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: Последовательные контейнеры STL.

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Токарев Павел Аркадьевич

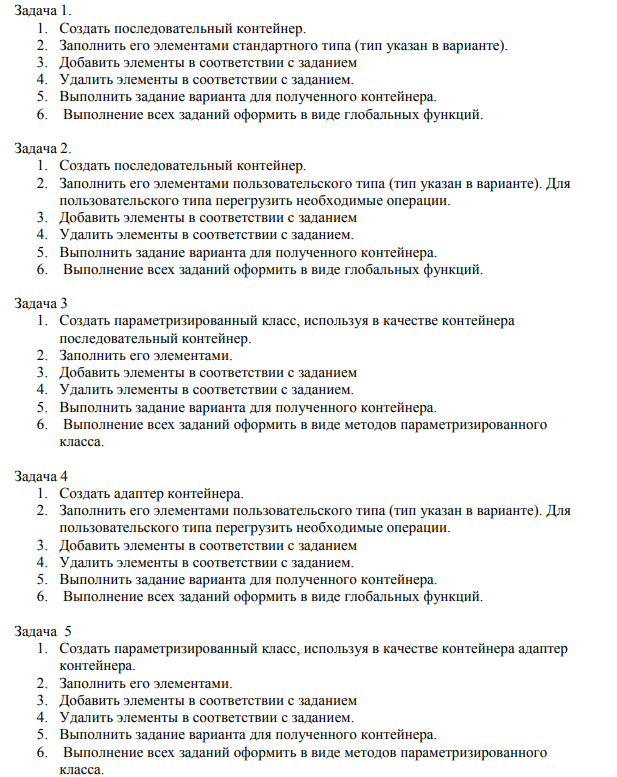
Проверил

Доцент кафедры ИТАС

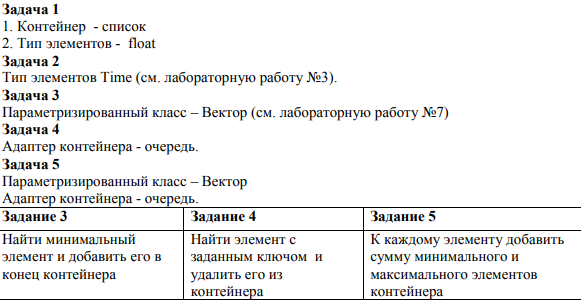
Полякова Ольга Андреевна

Г. Пермь-2023

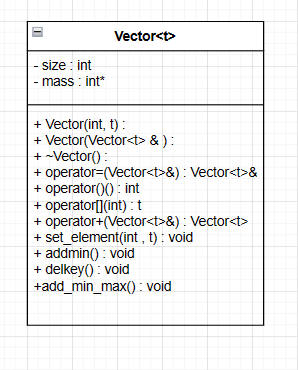
**Постановка задачи**



Вариант 2:

****

**Диаграмма UML**

****

**Код программы**

task1.h

#pragma once

#include <iostream>

#include<list>

#include"Time.h"

#include<algorithm>

using namespace std;

typedef list<float> lst;

void show(lst& a) {

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

cout << \*i << " ";

}

cout << endl;

}

lst makelist(int count) {

lst a;

float tmp;

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

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

cin >> tmp;

a.push\_back(tmp);

}

return a;

}

void find\_min\_and\_add(lst& a) {

float mn = 100000000;

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

if (\*i < mn) {

mn = \*i;

}

}

a.push\_back(mn);

}

void find\_key(lst& a,float key) {

auto j = a.begin();

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

if ((\*i) == key) {

break;

}

j++;

}

a.erase(j);

}

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

float mn = \*std::min\_element(a.begin(), a.end());

float mx = \*std::max\_element(a.begin(), a.end());

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

\*i += (mn + mx);

}

}

void task1() {

list<float> list1;

int n;

cout << "Введите размер списка : ";

cin >> n;

list1=makelist(n);

cout << "Найдём минимальный элемент и добавим его в конец" << endl;

find\_min\_and\_add(list1);

show(list1);

cout << "Найдём элемент с заданным ключём и удалим его" << endl;

float tmp;

cout << "Введите ключ для удаления : ";

cin >> tmp;

find\_key(list1, tmp);

show(list1);

cout << "Добавим сумму максимального и минимального элемента списка к всем элементам" << endl;

add\_min\_max(list1);

show(list1);

}

task2.h

#pragma once

#include <iostream>

#include<list>

#include"Time.h"

#include<algorithm>

using namespace std;

typedef list<Time> lst2;

void show(lst2& a) {

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

cout << \*i << " ";

}

cout << endl;

}

lst2 make\_list(int count) {

lst2 a;

Time tmp;

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

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

cin >> tmp;

a.push\_back(tmp);

}

return a;

}

void find\_min\_and\_add(lst2& a) {

Time mn(100000,100000);

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

if (\*i < mn) {

mn = \*i;

}

}

a.push\_back(mn);

}

void find\_key(lst2& a, Time key) {

auto j = a.begin();

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

if ((\*i) == key) {

break;

}

j++;

}

a.erase(j);

}

void add\_min\_max(lst2& a) {

Time mn(100000, 100000);

Time mx(0, 0);

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

if (\*i < mn) {

mn = \*i;

}

if (\*i > mx) {

mx = \*i;

}

}

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

(\*i) = (\*i) + (mx + mn);

}

}

void task2() {

list<Time> list1;

int n;

cout << "Введите размер списка : ";

cin >> n;

list1 = make\_list(n);

cout << "Найдём минимальный элемент и добавим его в конец" << endl;

find\_min\_and\_add(list1);

show(list1);

cout << "Найдём элемент с заданным ключём и удалим его" << endl;

Time tmp;

cout << "Введите ключ для удаления "<<endl;

cin >> tmp;

find\_key(list1, tmp);

show(list1);

cout << "Добавим сумму максимального и минимального элемента списка к всем элементам" << endl;

add\_min\_max(list1);

show(list1);

}

task3.h

#pragma once

#include"Time.h"

#include<vector>

template <typename t>

class Vector {

private:

int size;

vector<t> mass;

public:

Vector() {

size = 0;

}

Vector(int s, t k) {

size = s;

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

mass.push\_back(k);

}

}

Vector(const Vector<t>& a) {

size = a.size;

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

mass.push\_back(a.mass[i]);

}

}

~Vector() {

}

friend ostream& operator<<(ostream& out, const Vector<t>& a) {

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

out << i + 1 << " " << a.mass[i] << endl;

}

return out;

}

friend istream& operator>>(istream& in, Vector<t>& a) {

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

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

in >> a.mass[i];

}

return in;

}

Vector<t>& operator=(const Vector<t>& a) {

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

mass[i] = a.mass[i];

}

return \*this;

}

int operator()() {

return size;

}

t operator[](int k) {

return mass[k];

}

Vector<t> operator+(const Vector<t>& a) {

Time aw(2, 67);

Vector<t> tmp(a.size, aw);

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

tmp.mass.pop\_back();

}

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

tmp.mass.push\_back(mass[i] + a.mass[i]);

}

return tmp;

}

void set\_element(int index, t& a) {

mass[index] = a;

}

void make\_vector(int sz);

void show();

void add\_min();

void delete\_key(t key);

void add\_min\_max();

};

template <typename t>

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

t a;

size = sz;

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

cin >> a;

mass.push\_back(a);

}

}

template <typename t>

void Vector<t>::show() {

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

cout << \*i << " ";

}

cout << endl;

}

template <typename t>

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

t min(100000, 100000);

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

if ((\*i) < min) {

min = \*i;

}

}

mass.push\_back(min);

}

template <typename t>

void Vector<t>::delete\_key(t key) {

int j = 0;

bool fl = false;

vector <t> b;

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

if (\*i == key) {

if (fl == false) {

fl = true;

}

else{

b.push\_back(\*i);

}

}

else {

b.push\_back(\*i);

}

}

mass = b;

}

template <typename t>

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

t min(10000, 10000);

t max(0, 0);

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

if ((\*i) > max) {

max = \*i;

}

if ((\*i) < min) {

min = \*i;

}

}

mass.push\_back(min + max);

}

typedef Vector<Time> lst3;

void task3() {

lst3 tmp;

int n;

cout << "Введите размер вектора : ";

cin >> n;

tmp.make\_vector(n);

cout << "Найдём минимальный элемент и добавим его в конец" << endl;

tmp.add\_min();

tmp.show();

cout << "Найдём элемент с заданным ключём и удалим его" << endl;

Time ky;

cout << "Введите ключ для удаления " << endl;

cin >> ky;

tmp.delete\_key(ky);

tmp.show();

cout << "Добавим сумму максимального и минимального элемента вектора к всем элементам" << endl;

tmp.add\_min\_max();

tmp.show();

}

task4.h

#pragma once

#pragma once

#include <iostream>

#include<queue>

#include"Time.h"

#include<algorithm>

using namespace std;

typedef queue<Time> lst4;

void show(lst4 a) {

int t = a.size();

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

cout << a.front()<<" ";

a.pop();

}

cout << endl;

}

lst4 make\_\_list(int count) {

lst4 a;

Time tmp;

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

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

cin >> tmp;

a.push(tmp);

}

return a;

}

void find\_min\_and\_add(lst4& a) {

Time mn(100000, 100000);

int t = a.size();

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

if (a.front() < mn) {

mn = a.front();

}

a.push(a.front());

a.pop();

}

a.push(mn);

}

void find\_key(lst4& a, Time key) {

lst4 b;

while (!a.empty()) {

Time tmp = a.front();

if (tmp != key) {

b.push(tmp);

}

a.pop();

}

a = b;

}

void add\_min\_max(lst4& a) {

Time mn(100000, 100000);

Time mx(0, 0);

lst4 b=a;

Time tmp;

while (!b.empty()) {

if (b.front() > mx) {

mx =b.front();

}

if (b.front() < mn) {

mn = b.front();

}

b.pop();

}

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

tmp = a.front();

a.pop();

a.push(tmp + mn + mx);

}

}

void task4() {

queue<Time> list4;

int n;

cout << "Введите размер очереди : ";

cin >> n;

list4 = make\_\_list(n);

cout << "Найдём минимальный элемент и добавим его в конец" << endl;

find\_min\_and\_add(list4);

show(list4);

cout << "Найдём элемент с заданным ключём и удалим его" << endl;

Time tmp;

cout << "Введите ключ для удаления : ";

cin >> tmp;

find\_key(list4, tmp);

show(list4);

cout << "Добавим сумму максимального и минимального элемента очереди к всем элементам" << endl;

add\_min\_max(list4);

show(list4);

}

task5.h

#pragma once

#include<iostream>

#include<queue>

#include"Time.h"

using namespace std;

template <typename t>

class new\_vector {

private:

queue<t> nv;

int size;

public:

new\_vector();

new\_vector(int n);

~new\_vector();

void make\_vector(int sz);

void show();

void add\_min();

void delete\_key(t key);

void add\_min\_max();

};

template <typename t>

new\_vector<t>::new\_vector() {

}

template<typename t>

new\_vector<t>::new\_vector(int nt) {

t a;

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

cin >> a;

nv.push(a);

}

}

template <typename t>

new\_vector<t>::~new\_vector() {

}

template <typename t>

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

t a;

size = sz;

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

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

cin >> a;

nv.push(a);

}

}

template<typename t>

void new\_vector<t>::show() {

int k= nv.size();

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

cout << nv.front() << endl;

nv.push(nv.front());

nv.pop();

}

cout << endl;

}

template <typename t>

void new\_vector<t>::add\_min() {

queue<t> b=nv;

t min(100000,100000);

while (!b.empty()) {

if (b.front() < min) {

min = b.front();

}

b.pop();

}

nv.push(min);

}

template<typename t>

void new\_vector<t>::delete\_key(t key) {

int j = nv.size();

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

if (nv.front() != key) {

nv.push(nv.front());

}

nv.pop();

}

}

template<typename t>

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

t min(10000, 10000);

t max(0, 0);

int j = nv.size();

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

if (nv.front() < min) {

min = nv.front();

}

if (nv.front() > max) {

max = nv.front();

}

nv.push(nv.front());

nv.pop();

}

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

nv.push(nv.front() + max + min);

nv.pop();

}

}

typedef new\_vector<Time> nw1;

void task5() {

nw1 tmp;

int n;

cout << "Введите размер вектора : ";

cin >> n;

tmp.make\_vector(n);

cout << "Найдём минимальный элемент и добавим его в конец" << endl;

tmp.add\_min();

tmp.show();

cout << "Найдём элемент с заданным ключём и удалим его" << endl;

Time ky;

cout << "Введите ключ для удаления " << endl;

cin >> ky;

tmp.delete\_key(ky);

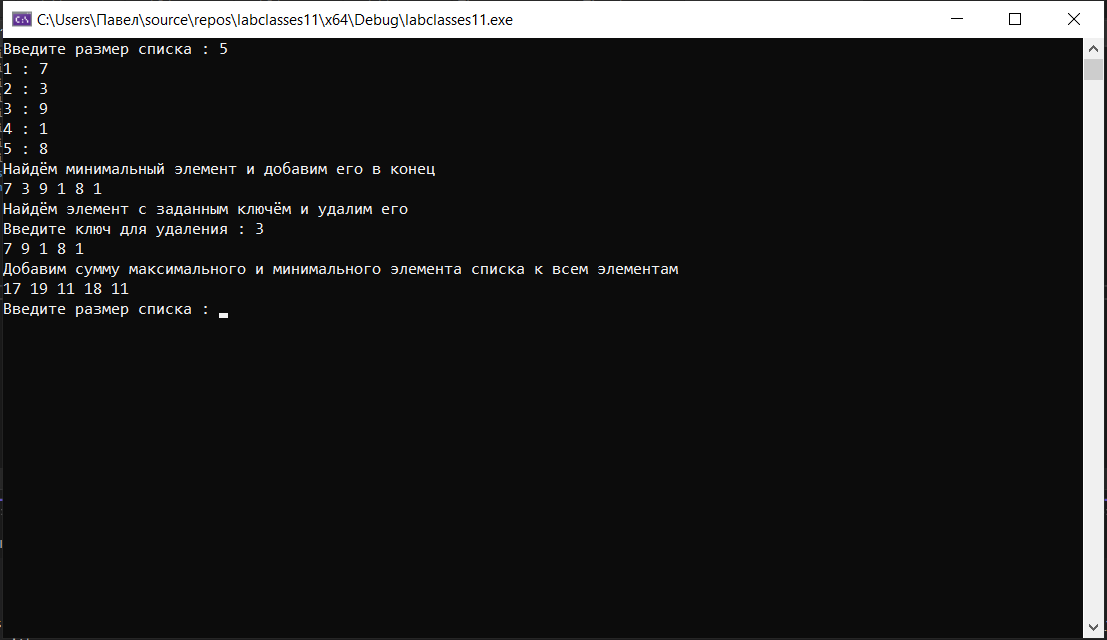
tmp.show();

cout << "Добавим сумму максимального и минимального элемента вектора к всем элементам" << endl;

tmp.add\_min\_max();

tmp.show();}

**Результат работы программы**

****

**Анализ результатов**

Удалось успешно реализовать все задачи через stl контейнеры. Для каждой задачи без ошибок выполняется операция добавления в конец минимального элемента, удаление элемента по ключу и прибавление суммы максимального и минимального элемента к всем элементам контейнера.

**Контрольные вопросы**

1.Из каких частей состоит библиотека STL?

набора контейнерных классов и набора обобщенных алгоритмов.

2.Какие типы контейнеров существуют в STL?

последовательные, ассоциативные, контейнеры-адаптеры и псевдоконтейнеры.

3.Что нужно сделать для использования контейнера STL в своей программе?

необходимо включить в нее соответствующий заголовочный файл. Тип объектов, сохраняемых в контейнере, задается с помощью аргумента шаблона

4.Что представляет собой итератор?

это обобщение концепции указателей: они ссылаются на элементы контейнера.

5.Какие операции можно выполнять над итераторами?

Их можно инкрементировать (++), как обычные указатели, для последовательного продвижения по контейнеру, а также разыменовывать для получения или изменения значения элемента (\*).

6.Каким образом можно организовать цикл для перебора контейнера с использованием итератора?

for (i = first; i != last; ++i),

7.Какие типы итераторов существуют?

• входные

• выходные

• прямые

• двунаправленные итераторы

• итераторы произвольного доступа.

8.Перечислить операции и методы общие для всех контейнеров.

size\_type, iterator, const\_iterator, reference, const\_reference, key\_type, key compare

9.Какие операции являются эффективными для контейнера vector?

индексации []

10.Какие операции являются эффективными для контейнера list?

не поддерживает произвольного доступа

11.Какие операции являются эффективными для контейнера deque?

эффективно поддерживает вставку и удаление первого элемента (так же, как и последнего)

12.Перечислить методы, которые поддерживает последовательный контейнер vector.

push\_back(), pop\_back(), insert, erase.

13.Перечислить методы, которые поддерживает последовательный контейнер list.

push\_back(T&key), pop\_back(), push\_front(T&key), pop\_front(), insert, erase, swap

14.Перечислить методы,которые поддерживает последовательный контейнер deque.

push\_back(T&key), pop\_back(), push\_front(T&key), pop\_front(), insert.

15.Задан контейнер vector. Как удалить из него элементы со 2 по 5?

erase(begin, end): удаляет элементы из диапазона, на начало и конец которого указывают итераторы begin и end. Возвращает итератор на элемент, следующий после последнего удаленного, или на конец контейнера, если удален последний элемент

16.Задан контейнер vector. Как удалить из него последний элемент?

Функцией pop\_back()

17.Задан контейнер list. Как удалить из него элементы со 2 по 5?

Erase(2,5)

18.Задан контейнер list. Как удалить из него последний элемент?

pop\_back()

19.Задан контейнер deque. Как удалить из него элементы со 2 по 5?

Erase(2,5)

20.Задан контейнер deque. Как удалить из него последний элемент?

pop\_back()

21.Написать функцию для печати последовательного контейнера с использованием итератора.

For(auti i : st){

Cout<<i<<endl;}

22.Что представляют собой адаптеры контейнеров?

Они используют какой-нибудь другой контейнер (например, дек или вектор) для хранения своих элементов, но при этом предоставляют свой набор функций для работы с ними

23.Чем отличаются друг от друга объявления stack<int> s и stack <int, list<int>> s?

Что 1 случай — это объявление stack. А 2 — это объявление стака внутри которого есть list

24.Перечислить методы, которые поддерживает контейнер stack.

Empty, pop, push, size, top

25.Перечислить методы, которые поддерживает контейнер queue.

Back, empty, front, pop, push, size

26.Чем отличаются друг от друга контейнеры queue и priority\_queue?

priority\_queue всегда является самым большим или имеет высший приоритет

27.Задан контейнер stack. Как удалить из него элемент с заданным номером?

Pop\_Back ()

28.Задан контейнер queue. Как удалить из него элемент с заданным номером?

Erase (value, pos)

29.Написать функцию для печати контейнера stack с использованием итератора.

while (stack != empty) {

cout << stack.top();

i++;

}

30. Написать функцию для печати контейнера queue с использованием итератора.

while (queue != empty) {

cout << queue.front();

i++;

}