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»

Тема:Стандартные обобщенные алгоритмы библиотеки STL.

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Протасов Н.С.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи**

### Задача1.

* + - 1. Создатьпоследовательныйконтейнер.
      2. Заполнить его элементами пользовательского типа (тип указан в варианте). Дляпользовательского типаперегрузитьнеобходимыеоперации.
      3. Заменитьэлементывсоответствиисзаданием(использовать алгоритмы

replace\_if(),replace\_copy(),replace\_copy\_if(),fill()).

* + - 1. Удалитьэлементывсоответствиисзаданием(использоватьалгоритмы

remove(),remove\_if(),remove\_copy\_if(),remove\_copy())

* + - 1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля(использоватьалгоритмsort()).
      2. Найти в контейнере заданный элемент (использовать алгоритмыfind(),find\_if(),count(),count\_if()).
      3. Выполнитьзаданиевариантадляполученногоконтейнера(использоватьалгоритм

for\_each()).

* + - 1. Для выполнения всех заданий использовать стандартные алгоритмы библиотекиSTL.

### Задача2.

1. Создать адаптерконтейнера.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Дляпользовательского типаперегрузитьнеобходимыеоперации.
3. Заменитьэлементывсоответствиисзаданием(использовать алгоритмы

replace\_if(),replace\_copy(),replace\_copy\_if(),fill()).

1. Удалитьэлементывсоответствиисзаданием(использоватьалгоритмы

remove(),remove\_if(),remove\_copy\_if(),remove\_copy())

1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля(использоватьалгоритм sort()).
2. Найтивконтейнереэлементсзаданным ключевымполем(использовать

алгоритмыfind(),find\_if(),count(),count\_if()).

1. Выполнитьзаданиевариантадляполученногоконтейнера(использоватьалгоритм

for\_each()).

1. Для выполнения всех заданий использовать стандартные алгоритмы библиотекиSTL.

### Задача3

1. Создатьассоциативныйконтейнер.
2. Заполнить его элементами пользовательского типа (тип указан в варианте). Дляпользовательского типаперегрузитьнеобходимыеоперации.
3. Заменитьэлементывсоответствиисзаданием(использовать алгоритмы

replace\_if(),replace\_copy(),replace\_copy\_if(),fill()).

1. Удалитьэлементывсоответствиисзаданием(использоватьалгоритмы

remove(),remove\_if(),remove\_copy\_if(),remove\_copy())

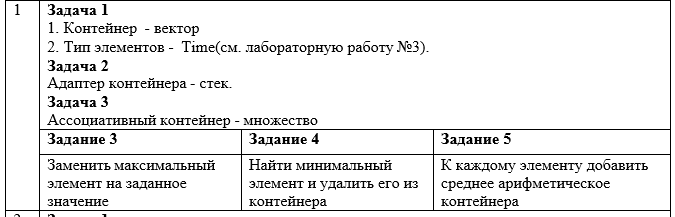
1. Отсортировать контейнер по убыванию и по возрастанию ключевого поля(использоватьалгоритмsort()).
2. Найтивконтейнереэлементсзаданным ключевымполем(использовать

алгоритмыfind(),find\_if(),count(),count\_if()).

1. Выполнитьзаданиевариантадляполученногоконтейнера(использоватьалгоритм

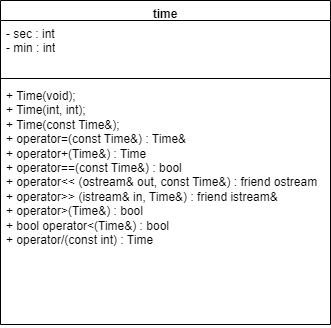
for\_each()).

1. Для выполнения всех заданий использовать стандартные алгоритмы библиотекиSTL.



**UML**

**1**

****

**Функции для решения задачи 1**

#pragma once

#include <iostream>

#include <vector>

#include <algorithm>

#include "Time.h"

using namespace std;

typedef vector<Time> TVector;

TVector MakeVector(int n)

{

Time a;

TVector v;

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

{

cin >> a;

v.push\_back(a);

}

return v;

}

void PrintVector(TVector v)

{

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

{

cout << v[i] << endl;

}

}

Time Average(TVector v)

{

Time s = v[0];

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

{

s = s + v[i];

}

return s / v.size();

}

void Execute1()

{

int n;

cout << "n = ";

cin >> n;

TVector v = MakeVector(n);

Time max = \*(max\_element(v.begin(), v.end()));

Time a;

cin >> a;

replace(v.begin(), v.end(), max, a);

PrintVector(v);

cout << endl;

auto i = remove(v.begin(), v.end(), \*min\_element(v.begin(), v.end()));

v.erase(i, v.end());

PrintVector(v);

cout << endl;

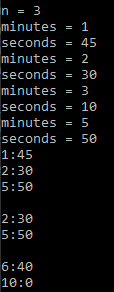
Time s = Average(v);

for\_each(v.begin(), v.end(), [&](Time& t) { t = t + s; });

PrintVector(v);

}

**Результаты работы программы**

****

**Функции для решения задачи 2**

#pragma once

#include <iostream>

#include <algorithm>

#include <vector>

#include <stack>

#include "Time.h"

using namespace std;

typedef stack<Time>St;

typedef vector<Time>Vec;

Vec copy\_stack\_to\_vector(St s)

{

Vec v;

while (!s.empty())

{

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

s.pop();

}

return v;

}

St copy\_vector\_to\_stack(Vec v)

{

St s;

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

{

s.push(v[v.size() - 1 - i]);

}

return s;

}

void PrintStack(St s)

{

while (!s.empty())

{

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

s.pop();

}

cout << endl;

}

St MakeStack(int n)

{

St s;

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

{

Time t;

cin >> t;

s.push(t);

}

return s;

}

Time Average(St s)

{

int n = s.size();

Time sum = s.top();

s.pop();

while (!s.empty())

{

sum = sum + s.top();

s.pop();

}

return sum / n;

}

void add(Time& t, Time& average)

{

t = t + average;

}

void ReplaceMaxElement(St& s, const Time& a)

{

Vec v = copy\_stack\_to\_vector(s);

replace(v.begin(), v.end(), \*max\_element(v.begin(), v.end()), a);

s = copy\_vector\_to\_stack(v);

}

void EraseMinElement(St& s)

{

Vec v = copy\_stack\_to\_vector(s);

auto i = remove(v.begin(), v.end(), \*min\_element(v.begin(), v.end()));

v.erase(i, v.end());

s = copy\_vector\_to\_stack(v);

}

void ForEachInc(St& s, Time a)

{

Vec v = copy\_stack\_to\_vector(s);

for\_each(v.begin(), v.end(), [&](Time& t)

{

t = t + a;

});

s = copy\_vector\_to\_stack(v);

}

void Execute2()

{

int n;

cout << "n = ";

cin >> n;

St s = MakeStack(n);

PrintStack(s);

Time a;

cin >> a;

ReplaceMaxElement(s, a);

PrintStack(s);

EraseMinElement(s);

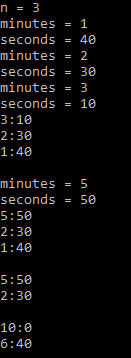
PrintStack(s);

ForEachInc(s, Average(s));

PrintStack(s);

}

**Результаты работы программы**

****

**Функции для решения задачи 3**

#pragma once

#include <iostream>

#include <set>

#include <algorithm>

using namespace std;

typedef set<int> Set;

void ReplaceMax(Set& container, int newValue)

{

if (!container.empty()) {

int maxElement = \*max\_element(container.begin(), container.end());

container.erase(maxElement);

container.insert(newValue);

}

}

void RemoveMin(Set& container)

{

if (!container.empty()) {

container.erase(\*container.begin());

}

}

void AddAverage(Set& container)

{

if (!container.empty()) {

int sum = 0;

for (int num : container) {

sum += num;

}

int average = sum / container.size();

Set tempContainer;

for (int num : container) {

tempContainer.insert(num + average);

}

container = move(tempContainer);

}

}

void PrintSet(Set container)

{

for (int num : container) {

cout << num << " ";

}

cout << endl;

}

Set MakeSet(int n)

{

Set temp;

int a;

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

{

cin >> a;

temp.insert(a);

}

return temp;

}

void Execute3()

{

int n;

cout << "n = ";

cin >> n;

Set container = MakeSet(n);

cout << "Replace n = ";

cin >> n;

ReplaceMax(container, n);

PrintSet(container);

RemoveMin(container);

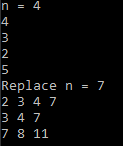
PrintSet(container);

AddAverage(container);

PrintSet(container);

}

**Объяснение результатов работы программы 3**

****