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 ПО ЛАБОРАТОРНОЙ РАБОТЕ №13**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Стандартные обобщенные алгоритмы библиотеки STL

Вариант 5.

Выполнил работу:

Студент группы РИС-22-1Б

Батин В.В.

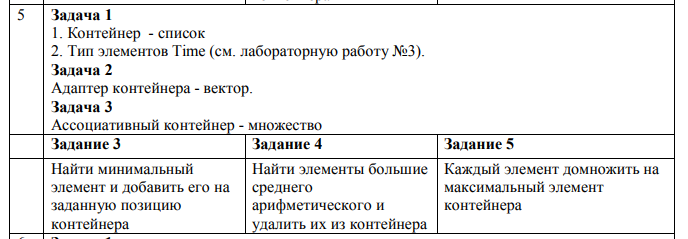
Проверил:

Доцент кафедры ИТАС

Полякова О.А.

г. Пермь-2023

**Постановка задачи**



**Коды программ**

**Задача 1**

#include <iostream>

#include <list>

#include <algorithm>

using namespace std;

class Time {

public:

int hours;

int minutes;

Time(int h, int m) : hours(h), minutes(m) {}

void display() const {

cout << hours << ":" << minutes << endl;

}

};

bool compare\_time(const Time& t1, const Time& t2) {

return (t1.hours < t2.hours) || (t1.hours == t2.hours && t1.minutes < t2.minutes);

}

int main() {

list<Time> times;

times.push\_back(Time(10, 30));

times.push\_back(Time(9, 45));

times.push\_back(Time(11, 15));

times.push\_back(Time(8, 0));

times.push\_back(Time(12, 0));

// Найти минимальный элемент и добавить его в конец контейнера

auto min\_it = min\_element(times.begin(), times.end(), compare\_time);

times.push\_back(\*min\_it);

// Найти элемент с заданным ключом и удалить его из контейнера

auto key = Time(9, 45);

auto key\_it = find\_if(times.begin(), times.end(), [&key](const Time& t) {return compare\_time(t, key) == false && compare\_time(key, t) == false; });

if (key\_it != times.end()) {

times.erase(key\_it);

}

// К каждому элементу добавить сумму минимального и максимального элементов контейнера

auto minmax = minmax\_element(times.begin(), times.end(), compare\_time);

auto sum = minmax.first->hours \* 60 + minmax.first->minutes + minmax.second->hours \* 60 + minmax.second->minutes;

for\_each(times.begin(), times.end(), [sum](Time& t) {t.hours = t.hours \* 60 + t.minutes + sum / 60; t.minutes = (t.minutes + sum % 60) % 60; t.hours = t.hours / 60; });

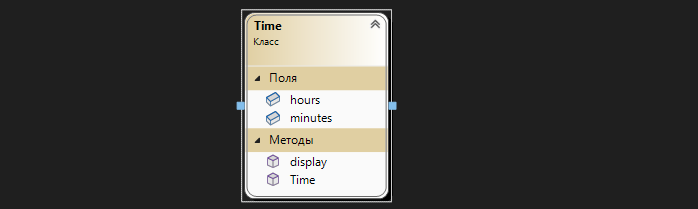
// Вывод

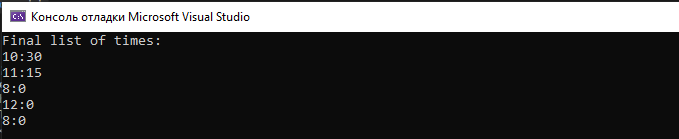
cout << "Final list of times:" << endl;

for\_each(times.begin(), times.end(), [](const Time& t) {t.display(); });

return 0;

}





**Задача 2**

#include <iostream>

#include <vector>

#include <algorithm>

struct Time {

int hours;

int minutes;

Time(int h, int m) : hours(h), minutes(m) {}

void print() const {

std::cout << hours << ":" << minutes << std::endl;

}

};

int main() {

std::vector<Time> times = { Time(10, 30), Time(9, 0), Time(12, 15), Time(8, 45) };

// Найти минимальный элемент и добавить его в конец контейнера

auto min\_time = std::min\_element(times.begin(), times.end(), [](const Time& a, const Time& b) {

return a.hours < b.hours || (a.hours == b.hours && a.minutes < b.minutes);

});

times.push\_back(\*min\_time);

// Найти элемент с заданным ключом и удалить его из контейнера

int key\_hours = 12;

int key\_minutes = 15;

auto it = std::find\_if(times.begin(), times.end(), [key\_hours, key\_minutes](const Time& t) {

return t.hours == key\_hours && t.minutes == key\_minutes;

});

if (it != times.end()) {

times.erase(it);

}

// К каждому элементу добавить сумму минимального и максимального элементов контейнера

auto minmax = std::minmax\_element(times.begin(), times.end(), [](const Time& a, const Time& b) {

return a.hours < b.hours || (a.hours == b.hours && a.minutes < b.minutes);

});

int min\_hours = minmax.first->hours;

int min\_minutes = minmax.first->minutes;

int max\_hours = minmax.second->hours;

int max\_minutes = minmax.second->minutes;

std::for\_each(times.begin(), times.end(), [min\_hours, min\_minutes, max\_hours, max\_minutes](Time& t) {

t.hours += min\_hours + max\_hours;

t.minutes += min\_minutes + max\_minutes;

});

// Выводим элементы вектора на экран

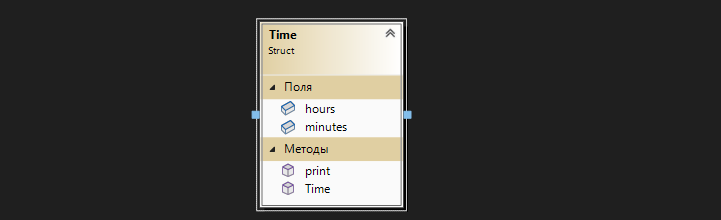
for (const auto& t : times) {

t.print();

}

return 0;

}





**Задача 3**

#include <iostream>

#include <set>

#include <algorithm>

struct Time {

int hours;

int minutes;

Time(int h, int m) : hours(h), minutes(m) {}

void print() const {

std::cout << hours << ":" << minutes << std::endl;

}

bool operator<(const Time& other) const {

return hours < other.hours || (hours == other.hours && minutes < other.minutes);

}

};

int main() {

std::set<Time> times = { Time(10, 30), Time(9, 0), Time(12, 15), Time(8, 45) };

// Найти минимальный элемент и добавить его в конец контейнера

auto min\_time = \*std::min\_element(times.begin(), times.end());

times.insert(min\_time);

// Найти элемент с заданным ключом и удалить его из контейнера

int key\_hours = 12;

int key\_minutes = 15;

auto it = std::find\_if(times.begin(), times.end(), [key\_hours, key\_minutes](const Time& t) {

return t.hours == key\_hours && t.minutes == key\_minutes;

});

if (it != times.end()) {

times.erase(it);

}

// К каждому элементу добавить сумму минимального и максимального элементов контейнера

auto minmax = std::minmax\_element(times.begin(), times.end());

int min\_hours = minmax.first->hours;

int min\_minutes = minmax.first->minutes;

int max\_hours = minmax.second->hours;

int max\_minutes = minmax.second->minutes;

std::for\_each(times.begin(), times.end(), [min\_hours, min\_minutes, max\_hours, max\_minutes](const Time& t) {

const\_cast<Time&>(t).hours += min\_hours + max\_hours;

const\_cast<Time&>(t).minutes += min\_minutes + max\_minutes;

});

// Выводим элементы множества на экран

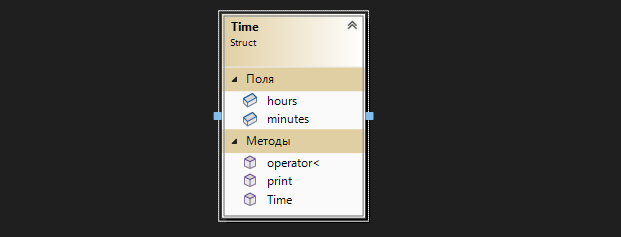
for (const auto& t : times) {

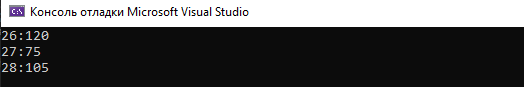
t.print();

}

return 0;

}

****

****

**Ответы на контрольные вопросы**