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: «Основы алгоритмизации и программирования»

Тема: «Ассоциативные контейнеры библиотеки STL»

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Желнин Н.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

Постановка задачи

Вариант 14

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.

Указания варианта:

1. Контейнер - multimap; 2. Тип элементов - float

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.

Указание варианта:

Тип элементов Pair (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.

Указание варианта:

Параметризированный класс – Список (см. лабораторную работу №7).

Дополнение варианта ко всем задачам:

1. Найти минимальный элемент и добавить его на заданную позицию контейнера;

2. Найти меньше среднего арифметического и удалить их из контейнера;

3. Каждый элемент разделить на максимальный элемент контейнера.

Код программы

Задание 1

#include <iostream>

#include <map>

using namespace std;

int main(){

multimap<int, float> data;

multimap<int, float>::iterator itr;

int key;

float value;

//enter elements

int n;

cout << "Enter count: ";

cin >> n;

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

cout << "Key: "; cin >> key;

cout << "Value: "; cin >> value;

data.insert({key, value});

}

//print elements

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " : " << itr->second << endl;

}

//find min

float min1 = 10000.0;

for (itr = data.begin(); itr != data.end(); ++itr) {

if(itr->second < min1) min1 = itr->second;

}

//insert min

int key1;

cout << "Enter key: "; cin >> key1;

data.insert({key1, min1});

//print elements

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " : " << itr->second << endl;

}

//udalit' < srednego

float sr;

int k = 0;

float sum = 0;

for (itr = data.begin(); itr != data.end(); ++itr) {

k++;

sum += itr->second;

}

sr = sum / k;

cout << "Srednee: " << sr;

cout << endl;

for (itr = data.begin(); itr != data.end();) {

if(itr->second < sr){

itr = data.erase(itr);

}

else ++itr;

}

//print elements

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " : " << itr->second << endl;

}

//find max and delenie

float max1 = -10000.0;

for (itr = data.begin(); itr != data.end(); ++itr) {

if(itr->second > max1) max1 = itr->second;

}

for (itr = data.begin(); itr != data.end(); ++itr) {

itr->second = itr->second / max1;

}

cout << "Posle deleniya na max:" << endl;

//print elements

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " : " << itr->second << endl;

}

}

Задание 2

//MAIN

#include <iostream>

#include "pair.h"

#include <map>

using namespace std;

int main(){

multimap<int, Pair> data;

multimap<int, Pair>::iterator itr;

int key;

int value1;

double value2;

//enter elements

int n;

cout << "Enter count: ";

cin >> n;

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

cout << "Key: "; cin >> key;

cout << "Value1: "; cin >> value1;

cout << "Value2: "; cin >> value2;

Pair a(value1, value2);

data.insert({key, a});

}

//print elements

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " :: ";

cout << itr->second.first << " : " << itr->second.second;

cout << endl;

}

//find min

Pair min1(10000, 10000.0);

for (itr = data.begin(); itr != data.end(); ++itr) {

if(itr->second < min1) min1 = itr->second;

}

//insert min

int key1;

cout << "Enter key: "; cin >> key1;

data.insert({key1, min1});

//print elements

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " :: ";

cout << itr->second.first << " : " << itr->second.second;

cout << endl;

}

//udalit' < srednego

int sr1; double sr2;

int k = 0;

int sum1 = 0; double sum2 = 0;

for (itr = data.begin(); itr != data.end(); ++itr) {

k++;

sum1 = sum1 + itr->second.first;

sum2 = sum2 + itr->second.second;

}

sr1 = sum1 / k; sr2 = sum2 / k;

Pair sr(sr1, sr2);

for (itr = data.begin(); itr != data.end();) {

if(itr->second < sr){

itr = data.erase(itr);

}

else ++itr;

}

//print elements

cout << "After delete: " << endl;

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " :: ";

cout << itr->second.first << " : " << itr->second.second;

cout << endl;

}

//find max and delenie

Pair max1(-10000, -10000.0);

for (itr = data.begin(); itr != data.end(); ++itr) {

if(itr->second > max1) max1 = itr->second;

}

for (itr = data.begin(); itr != data.end(); ++itr) {

itr->second.first = itr->second.first / max1.first;

itr->second.second = itr->second.second / max1.second;

}

cout << "Posle deleniya na max:" << endl;

//print elements

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << itr->first << " :: ";

cout << itr->second.first << " : " << itr->second.second;

cout << endl;

}

}

//PAIR

#include <iostream>

using namespace std;

class Pair {

public:

int first;

double second;

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

Pair& operator /(const int X);

};

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

Pair Pair::operator =(const Pair& p) {

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

first = p.first;

second = p.second;

return\*this;

}

bool Pair::operator <(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

bool Pair::operator >(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

Задание 3

#include <iostream>

#include "pair.h"

#include "list.h"

#include <map>

using namespace std;

int main(){

int n;

cin >> n;

List<Pair> data(n);

data.Print();

data.del\_sr();

cout << "After delete: " << endl;

data.Print();

data.add\_min();

cout << "After add: " << endl;

data.Print();

data.delenie();

cout << "After delenie: " << endl;

data.Print();

}

#include <iostream>

#pragma once

using namespace std;

class Pair {

public:

int first;

double second;

Pair() { first = 0; second = 0; };

Pair(int F, double S) { first = F; second = S; };

Pair(const Pair& t) { first = t.first;second = t.second; };

~Pair() {};

Pair operator =(const Pair&);

bool operator <(const Pair&) const;

bool operator >(const Pair&) const;

Pair& operator /(const int X);

};

Pair& Pair::operator /(const int X) {

first = first / X;

second = second / X;

return \*this;

}

Pair Pair::operator =(const Pair& p) {

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

first = p.first;

second = p.second;

return\*this;

}

bool Pair::operator <(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis < sother) return true;

return false;

}

bool Pair::operator >(const Pair& p) const {

double sthis = this->first + this->second;

double sother = p.first + p.second;

if (sthis > sother) return true;

return false;

}

#pragma once

#include <map>

#include <iostream>

using namespace std;

template <class T>

class List {

multimap <int, T> data;

multimap<int, Pair>::iterator itr;

int size;

public:

List(void);

List(int n);

void Print();

~List(void);

void del\_sr();

void add\_min();

void delenie();

};

template <class T>

List<T>::List() {

size = 0;

}

template <class T>

List<T>::~List() {}

template <class T>

List<T>::List(int n) {

T a;

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

cin >> a.first;

cin >> a.second;

data.insert({i, a});

}

size = data.size();

}

template <class T>

void List<T>::Print() {

int k = 1;

for (itr = data.begin(); itr != data.end(); ++itr) {

cout << k++ << " :: ";

cout << itr->second.first;

cout << " : ";

cout << itr->second.second;

cout << endl;

}

}

template <class T>

void List<T>::del\_sr() {

int sr1; double sr2;

int k = 0;

int sum1 = 0; double sum2 = 0;

for (itr = data.begin(); itr != data.end(); ++itr) {

k++;

sum1 = sum1 + itr->second.first;

sum2 = sum2 + itr->second.second;

}

sr1 = sum1 / k; sr2 = sum2 / k;

Pair sr(sr1, sr2);

for (itr = data.begin(); itr != data.end();) {

if(itr->second < sr){

itr = data.erase(itr);

}

else ++itr;

}

}

template <class T>

void List<T>::add\_min() {

//find min

int k = 1;

Pair min1(10000, 10000.0);

for (itr = data.begin(); itr != data.end(); ++itr) {

if(itr->second < min1) min1 = itr->second;

k++;

}

//insert min

data.insert({k+1, min1});

}

template <class T>

void List<T>::delenie() {

//find max and delenie

Pair max1(-10000, -10000.0);

for (itr = data.begin(); itr != data.end(); ++itr) {

if(itr->second > max1) max1 = itr->second;

}

for (itr = data.begin(); itr != data.end(); ++itr) {

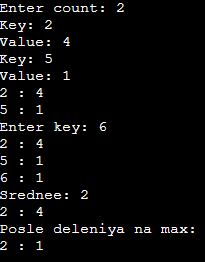
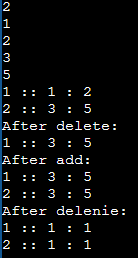
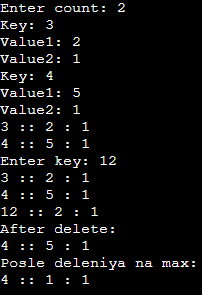
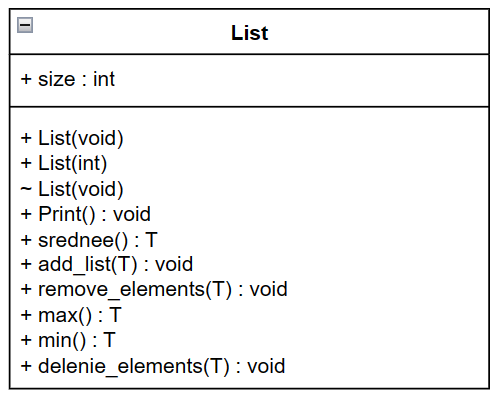
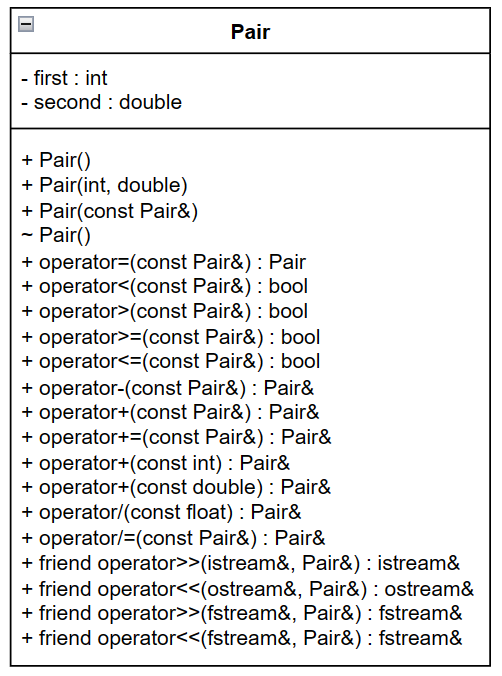
itr->second.first = itr->second.first / max1.first;

itr->second.second = itr->second.second / max1.second;

}

}

Диаграмма и вывод в консоль



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.