return pos;

}

void del(Tmap2& container)

{

int min = Min(container);

auto it = container.begin();

std::advance(it, min);

container.erase(it);

}

void add(Tmap2& container)

{

int max = Max(container);

it2 maxIterator = container.begin();

std::advance(maxIterator, max);

auto& maxElement = \*maxIterator;

container.insert(container.begin()--, maxElement);

}

void add\_avrg(Tmap2& container)

{

Time avrg = Average(container);

for (auto& elem : container)

{

elem.second = elem.second + avrg;

}

}

void printMultimap(const Tmap2& container)

{

for (const auto& element : container)

{

cout << "First: " << element.first << ", second: " << element.second << endl;

}

}

void Execute2()

{

int n;

cout << "n = ";

cin >> n;

Tmap2 container = createAndFillMultimap2(n);

printMultimap(container);

cout << endl;

add(container);

printMultimap(container);

cout << endl;

del(container);

printMultimap(container);

cout << endl;

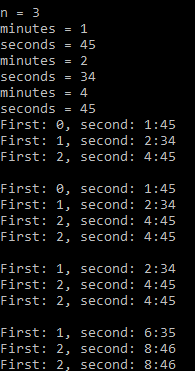
add\_avrg(container);

printMultimap(container);

cout << endl;

}

**Результаты работы программы**

****

**Описание параметризированного класса для решения задачи 3**

#pragma once

#include <iostream>

using namespace std;

template <class T>

class Vector

{

public:

Vector(int s);

Vector(int s, T k);

Vector(const Vector<T>& a);

int Size()

{

return size;

}

void erase(int pos)

{

auto temp = new T[size - 1];

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

{

temp[i] = data[i];

}

for (int i = pos + 1; i < size; i++)

{

temp[i - 1] = data[i];

}

data = temp;

size--;

}

void insert(int pos, T elem)

{

auto temp = new T[size + 1];

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

{

temp[i] = data[i];

}

for (int i = pos; i < size; i++)

{

temp[i + 1] = data[i];

}

temp[pos] = elem;

data = temp;

size++;

}

Vector& operator=(const Vector<T>& a);

T& operator[](int index);

Vector operator+(const T k);

int Max();

int Min();

T Average();

void Print();

friend ostream& operator<< <>(ostream& out, Vector<T>& a);

friend istream& operator>> <>(istream& in, Vector<T>& a);

private:

int size;

T\* data;

};

template <class T>

Vector<T>::Vector(int s)

{

size = s;

data = new T[size];

}

template <class T>

Vector<T>::Vector(int s, T k)

{

size = s;

data = new T[size];

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

{

data[i] = k;

}

}

template <class T>

Vector<T>::Vector(const Vector& a)

{

size = a.size;

data = new T[size];

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

{

data[i] = a.data[i];

}

}

template <class T>

Vector<T>& Vector<T>::operator=(const Vector<T>& a)

{

if (this == &a) return \*this;

size = a.size;

if (data != 0) delete[] data;

data = new T[size];

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

{

data[i] = a.data[i];

}

return \*this;

}

template <class T>

T& Vector<T>::operator[](int index)

{

if (index < size) return data[index];

else cout << "Error! Index > size\n";

}

template <class T>

Vector<T> Vector<T>::operator+(const T k)

{

Vector<T> temp(\*this);

for (int i = 0; i < size; ++i) temp.data[i] = data[i] + k;

return temp;

}

template <class T>

int Vector<T>::Min()

{

T temp = data[0];

int n = 0;

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

{

if (data[i] < temp)

{

temp = data[i];

n = i;

}

}

return n;

}

template <class T>

int Vector<T>::Max()

{

T temp = data[0];

int n = 0;

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

{

if (data[i] > temp)

{

temp = data[i];

n = i;

}

}

return n;

}

template <class T>

T Vector<T>::Average()

{

T s = data[0];

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

{

s = s + data[i];

}

return s / size;

}

template <class T>

void Vector<T>::Print()

{

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

{

cout << data[i] << endl;

}

}

template <class T>

ostream& operator<<(ostream& out, Vector<T>& a)

{

for (int i = 0; i < a(); ++i) out << a.data[i] << " ";

out << "\n";

return out;

}

template <class T>

istream& operator>>(istream& in, Vector<T>& a)

{

for (int i = 0; i < a(); ++i) in >> a.data[i];

return in;

}

**Функции для решения задачи 3**

#pragma once

#include <iostream>

#include <map>

#include "Time.h"

#include "Vector.h"

using namespace std;

void Execute3()

{

int n;

cout << "n = ";

cin >> n;

Vector<Time> container(n);

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

{

Time t;

cin >> t;

container[i] = t;

}

container.Print();

cout << endl;

int pos = container.Max();

container.insert(0, container[pos]);

container.Print();

cout << endl;

int b = container.Min();

container.erase(b);

container.Print();

cout << endl;

Time c = container.Average();

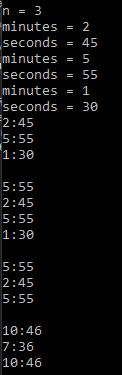
container = container + c;

container.Print();

cout << endl;

}

**Результаты работы программы**



**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. Привестипримерыметодов, используемыхвассоциативныхконтейнерах.

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? Привестипримеры.

#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.