**Пермский национальный исследовательский политехнический университет**

Кафедра “Информационные технологии и автоматизированные системы”

**Лабораторная работа №28**

По дисциплине «Основы алгоритмизации и программирования»

**Тема:**

Последовательные контейнеры библиотеки STL.

Выполнила:

Студентка группы ИВТ-19-1б

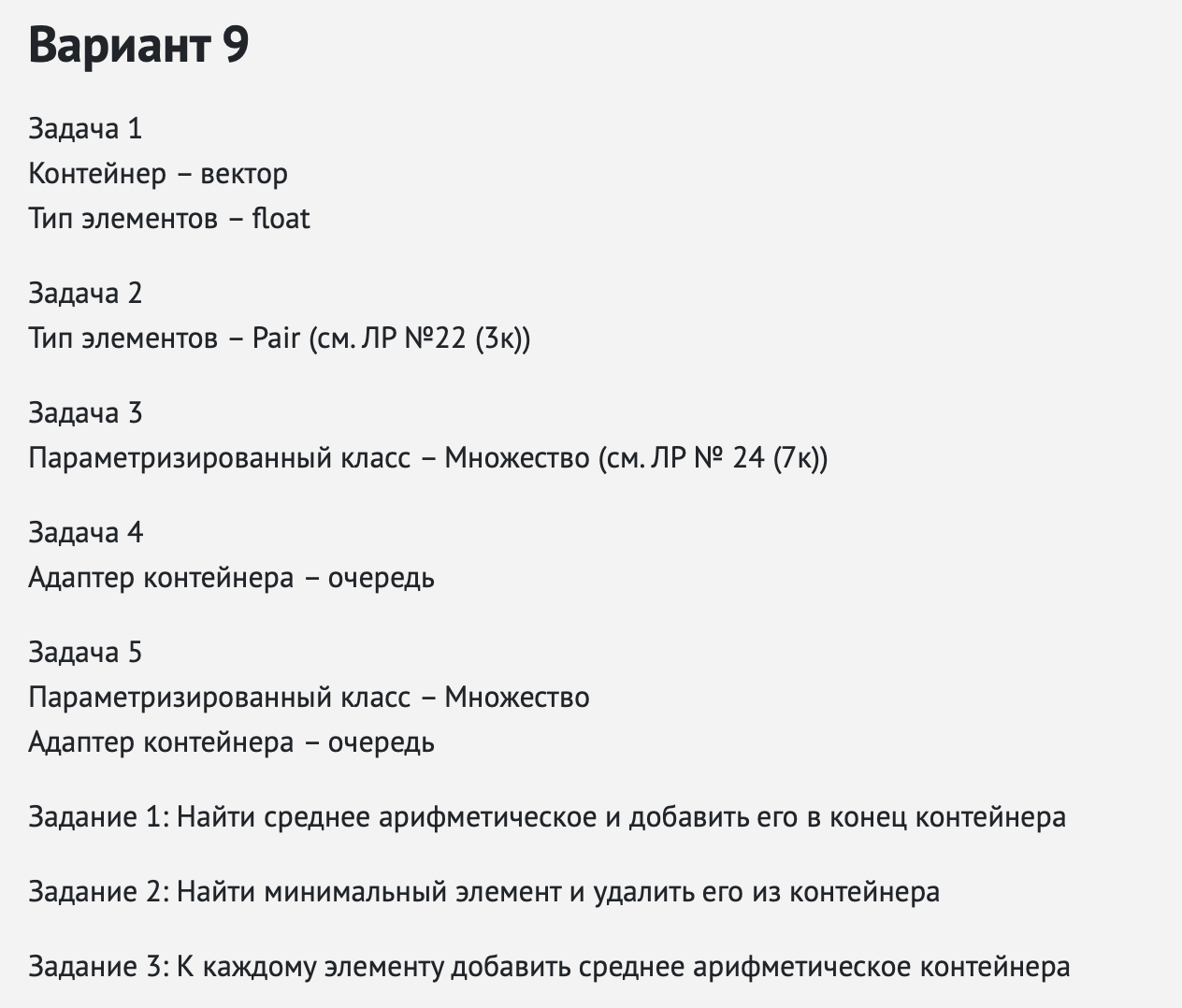
Еремеева Полина Алексеевна

Проверила:

доцент кафедры “ИТАС”

Полякова О.А.

Пермь, 2020

**Задача 1**

**Код**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

**using** **namespace** std;

**int** n;

**bool** q = **false**;

**void** push(vector<**double**>&, **int**);

**void** print(vector<**double**>&);

**void** modify\_lst(vector<**double**>);

**double** sredn(vector<**double**>& l);

**void** del(vector<**double**>& l, **double** k);

**void** zad3(vector<**double**>&);

**int** main()

{

srand(time(0));

setlocale(LC\_ALL, "rus");

cout << "Найти среднее арифметическое и добавить его в конец контейнера\n";

vector<**double**> l;

cout << "Введите кол-во элементов контейнера: ";

cin >> n;

cout << endl;

push(l, n);

print(l);

cout << endl;

**double** s = sredn(l);

l.push\_back(s);

print(l);

cout << "\n---------------------------------------------------------\n";

cout << "Найти элемент с заданным ключом и удалить его из контейнера\n";

cout << "\nВведите искомый элемент: ";

**double** k;

cin >> k;

del(l,k);

print(l);

cout << "\n---------------------------------------------------------\n";

cout << "Из каждого элемента вычесть разность максимального и минимального элементов контейнера\n";

zad3(l);

print(l);

}

**void** push(vector<**double**>& l, **int** n)

{

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

{

**double** a = rand() % 50-25;

l.push\_back(a\*1.11);

}

}

**void** print(vector<**double**>& l)

{

cout << "Vector: ";

**for** (vector<**double**>::iterator it = l.begin(); it != l.end(); it++)

{

cout << \*it;

cout << ' ';

}

}

**double** sredn(vector<**double**>& l)

{

**double** s = 0;//перебор вектора

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

s += l[i];

**int** n = l.size();//количество элементов в векторе

**return** s/n;

}

**void** del(vector<**double**>& l, **double** k)

{

vector<**double**>::iterator it = l.begin();

vector<**double**>::iterator w;

**for** (vector<**double**>::iterator it = l.begin(); it != l.end(); it++)

{

**if** ((\*it == k) && (q == **false**))

{

w = it;

q = **true**;

}

}

**if** (q == **false**)

{

cout << "Такого числа нет в контейнере!\n";

}

**else**

{

l.erase(w);

}

}

**void** zad3(vector<**double**>& l)

{

vector<**double**>::iterator it = l.begin();

vector<**double**>::iterator z = it;

vector<**double**>::iterator v = it;

**for** (vector<**double**>::iterator it = l.begin(); it != l.end(); it++)

{

**if** (\*z > \* it)

{

z = it;

}

}

**double** min = \*z;

**for** (vector<**double**>::iterator it = l.begin(); it != l.end(); it++)

{

**if** (\*v < \* it)

{

v = it;

}

}

**double** max = \*v;

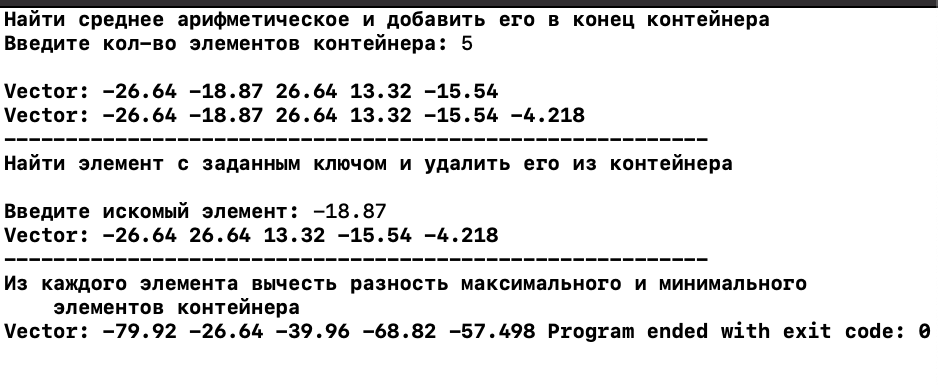
**for** (vector<**double**>::iterator it = l.begin(); it != l.end(); it++)

{

\*it = (\*it) - (max - min);

}

}

**Тестирование**

**Задача 2**

**Код**

**main.cpp**

#include <iostream>

#include "Time.hpp"

#include <vector>

#include <cstdlib>

**bool** j = **true**;

**using** **namespace** std;

Time& Time::**operator**=(**const** Time& t)

{

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

min = t.min;

sec = t.sec;

**return** \***this**;

}

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

{

cout << "min?"; cin >> t.min;

cout << "sec?"; cin >> t.sec;

**return** in;

}

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

{

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

}

**bool** Time::**operator**<(**const** Time& t)

{

**if** (min < t.min)**return** **true**;

**if** (min == t.min && sec < t.sec)**return** **true**;

**return** **false**;

}

**bool** Time::**operator**>(**const** Time& t)

{

**if** (min > t.min)**return** **true**;

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

**return** **false**;

}

Time **operator**\*(**const** Time& t, **const** Time& r)

{

**int** temp1 = t.min \* 60 + t.sec;

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

Time p;

p.min = (temp1 \* temp2) / 60;

p.sec = (temp1 \* temp2) % 60;

**return** p;

}

Time **operator**-(**const** Time& t, **const** Time& r)

{

**int** temp1 = t.min \* 60 + t.sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**+(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** **int**& t)

{

**int** temp1 = min \* 60 + sec;

Time p;

p.min = (temp1 / t) / 60;

p.sec = (temp1 / t) % 60;

**return** p;

}

**typedef** vector<Time>Vec;

Vec make\_vector(**int** n)

{

Vec v;

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

{

Time a;

cin >> a;

v.push\_back(a);

}

**return** v;

}

**void** print\_vector(Vec v)

{

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

cout << v[i] << " ";

cout << endl;

}

**bool** **operator** ==(**const** Time& a, **const** Time& b)

{

**int** k = a.min \* 60 + a.sec;

**int** e = b.min \* 60 + b.sec;

**if** (k == e)

**return** **true**;

**else**

**return** **false**;

}

**void** add\_vector(Vec& v, **int** s, **int** pos)

{

swap(v[s], v[pos]);

}

Vec srednee(Vec v)

{

Time s;//переборвектора

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

s =s + v[i];

}

**int** n = v.size();//количество элементов в векторе

v.push\_back(s / n);

**return** v;

}

**int** elem(Vec v,**int** z)

{

**int** m = 0;

Time k = v[z];

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

{

**if** (k == v[i])

{

k = v[i];

m = i;

}

}

**return** m;

}

**int** min\_elem(Vec v)

{

**int** m = 0;

Time k = v[0];

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

{

**if** (k > v[i])

{

k = v[i];

m = i;

}

}

**return** m;

}

**int** max\_elem(Vec v)

{

**int** m = 0;

Time k = v[0];

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

{

**if** (k < v[i])

{

k = v[i];

m = i;

}

}

**return** m;

}

**void** razn(Vec& v,**int** m, **int** x)

{

Time t = v[m];

Time z = v[x];

t = t + v[x];

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

{

v[i] = v[i] - t;

}

}

**int** main()

{

**int** sw;

setlocale(LC\_ALL, "rus");

**try**

{

vector<Time> v;

vector<Time>::iterator vi = v.begin();

**int** n;

cout << "Задание 1\n";

cout << "N?";

cin >> n;

v = make\_vector(n);

print\_vector(v);

**if** (j)

{

cout << "\nCреднее арифметическое в конец контейнера: " ;

v=srednee(v);

print\_vector(v);

cout << "\n---------------------------------------------------------\n";

cout << "Найти элемент с заданным ключом и удалить его из контейнера\n\n";

cout << "\nВведите искомый элемент: ";

**int** k;

cin >> k;

**int** Elem = elem(v, k);

v.erase(v.begin() + Elem);

print\_vector(v);

cout << "\n---------------------------------------------------------\n";

cout << "Из каждого элемента вычесть разность максимального и минимального элементов контейнера\n\n";

**int** MIN = min\_elem(v);

**int** MAX = max\_elem(v);

razn(v,MIN, MAX);

print\_vector(v);

}

**else**

{

cout << "\nДанного объекта в векторе нет\n";

}

}

**catch** (**int**)

{

cout << "error!";

}

}

**Time.hpp**

#pragma once

#include <iostream>

**using** **namespace** std;

**class** Time

{

**int** min, sec;

**public**:

Time() { min = 0; sec = 0; }

Time(**int** m, **int** s) { 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; }

Time& **operator**=(**const** Time&);

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

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

Time **operator**+(**const** Time&);

Time **operator**/(**const** Time&);

Time **operator**/(**const** **int**&);

**bool** **operator** > (**const** Time&);

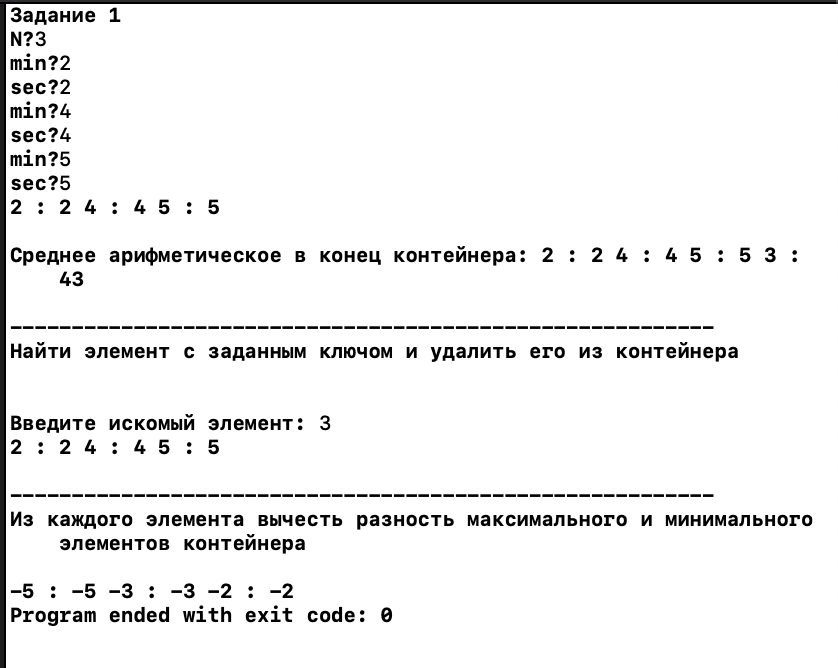
**bool** **operator** < (**const** Time&);

**friend** **bool** **operator** ==(**const** Time&, **const** Time&);

**friend** Time **operator**\*(**const** Time& t, **const** Time& r);

**friend** Time **operator**-(**const** Time& t, **const** Time& r);

};

**Тестирование**

**Задача 3**

**Код**

**Main.cpp**

#include <iostream>

#include "Time.hpp"

#include <vector>

#include <cstdlib>

**bool** j = **true**;

**using** **namespace** std;

Time& Time::**operator**=(**const** Time& t)

{

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

min = t.min;

sec = t.sec;

**return** \***this**;

}

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

{

cout << "min?"; cin >> t.min;

cout << "sec?"; cin >> t.sec;

**return** in;

}

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

{

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

}

**bool** Time::**operator**<(**const** Time& t)

{

**if** (min < t.min)**return** **true**;

**if** (min == t.min && sec < t.sec)**return** **true**;

**return** **false**;

}

**bool** Time::**operator**>(**const** Time& t)

{

**if** (min > t.min)**return** **true**;

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

**return** **false**;

}

Time **operator**\*(**const** Time& t, **const** Time& r)

{

**int** temp1 = t.min \* 60 + t.sec;

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

Time p;

p.min = (temp1 \* temp2) / 60;

p.sec = (temp1 \* temp2) % 60;

**return** p;

}

Time **operator**-(**const** Time& t, **const** Time& r)

{

**int** temp1 = t.min \* 60 + t.sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**+(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** **int**& t)

{

**int** temp1 = min \* 60 + sec;

Time p;

p.min = (temp1 / t) / 60;

p.sec = (temp1 / t) % 60;

**return** p;

}

**typedef** vector<Time>Vec;

Vec make\_vector(**int** n)

{

Vec v;

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

{

Time a;

cin >> a;

v.push\_back(a);

}

**return** v;

}

**void** print\_vector(Vec v)

{

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

cout << v[i] << " ";

cout << endl;

}

**bool** **operator** ==(**const** Time& a, **const** Time& b)

{

**int** k = a.min \* 60 + a.sec;

**int** e = b.min \* 60 + b.sec;

**if** (k == e)

**return** **true**;

**else**

**return** **false**;

}

**void** add\_vector(Vec& v, **int** s, **int** pos)

{

swap(v[s], v[pos]);

}

Vec srednee(Vec v)

{

Time s;//переборвектора

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

s = s + v[i];

}

**int** n = v.size();//количество элементов в векторе

v.push\_back(s / n);

**return** v;

}

**int** elem(Vec v, **int** z)

{

**int** m = 0;

Time k = v[z];

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

{

**if** (k == v[i])

{

k = v[i];

m = i;

}

}

**return** m;

}

**int** min\_elem(Vec v)

{

**int** m = 0;

Time k = v[0];

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

{

**if** (k > v[i])

{

k = v[i];

m = i;

}

}

**return** m;

}

**int** max\_elem(Vec v)

{

**int** m = 0;

Time k = v[0];

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

{

**if** (k < v[i])

{

k = v[i];

m = i;

}

}

**return** m;

}

**void** raz(Vec& v, **int** m, **int** x)

{

Time t = v[m];

Time z = v[x];

t = t + v[x];

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

{

v[i] = v[i] - t;

}

}

**int** main()

{

**int** sw;

setlocale(LC\_ALL, "rus");

**try**

{

vector<Time> v;

vector<Time>::iterator vi = v.begin();

**int** n;

cout << "Задание 1\n";

cout << "N?";

cin >> n;

v = make\_vector(n);

print\_vector(v);

**if** (j)

{

cout << "\nCреднее арифметическое в конец контейнера: ";

v = srednee(v);

print\_vector(v);

cout << "\n---------------------------------------------------------\n";

cout << "Задание 2\n\n";

cout << "\nВведите искомый элемент: ";

**int** k;

cin >> k;

**int** Elem = elem(v, k);

v.erase(v.begin() + Elem);

print\_vector(v);

cout << "\n---------------------------------------------------------\n";

cout << "Задание 3\n\n";

**int** MIN = min\_elem(v);

**int** MAX = max\_elem(v);

raz(v, MIN, MAX);

print\_vector(v);

}

**else**

{

cout << "\nДанного объекта в векторе нет\n";

}

}

**catch** (**int**)

{

cout << "error!";

}

}

**Time.hpp**

#pragma once

#include <iostream>

**using** **namespace** std;

**class** Time

{

**int** min, sec;

**public**:

Time() { min = 0; sec = 0; }

Time(**int** m, **int** s) { 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; }

Time& **operator**=(**const** Time&);

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

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

Time **operator**+(**const** Time&);

Time **operator**/(**const** Time&);

Time **operator**/(**const** **int**&);

**bool** **operator** > (**const** Time&);

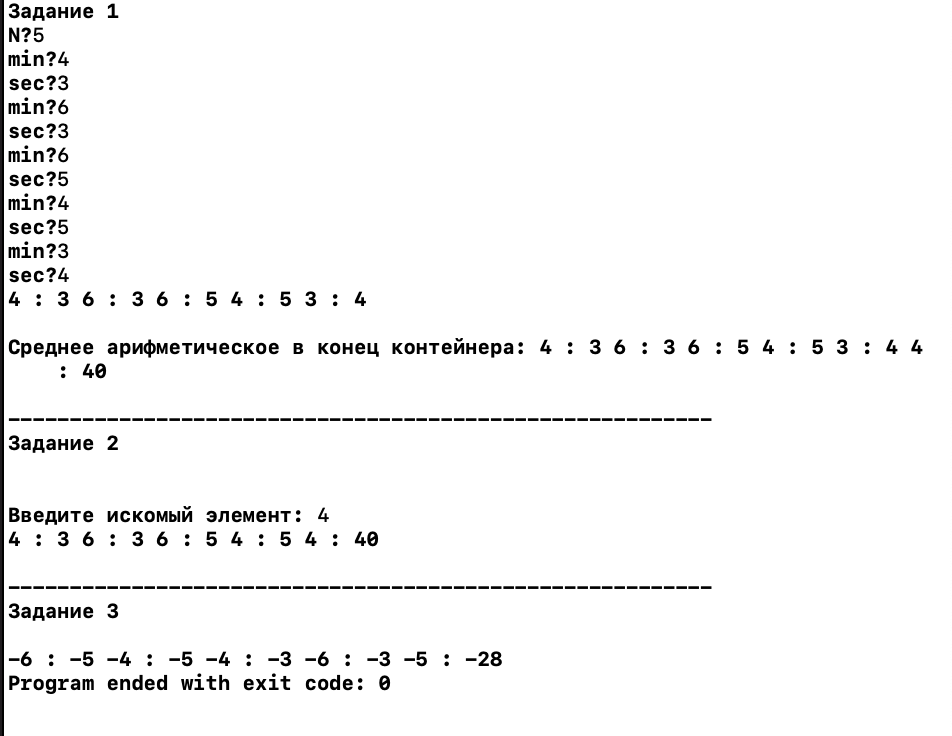
**bool** **operator** < (**const** Time&);

**friend** **bool** **operator** ==(**const** Time&, **const** Time&);

**friend** Time **operator**\*(**const** Time& t, **const** Time& r);

**friend** Time **operator**-(**const** Time& t, **const** Time& r);

};

**Тестирование**

**Задача 4**

**Код**

**Main.cpp**

#include "Time.hpp"

#include <iostream>

#include <stack>

#include <vector>

**using** **namespace** std;

**typedef** stack<Time>st;

**typedef** vector<Time>Vec;

st make\_stack(**int** n)

{

st s;

Time m;

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

{

cin >> m;

s.push(m);

}

**return** s;

}

**void** print\_stack(st s) {

**while** (!s.empty()) {

cout << s.top() << ' ';

s.pop();

}

}

Vec copy\_stack(st s)

{

Vec v;

**while** (!s.empty())

{

v.push\_back(s.top());

s.pop();

}

**return** v;

}

st copy\_vector(Vec v)

{

st s;

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

{

s.push(v[i]);

}

**return** s;

}

Time Srednee(st s)

{

Vec v;

**int** n = 1;

Time sum = s.top();

s.pop();

**while** (!s.empty())

{

sum = sum + s.top();

s.pop();

n++;

}

s = copy\_vector(v);

**return** sum / n;

}

**void** Add\_sredn(st& s)

{

Time z;

Vec v = copy\_stack(s);

Time m = Srednee(s);

v.push\_back(m);

s = copy\_vector(v);

}

**void** Del\_el(st& s, **int** pos)

{

Vec v;

Time m;

Time k;

**int** i = 1;

**while** (!s.empty())

{

m = s.top();

**if** (i != pos)v.push\_back(m);

s.pop();

i++;

}

s = copy\_vector(v);

}

Time Min(st s)

{

Time m = s.top();

Vec v = copy\_stack(s);

**while** (!s.empty())

{

**if** (s.top() < m)m = s.top();

s.pop();

}

s = copy\_vector(v);

**return** m;

}

Time Max(st s)

{

Time m = s.top();

Vec v = copy\_stack(s);

**while** (!s.empty())

{

**if** (s.top() > m)m = s.top();

s.pop();

}

s = copy\_vector(v);

**return** m;

}

**void** Razn(st& s)

{

Time m = Min(s);

m = m + Max(s);

Vec v;

Time t;

**while** (!s.empty())

{

t = s.top();

v.push\_back(t - m);

s.pop();

}

s = copy\_vector(v);

}

**int** main() {

setlocale(LC\_ALL, "ru");

Time z;

Time m;

st s;

**int** n;

cout << "Количество элементов стека : ";

cin >> n;

s = make\_stack(n);

print\_stack(s);

cout << "Найти элемент с заданным ключом и удалить его из контейнера : ";

**int** pos;

cin >> pos;

Del\_el(s, pos);

print\_stack(s);

cout << "Cреднее арифметическое в конеце контейнера : " << endl;

Add\_sredn(s);

print\_stack(s);

cout << "Из каждого элемента вычесть разность максимального и минимального элементов контейнера : " << endl;

Razn(s);

print\_stack(s);

}

**Time.hpp**

#include <iostream>

**using** **namespace** std;

**class** Time

{

**int** min;

**int** sec;

**public**:

Time(**void**);

**public**:

**virtual** ~Time(**void**);

Time(**long**, **int**);

Time(**const** Time&);

**long** Get\_min() { **return** min; }

**int** Get\_sec() { **return** sec; }

**void** Set\_min(**long**);

**void** Set\_sec(**int**);

Time& **operator**=(**const** Time&);

Time **operator**+(**const** Time&);

Time **operator**/(**const** Time&);

Time **operator**/(**const** **int**&);

Time **operator**\*(**const** Time&);

Time **operator**\*(**const** **int**&);

**bool** **operator** >(**const** Time&);

**bool** **operator** <(**const** Time&);

**friend** istream& **operator**>>(istream& in, Time& c);

**friend** ostream& **operator**<<(ostream& out, **const** Time& c);

**friend** Time **operator**-(**const** Time& t, **const** Time& r);

};

Time::Time(**void**)

{

min = 0;

sec = 0;

}

Time::~Time(**void**)

{

}

Time::Time(**long** N, **int** S)

{

**int** b, c;

**if** (S >= 60)

{

b = S / 60;

c = S % 60;

S = c;

N = N + b;

}

**this**->min = N;

**this**->sec = S;

}

Time::Time(**const** Time& Time)

{

min = Time.min;

sec = Time.sec;

}

**void** Time::Set\_sec(**int** C)

{

sec = C;

}

**void** Time::Set\_min(**long** M)

{

min = M;

}

Time& Time::**operator**=(**const** Time& c)

{

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

min = c.min;

sec = c.sec;

**return** \***this**;

}

istream& **operator**>>(istream& in, Time& c)

{

cout << "\nmin : "; in >> c.min;

cout << "\nsec : "; in >> c.sec;

**return** in;

}

ostream& **operator**<<(ostream& out, **const** Time& c)

{

out << c.min << ':' << c.sec;

out << "\n";

**return** out;

}

**bool** Time::**operator** <(**const** Time& t)

{

**if** (min < t.min)**return** **true**;

**if** (min == t.min && sec < t.sec)**return** **true**;

**return** **false**;

}

**bool** Time::**operator** >(**const** Time& t)

{

**if** (min > t.min)**return** **true**;

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

**return** **false**;

}

Time Time::**operator**+(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** **int**& t)

{

**int** temp1 = min \* 60 + sec;

Time p;

p.min = (temp1 / t) / 60;

p.sec = (temp1 / t) % 60;

**return** p;

}

Time Time::**operator**\*(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

p.min = (temp1 \* temp2) / 60;

p.sec = (temp1 \* temp2) % 60;

**return** p;

}

Time Time::**operator**\*(**const** **int**& t)

{

**int** temp1 = min \* 60 + sec;

Time p;

p.min = (temp1 \* t) / 60;

p.sec = (temp1 \* t) % 60;

**return** p;

}

Time **operator**-(**const** Time& t, **const** Time& r)

{

**int** temp1 = t.min \* 60 + t.sec;

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

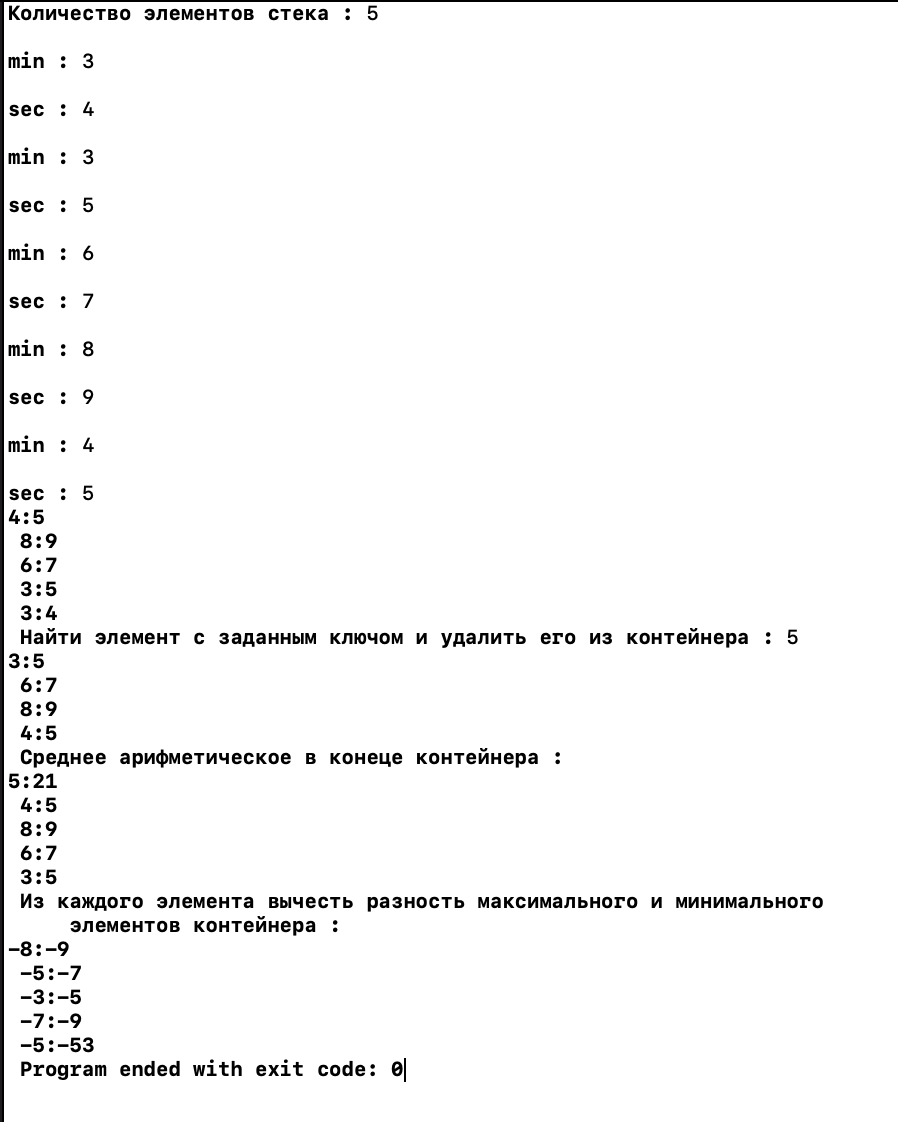
Time p;

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

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

**return** p;

}

**Тестирование**

**Задача 5**

**Код**

**Main.cpp**

#include <iostream>

#include "Vector.hpp"

#include "Time.hpp"

#include <vector>

#include <stack>

**using** **namespace** std;

**int** main()

{

setlocale(LC\_ALL, "ru");

Vector<Time>v(3);

v.print();

cout << "Найти элемент с заданным ключом и удалить его из контейнера : ";

**int** nom;

cin >> nom;

v.add(nom);

v.print();

cout << "Cреднее арифметическое в конеце контейнера : " << endl;

v.add\_sred();

v.print();

cout << "Из каждого элемента вычесть разность максимального и минимального элементов контейнера :" << endl;

v.razn();

v.print();

}

**Vector.hpp**

#pragma once

#include <iostream>

#include <vector>

#include <stack>

**using** **namespace** std;

**template**<**class** T> **class** Vector

{

stack <T> s;

**int** len;

**public**:

Vector();

Vector(**int** n);

Vector(**const** Vector<T>&);

**void** add(**int** pos);

**void** print();

T sredn();

T add\_sred();

T max();

T min();

**void** razn();

};

**template** <**class** T>

vector<T> copy\_stack(stack<T> s)

{

vector<T> v;

**while** (!s.empty())

{

v.push\_back(s.top());

s.pop();

}

**return** v;

}

**template** <**class** T>

stack<T> copy\_vector(vector<T> v)

{

stack<T> s;

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

{

s.push(v[i]);

}

**return** s;

}

**template** <**class** T>

Vector<T>::Vector()

{

len = 0;

}

**template** <**class** T>

Vector<T>::Vector(**int** n)

{

T a;

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

{

cin >> a;

s.push(a);

}

len = s.size();

}

**template** <**class** T>

Vector<T>::Vector(**const** Vector<T>& Vec)

{

vector<T> v;

len = v.len;

vector v = copy\_stack(Vec.s);

s = copy\_vector(v);

}

**template** <**class** T>

**void** Vector<T>::add(**int** pos)

{

vector <T>v;

T t;

**int** i = 1;

**while** (!s.empty())

{

t = s.top();

**if** (i != pos)v.push\_back(t);

s.pop();

i++;

}

s = copy\_vector(v);

}

**template** <**class** T>

**void** Vector<T>::print()

{

vector<T> v = copy\_stack(s);

**while** (!s.empty())

{

cout << s.top() << endl;

s.pop();

}

s = copy\_vector(v);

}

**template** <**class** T>

T Vector<T>::sredn()

{

vector<T> v = copy\_stack(s);

**int** n = 1;

T sum = s.top();

s.pop();

**while** (!s.empty())

{

sum = sum + s.top();

s.pop();

n++;

}

s = copy\_vector(v);

**return** sum / n;

}

**template** <**class** T>

T Vector<T>::add\_sred()

{

T m = sredn();

vector<T> v = copy\_stack(s);

T t;

**int** i = 1;

v.push\_back(m);

s = copy\_vector(v);

**return** m;

}

**template** <**class** T>

T Vector<T>::max()

{

T m = s.top();

vector<T> v = copy\_stack(s);

**while** (!s.empty())

{

**if** (s.top() > m)m = s.top();

s.pop();

}

s = copy\_vector(v);

**return** m;

}

**template** <**class** T>

T Vector<T>::min()

{

T m = s.top();

vector<T> v = copy\_stack(s);

**while** (!s.empty())

{

**if** (s.top() < m)m = s.top();

s.pop();

}

s = copy\_vector(v);

**return** m;

}

**template** <**class** T>

**void** Vector<T>::razn()

{

T m = min();

m = m + max();

vector<T> v;

T t;

**while** (!s.empty())

{

t = s.top(); v.push\_back(t - m);

s.pop();

}

s = copy\_vector(v);

}

**Time.hpp**  
#include <iostream>

**using** **namespace** std;

**class** Time

{

**int** min;

**int** sec;

**public**:

Time(**void**);

**public**:

**virtual** ~Time(**void**);

Time(**long**, **int**);

Time(**const** Time&);

**long** Get\_min() { **return** min; }

**int** Get\_sec() { **return** sec; }

**void** Set\_min(**long**);

**void** Set\_sec(**int**);

Time& **operator**=(**const** Time&);

Time **operator**+(**const** Time&);

Time **operator**/(**const** Time&);

Time **operator**/(**const** **int**&);

Time **operator**\*(**const** Time&);

Time **operator**\*(**const** **int**&);

**bool** **operator** >(**const** Time&);

**bool** **operator** <(**const** Time&);

**friend** istream& **operator**>>(istream& in, Time& c);

**friend** ostream& **operator**<<(ostream& out, **const** Time& c);

**friend** Time **operator**-(**const** Time& t, **const** Time& r);

};

Time::Time(**void**)

{

min = 0;

sec = 0;

}

Time::~Time(**void**)

{

}

Time::Time(**long** N, **int** S)

{

**int** b, c;

**if** (S >= 60)

{

b = S / 60;

c = S % 60;

S = c;

N = N + b;

}

**this**->min = N;

**this**->sec = S;

}

Time::Time(**const** Time& Time)

{

min = Time.min;

sec = Time.sec;

}

**void** Time::Set\_sec(**int** C)

{

sec = C;

}

**void** Time::Set\_min(**long** M)

{

min = M;

}

Time& Time::**operator**=(**const** Time& c)

{

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

min = c.min;

sec = c.sec;

**return** \***this**;

}

istream& **operator**>>(istream& in, Time& c)

{

cout << "\nmin : "; in >> c.min;

cout << "\nsec : "; in >> c.sec;

**return** in;

}

ostream& **operator**<<(ostream& out, **const** Time& c)

{

out << c.min << ':' << c.sec;

out << "\n";

**return** out;

}

**bool** Time::**operator** <(**const** Time& t)

{

**if** (min < t.min)**return** **true**;

**if** (min == t.min && sec < t.sec)**return** **true**;

**return** **false**;

}

**bool** Time::**operator** >(**const** Time& t)

{

**if** (min > t.min)**return** **true**;

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

**return** **false**;

}

Time Time::**operator**+(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

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

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

**return** p;

}

Time Time::**operator**/(**const** **int**& t)

{

**int** temp1 = min \* 60 + sec;

Time p;

p.min = (temp1 / t) / 60;

p.sec = (temp1 / t) % 60;

**return** p;

}

Time Time::**operator**\*(**const** Time& t)

{

**int** temp1 = min \* 60 + sec;

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

Time p;

p.min = (temp1 \* temp2) / 60;

p.sec = (temp1 \* temp2) % 60;

**return** p;

}

Time Time::**operator**\*(**const** **int**& t)

{

**int** temp1 = min \* 60 + sec;

Time p;

p.min = (temp1 \* t) / 60;

p.sec = (temp1 \* t) % 60;

**return** p;

}

Time **operator**-(**const** Time& t, **const** Time& r)

{

**int** temp1 = t.min \* 60 + t.sec;

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

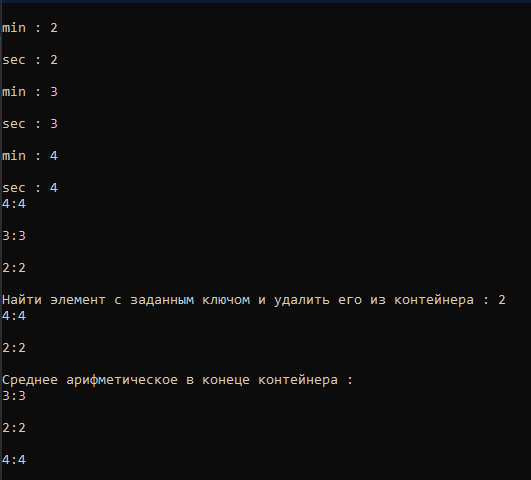
Time p;

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

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

**return** p;

}

**Тестирование**

