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

Токарев Павел Аркадьевич

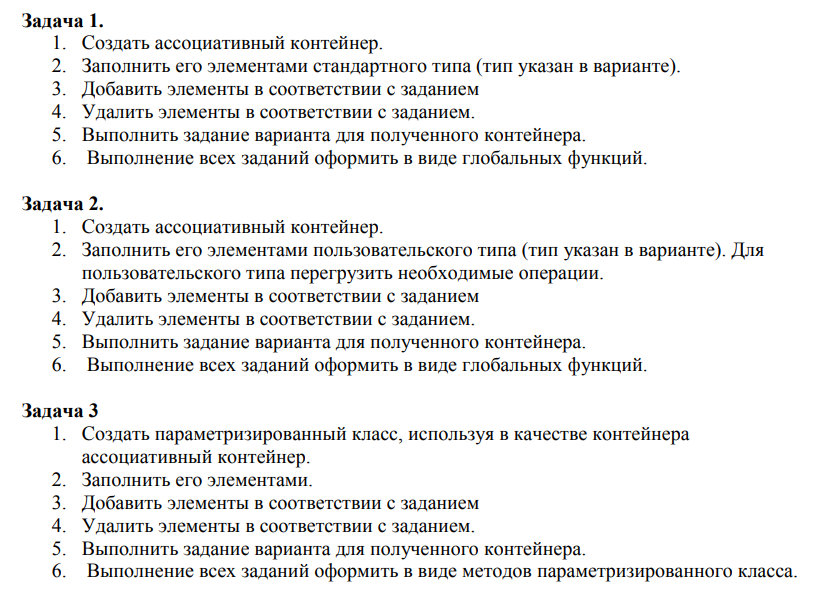
Проверил

Доцент кафедры ИТАС

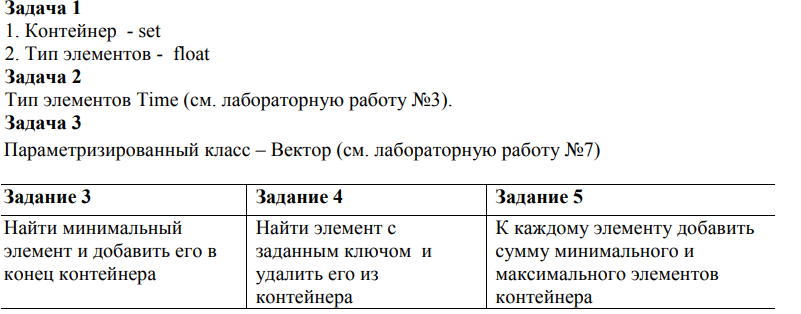
Полякова Ольга Андреевна

Г. Пермь-2023

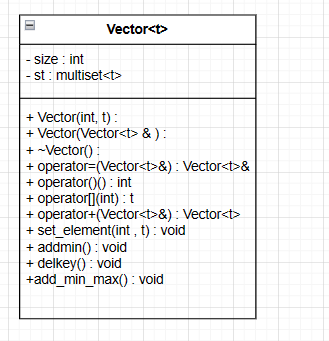
**Постановка задачи**

****

Вариант 2:



**Диаграмма UML**

****

**Код программы**

task1.h

#include<iostream>

#include<set>

#include<queue>

using namespace std;

typedef set<float> set1;

void makeset(int n,set1&a) {

float tmp;

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

cout << i + 1 << " : ";

cin >> tmp;

a.insert(tmp);

}

}

void show(set1& a) {

for (auto i = a.begin();i != a.end();i++) {

cout << \*i << " ";

}

cout << endl;

}

void findmin(set1&a){

float min = 100000.0;

for(auto i=a.begin();i!=a.end();i++){

if (\*i < min) {

min = \*i;

}

}

a.insert(min);

}

void findkey(set1 & a,float key) {

a.erase(key);

}

void add\_min\_max(set1& a) {

float min = \*a.begin();

float max = \*a.begin();

for (auto& i : a) {

if (i > max) {

max = i;

}

if (i < min) {

min = i;

}

}

set1 tmp;

for (auto& i : a) {

tmp.insert(i + max + min);

}

a = tmp;

}

void task1() {

set<float> set2;

cout << "Введите количество элементов для добавления в множество: ";

int n;

cin >> n;

makeset(n, set2);

cout << "Добавим минимальный элемент в множество" << endl;

findmin(set2);

show(set2);

cout << "Удалим элемент из множества по ключу" << endl;

float tmp;

cout << "Введите ключ : ";

cin >> tmp;

findkey(set2,tmp);

show(set2);

cout << "Добавим к всем эллементам сумму мин и макс элементов" << endl;

add\_min\_max(set2);

show(set2);

}

task2.h

#include<iostream>

#include<set>

#include"Time.h"

using namespace std;

typedef set<Time> set2;

void makeset(int n, set2& a) {

Time tmp;

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

cout << i + 1 << " : ";

cin >> tmp;

a.insert(tmp);

}

}

void show(set2& a) {

for (auto i = a.begin();i != a.end();i++) {

cout << \*i << " ";

}

cout << endl;

}

void findmin(set2& a) {

Time min(100000, 10000);

for (auto& i : a) {

Time tmp = i;

int lft = tmp.get\_minute() \* 60 + tmp.get\_second();

int right = min.get\_minute() \* 60 + min.get\_second();

if (lft < right) {

min = tmp;

}

}

a.insert(min);

}

void findkey(set2& a, Time key) {

a.erase(key);

}

void add\_min\_max(set2& a) {

Time min = \*a.begin();

Time max = \*a.begin();

for (auto& i : a) {

Time tmp=i;

int lft = tmp.get\_minute() \* 60 + tmp.get\_second();

int right = min.get\_minute() \* 60 + min.get\_second();

int rght = max.get\_minute() \* 60 + max.get\_second();

if (lft > rght) {

max = i;

}

if (lft < right) {

min = i;

}

}

set2 tmp1;

for (auto& i : a) {

Time tmp = i;

tmp1.insert(tmp + max + min);

}

a = tmp1;

}

void task2() {

set<Time> set2;

cout << "Введите количество элементов для добавления в множество: ";

int n;

cin >> n;

makeset(n, set2);

cout << "Добавим минимальный элемент в множество" << endl;

findmin(set2);

show(set2);

cout << "Удалим элемент из множества по ключу" << endl;

Time tmp;

cout << "Введите ключ : ";

cin >> tmp;

findkey(set2, tmp);

show(set2);

cout << "Добавим к всем эллементам сумму мин и макс элементов" << endl;

add\_min\_max(set2);

show(set2);

}

task3.h

#pragma once

#include <iostream>

#include<set>

#include"Time.h"

using namespace std;

template<typename t>

class Vector {

private:

multiset<t> st;

int size;

public:

Vector();

Vector(int n);

~Vector();

void make\_vector(int sz);

void show();

void addmin();

void popkey(t key);

void add\_min\_max();

};

template<typename t>

Vector<t>::Vector() {

size = 0;

}

template <typename t>

Vector<t>::Vector(int n) {

size = n;

}

template<typename t>

Vector<t>::~Vector() {

}

template <typename t>

void Vector<t>::make\_vector(int sz) {

t a;

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

cout << i + 1 << " : ";

cin >> a;

st.insert(a);

}

}

template <typename t>

void Vector<t>::show() {

for (auto& i : st) {

cout << i << " ";

}

cout << endl;

}

template <typename t>

void Vector<t>::addmin() {

st.insert(\*st.begin());

}

template <typename t>

void Vector<t>::popkey(t key) {

bool fl=false;

multiset<t> ns;

for (auto& i : st) {

t tmp = i;

if (tmp == key) {

if (fl == false) {

fl = true;

}

else {

ns.insert(tmp);

}

}

else {

ns.insert(tmp);

}

}

st = ns;

}

template<typename t>

void Vector<t>::add\_min\_max() {

t min = \*st.begin();

t max = \*st.begin();

for (auto& i : st) {

t tmp = i;

int lft = tmp.get\_minute() \* 60 + tmp.get\_second();

int right = min.get\_minute() \* 60 + min.get\_second();

int rght = max.get\_minute() \* 60 + max.get\_second();

if (lft > rght) {

max = i;

}

if (lft < right) {

min = i;

}

}

multiset<t> alf;

for (auto& i : st) {

t tmp = i;

alf.insert(tmp + max + min);

}

st = alf;

}

void task3() {

Vector<Time> vec;

cout << "Введите количество элементов для добавления в вектор: ";

int n;

cin >> n;

vec.make\_vector(n);

cout << "Добавим минимальный элемент в вектор" << endl;

vec.addmin();

vec.show();

cout << "Удалим элемент из вектора по ключу" << endl;

Time tmp;

cout << "Введите ключ : "<<endl;

cin >> tmp;

vec.popkey(tmp);

vec.show();

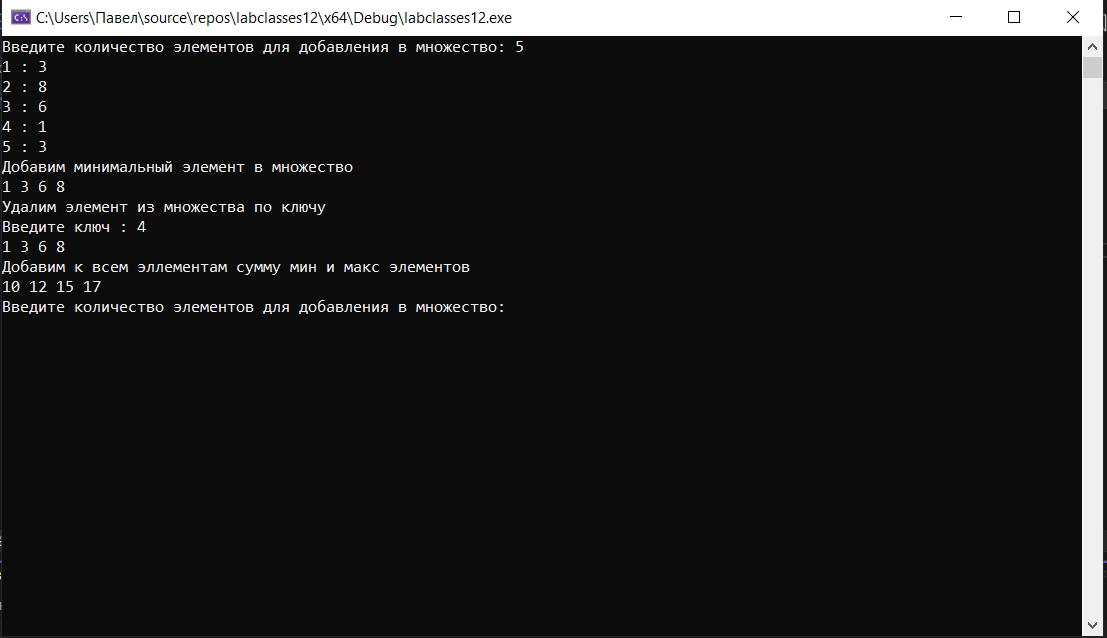
cout << "Добавим к всем эллементам сумму мин и макс элементов" << endl;

vec.add\_min\_max();

vec.show();

}

**Результат работы программы**

****

**Анализ результатов**

Удалось успешно реализовать все ассоциативные контейнеры. Для каждой реализации успешно выполняются операции добавления минимального элемента в конец контейнера, удаление элемента по ключу и увеличение всех элементов на сумму минимального и максимального элемента.

**Контрольные вопросы**

1.Что представляет собой ассоциативный контейнер?

Это обобщение понятия ассоциативного массива.

2.Перечислить ассоциативные контейнеры библиотеки STL.

container s0;//пустой контейнер

container s1 (s0);//копирование

container s2 (begin, end);//инициализация диапазоном

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

Доступ к элементам контейнера осуществляется по ключу.

4.Привести примеры методов, используемых в ассоциативных контейнерах.

insert(), erase(), clear(), swap(), find(), count(), etc.

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

Ассоциативный контейнер map требует, чтобы для типов ключа существовала операция “,class Allocator = allocator > class std::map {. . . }; class Key – параметр ключ, class T – параметр, определяющий тип значений, class Comp = less - параметр, определяющий критерий упорядочения, по умолчанию less (по возрастанию ключа).

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

Он хранит свои элементы отсортированными по ключу так, что перебор происходит по порядку.

Идти с конца. V.end(), V.begin().

7.Какие операции определены для контейнера map?

Определены следующие операции: ==, <=, !=, >, >=.

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

void add() {

map<int, float> tmp;

auto p1 = std::make\_pair(5, 7.8);

map.insert(p1);

}

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

void add() {

map<int, float> tmp;

tmp[5] = 8.4;

}

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

void add() {

for (int i = 0; it != mp.end(); it++, i++) {

cout << i << ") Ключ " << it->first << ", значение " << it->second << endl;}

}

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

void add() {

for (int i = 0;i < 10;i++) {

cout << i << " " << tmp[i];

}

}

12.Чем отличаются контейнеры map и multimap?

map - ассоциативный массив, по ключу в контейнере хранится одно значение multimap - ассоциативный массив с повторяющимися ключами

13.Что представляет собой контейнер set?

Множества set можно рассматривать как ассоциативные массивы, в которых значения не играют роли, так что мы отслеживаем только ключи.

14.Чем отличаются контейнеры map и set?

map - ассоциативный массив, по ключу в контейнере хранится одно значение multimap - ассоциативный массив с повторяющимися ключами set - массив уникальных ключей без значений

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

template , class Allocator =allocator > class std::set{. . . }; class T – параметр, определяющий тип значений, class Comp = less - параметр, определяющий критерий Имеется три способа определить объект типа set: set set1; // создается пустое множество int а[5] = { 1. 2. 3. 4, 5}; set set2(a, а + 5);// инициализация копированием set set3(set2); // инициализация другим множеством

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

упорядоченные по некоторому ключу, являющемуся атрибутом самого объекта.

17.Какие операции определены для контейнера set?

< “меньше”

18.Написать функцию для добавления элементов в контейнер set.

void add(set2& a) {

int t;

cin >> t;

a.insert(t);

}

19.Написать функцию для печати контейнера set.

void show(set2& a) {

for (auto i = a.begin();i != a.end();i++) {

cout << \*i << " ";

}

cout << endl;

}20.Чем отличаются контейнеры set и multiset?

multiset — это контейнер, который также будет содержать элементы в отсортированном порядке при добавлении, но он хранит повторяющееся элементы, по сравнению с множеством set