Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа по классам № 13(1, 2, 3)**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**семестр** 2

Выполнил студент гр. ИВТ-21-1б

Балдин Алексей Игоревич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь – 2022

**Цель**

Использование стандартных обобщенных алгоритмов из библиотеки STL в ОО программе.

**Постановка задачи**

* + - 1. Создать последовательный контейнер.

2. Заполнить его элементами пользовательского типа (тип указан в варианте). Для пользовательского типа перегрузить необходимые операции.

3. Заменить элементы в соответствии с заданием (использовать алгоритмы

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

4. Удалить элементы в соответствии с заданием (использовать алгоритмы

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

5. Отсортировать контейнер по убыванию и по возрастанию ключевого поля (использовать алгоритм sort()).

6. Найти в контейнере заданный элемент (использовать алгоритмы find(), find\_if(), count(), count\_if()).

7. Выполнить задание варианта для полученного контейнера (использовать алгоритм for\_each()) .

8 Для выполнения всех заданий использовать стандартные алгоритмы библиотеки STL.

**Исходные данные**

Вариант 5:

|  |  |  |
| --- | --- | --- |
| **Задача 1**   1. Контейнер - список 2. Тип элементов Time (см. лабораторную работу №3).   **Задача 2**  Адаптер контейнера - вектор.  **Задача 3**  Ассоциативный контейнер - множество | | |
| **Задание 3** | **Задание 4** | **Задание 5** |
| Найти минимальный элемент и добавить его на  заданную позицию контейнера | Найти элементы большие среднего  арифметического и удалить их из контейнера | Каждый элемент домножить на максимальный элемент контейнера |

**Разбор кода:**

В реализации (13.1) : Беру реализацию класса Time – из лаб.р. номер 3. Контейнером в моем варианте будет - список. Перегружаю все операторы, которые будут нужны в задаче, а также операторы ввода и вывода. Имеются методы создания списка, добавления минимального элемента в список, вывода списка, подсчета среднего элемента в списке, умножение каждого элемента списка на максимальный элемент из этого же списка.

Также, эти все действия, методы, взаимодействуют с функционалом библиотеки STL (push\_back, pop\_front, front, min\_element, max\_element, for\_each, begin, end).

Код программы (13.1):

Здесь представлен код в виде одного проекта (т.е. файлы cpp, .h – отдельно подключать не нужно), в GitHub - код представлен в виде трёх файлов – (cpp, .h, main).

#pragma once

#include <iostream>

using namespace std;

class Time

{

protected:

//àòòðèáóòû

int min;

int sec;

public:

//êîíñòðóêòîðû

Time(void);//áåç ïàðàìåòðîâ

Time(int, int);//ñ ïàðàìåòðàìè

Time(const Time&);//êîïèðîâàíèÿ

//äåñòðóêòîð

~Time(void);

//ìîäèôèêàòîðû

void set\_min(int);

void set\_sec(int);

//ñåëåêòîðû

int get\_min();

int get\_sec();

//ïåðåãðóçêà áèíàðíûõ îïåðàòîðîâ

Time& operator+ (const Time&);

Time& operator- (const Time&);

Time& operator\* (const Time&);

Time& operator/ (int);

Time& operator= (const Time&);

//ïåðåãðóçêà óíàðíûõ îïåðàòîðîâ

Time& operator++ ();

Time& operator-- ();

//ïåðåãðóçêà îïåðàòîðîâ ñðàâíåíèÿ

bool operator<(const Time&);

bool operator<=(const Time&);

bool operator>(const Time&);

bool operator>=(const Time&);

bool operator==(const Time&);

bool operator!=(const Time&);

//ïåðåãðóçêà îïåðàòîðîâ äëÿ ðàáîòû ñ âõîäíûì è âûõîäíûì ïîòîêàìè

friend ostream& operator<< (ostream&, const Time&);

friend istream& operator>> (istream&, Time&);

};

#include "Time.h"

//êîíñòðóêòîðû

Time::Time(void)//áåç ïàðàìåòðîâ

{

min = 0;

sec = 0;

}

Time::Time(int mn, int sc)//ñ ïàðàìåòðàìè

{

int buf = mn \* 60 + sc;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

}

Time::Time(const Time& t)//êîïèðîâàíèÿ

{

min = t.min;

sec = t.sec;

}

//äåñòðóêòîð

Time::~Time(void) = default;

//ìîäèôèêàòîðû

void Time::set\_min(int v\_min) { min = v\_min; }

void Time::set\_sec(int v\_sec)

{

int buf = min \* 60 + v\_sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

}

//ñåëåêòîðû

int Time::get\_min() { return min; }

int Time::get\_sec() { return sec; }

//ïåðåãðóçêà áèíàðíûõ îïåðàöèé

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

{

int buf = (min \* 60) + (t.min \* 60) + sec + t.sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator- (const Time & t)

{

int buf = (min \* 60 + sec) - (t.min \* 60 + t.sec);//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

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

{

int buf = (min \* 60 + sec) \* (t.min \* 60 + t.sec);

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator/ (int n)

{

int buf = (min \* 60 + sec) / n;

min = buf / 60;

sec = buf % 60;

return \*this;

}

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

{

min = t.min;

sec = t.sec;

return \*this;

}

//ïåðåãðóçêà óíàðíûõ îïåðàöèé

Time& Time::operator++ ()

{

int buf = min \* 60 + sec + 1;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator-- ()

{

int buf = min \* 60 + sec - 1;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

//ïåðåãðóçêà îïåðàòîðîâ ñðàâíåíèÿ

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

bool Time::operator!= (const Time & t)

{

if (min != t.min || sec != t.sec) return true;

return false;

}

//ïåðåãðóçêà îïåðàòîðîâ äëÿ ðàáîòû ñ âûõîäíûì è âõîäíûì ïîòîêàìè

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

{

out << t.min << ":" << t.sec;

return out;

}

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

{

int buf\_min;

int buf\_sec;

cout << "Enter min: "; in >> buf\_min;

cout << "Enter sec: "; in >> buf\_sec;

buf\_min = buf\_min \* 60 + buf\_sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

t.set\_min(buf\_min / 60);

t.set\_sec(buf\_min % 60);

return in;

}

#include <iostream>

#include <list>

#include <algorithm>

#include "Time.h"

using namespace std;

typedef list<Time> Tlist;

Time s;//äîáàâèì ãëîáàëüíóþ ïåðåìåííóþ äëÿ ñðàâíåíèÿ ñ çàäàííûì çíà÷åíèåì

//äîáàâèì ïðåäèêàò äëÿ ñðàâíåíèÿ çàäàííîãî çíà÷åíèÿ ñ îáúåòîì òèïà Time

struct Greater\_s

{

bool operator() (Time t)

{

if (t > s) return true;

else return false;

}

};

//ñîçäàíèå ñïèñêà

Tlist make\_list(int n)

{

Tlist l;

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

{

Time a;

cin >> a;

l.push\_back(a);

}

return l;

}

//ïå÷àòü ñïèñêà

void print\_list(Tlist& l)

{

Tlist buf;

int size = l.size();

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

{

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

buf.push\_back(l.front());

l.pop\_front();

}

cout << "\n";

l = buf;

}

//äîáàâëåíèå ìèí ýëåìåíòà íà çàäàííóþ ïîçèöèþ

void add\_min(Tlist& l, int n)

{

if (n > l.size() - 1)

{

cout << "Error! Position > size\n";

return;

}

Tlist::iterator i;

i = min\_element(l.begin(), l.end());

Time min = \*i;

Tlist buf;

for (int j = 0; j < n - 1; j++)

{

buf.push\_back(l.front());

l.pop\_front();

}

buf.push\_back(min);

int size = l.size();

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

{

buf.push\_back(l.front());

l.pop\_front();

}

l = buf;

}

//âû÷èñëåíèå ñðåäíåãî

Time averege(Tlist& l)

{

Tlist buf;

int size = l.size();

Time sum;

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

{

sum = sum + l.front();

buf.push\_back(l.front());

l.pop\_front();

}

l = buf;

return (sum / size);

}

//óìíîæèòü ýëåìåíò

void multi(Time& t)

{

t = t \* s;

}

int main()

{

int n;

cout << "Enter n >> "; cin >> n;

Tlist l;

l = make\_list(n);

cout << "Your list: "; print\_list(l);

cout << "\nEnter position for add min >> "; cin >> n;

add\_min(l, n);

cout << "Add min: "; print\_list(l);

s = averege(l);

cout << "\nAverage: " << s << "\n";

l.erase(remove\_if(l.begin(), l.end(), Greater\_s()), l.end());

cout << "Del elements more avereage: "; print\_list(l);

s = \*max\_element(l.begin(), l.end());

for\_each(l.begin(), l.end(), multi);

cout << "\nMultiple all elements: "; print\_list(l);

return 0;

}

Результат задачи 13.1:

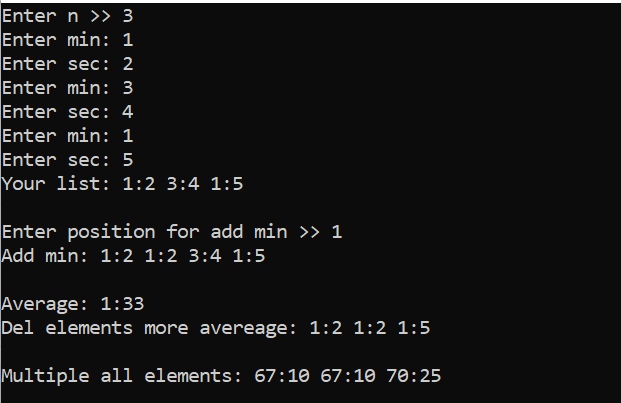
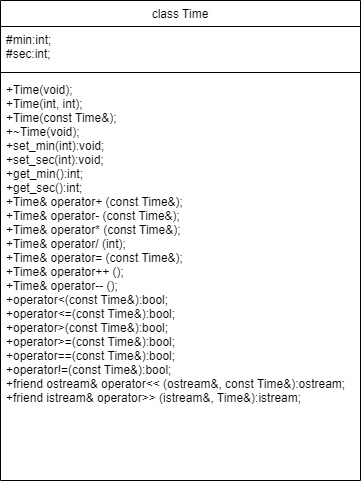


Диаграмма для 13.1:



В реализации (13.2) : Беру реализацию класса Time – из лаб.р. номер 3. Контейнером в моем варианте будет – список, но добавляется адаптер – вектор. Перегружаю все операторы, которые будут нужны в задаче, а также операторы ввода и вывода. Имеются методы создания списка, добавления минимального элемента в список, вывода списка, подсчета среднего элемента в списке, умножение каждого элемента списка на максимальный элемент из этого же списка.

Также, эти все действия, методы, взаимодействуют с функционалом библиотеки STL (push\_back, pop\_front, front, min\_element, max\_element, for\_each, begin, end). Также за счет появление адаптера – появляются методы (copy\_vector\_to\_list, copy\_list\_to\_vector)

Код программы 13.2:

(Аналогично 13.1, код в GitHub – представлен иначе)

#pragma once

#include <iostream>

using namespace std;

class Time

{

protected:

//àòòðèáóòû

int min;

int sec;

public:

//êîíñòðóêòîðû

Time(void);//áåç ïàðàìåòðîâ

Time(int, int);//ñ ïàðàìåòðàìè

Time(const Time&);//êîïèðîâàíèÿ

//äåñòðóêòîð

~Time(void);

//ìîäèôèêàòîðû

void set\_min(int);

void set\_sec(int);

//ñåëåêòîðû

int get\_min();

int get\_sec();

//ïåðåãðóçêà áèíàðíûõ îïåðàòîðîâ

Time& operator+ (const Time&);

Time& operator- (const Time&);

Time& operator\* (const Time&);

Time& operator/ (int);

Time& operator= (const Time&);

//ïåðåãðóçêà óíàðíûõ îïåðàòîðîâ

Time& operator++ ();

Time& operator-- ();

//ïåðåãðóçêà îïåðàòîðîâ ñðàâíåíèÿ

bool operator<(const Time&);

bool operator<=(const Time&);

bool operator>(const Time&);

bool operator>=(const Time&);

bool operator==(const Time&);

bool operator!=(const Time&);

//ïåðåãðóçêà îïåðàòîðîâ äëÿ ðàáîòû ñ âõîäíûì è âûõîäíûì ïîòîêàìè

friend ostream& operator<< (ostream&, const Time&);

friend istream& operator>> (istream&, Time&);

};

#include "Time.h"

//êîíñòðóêòîðû

Time::Time(void)//áåç ïàðàìåòðîâ

{

min = 0;

sec = 0;

}

Time::Time(int mn, int sc)//ñ ïàðàìåòðàìè

{

int buf = mn \* 60 + sc;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

}

Time::Time(const Time& t)//êîïèðîâàíèÿ

{

min = t.min;

sec = t.sec;

}

//äåñòðóêòîð

Time::~Time(void) = default;

//ìîäèôèêàòîðû

void Time::set\_min(int v\_min) { min = v\_min; }

void Time::set\_sec(int v\_sec)

{

int buf = min \* 60 + v\_sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

}

//ñåëåêòîðû

int Time::get\_min() { return min; }

int Time::get\_sec() { return sec; }

//ïåðåãðóçêà áèíàðíûõ îïåðàöèé

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

{

int buf = (min \* 60) + (t.min \* 60) + sec + t.sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator- (const Time & t)

{

int buf = (min \* 60 + sec) - (t.min \* 60 + t.sec);//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

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

{

int buf = (min \* 60 + sec) \* (t.min \* 60 + t.sec);

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator/ (int n)

{

int buf = (min \* 60 + sec) / n;

min = buf / 60;

sec = buf % 60;

return \*this;

}

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

{

min = t.min;

sec = t.sec;

return \*this;

}

//ïåðåãðóçêà óíàðíûõ îïåðàöèé

Time& Time::operator++ ()

{

int buf = min \* 60 + sec + 1;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator-- ()

{

int buf = min \* 60 + sec - 1;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

//ïåðåãðóçêà îïåðàòîðîâ ñðàâíåíèÿ

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

bool Time::operator!= (const Time & t)

{

if (min != t.min || sec != t.sec) return true;

return false;

}

//ïåðåãðóçêà îïåðàòîðîâ äëÿ ðàáîòû ñ âûõîäíûì è âõîäíûì ïîòîêàìè

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

{

out << t.min << ":" << t.sec;

return out;

}

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

{

int buf\_min;

int buf\_sec;

cout << "Enter min: "; in >> buf\_min;

cout << "Enter sec: "; in >> buf\_sec;

buf\_min = buf\_min \* 60 + buf\_sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

t.set\_min(buf\_min / 60);

t.set\_sec(buf\_min % 60);

return in;

}

#include <iostream>

#include <list>

#include <vector>

#include <algorithm>

#include "Time.h"

using namespace std;

typedef list<Time> Tlist;

typedef vector<Time> Tvector;

Time s;//äîáàâèì ãëîáàëüíóþ ïåðåìåííóþ äëÿ ñðàâíåíèÿ ñ çàäàííûì çíà÷åíèåì

//äîáàâèì ïðåäèêàò äëÿ ñðàâíåíèÿ çàäàííîãî çíà÷åíèÿ ñ îáúåòîì òèïà Time

struct Greater\_s

{

bool operator() (Time t)

{

if (t > s) return true;

else return false;

}

};

//ñîçäàòü âåêòîð

Tvector make\_vector(int n)

{

Tvector v;

Time t;

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

{

cin >> t;

v.push\_back(t);

}

return v;

}

//êîïèðîâàòü âåêòîð â ñïèñîê

Tlist copy\_vector\_to\_list(Tvector& v)

{

Tlist l;

int size = v.size();

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

l.push\_back(v[i]);

return l;

}

//êîïèðîâàòü ñïèñîê â âåêòîð

Tvector copy\_list\_to\_vector(Tlist l)

{

Tvector v;

int size = l.size();

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

{

v.push\_back(l.front());

l.pop\_front();

}

return v;

}

//âûâîä âåêòîðà

void print\_vector(Tvector v)

{

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

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

cout << "\n";

}

//äîáàâëåíèå ìèí ýëåìåíòà íà çàäàííóþ ïîçèöèþ

void add\_min(Tlist& l, int n)

{

if (n > l.size() - 1)//åñëè ïîçèöèÿ áîëüøå ðàçìåðà

{

cout << "Error! Position > size\n";

return;

}

Tlist::iterator i;

i = min\_element(l.begin(), l.end());//ïîèñê ìèí. ýëåìåíòà

Time min = \*i;

Tlist buf;

for (int j = 0; j < n - 1; j++)//êîïèðóåì ñïèñîê â áóôåðíûé äî n

{

buf.push\_back(l.front());

l.pop\_front();

}

buf.push\_back(min);//äîáàâëÿåì íà ïîçèöèþ n ýëåìåíò

int size = l.size();

for (int i = 0; i < size; i++)//ïðîäîëæàåì çàïîëíÿòü ëèñò

{

buf.push\_back(l.front());

l.pop\_front();

}

l = buf;

}

//âû÷èñëåíèå ñðåäíåãî

Time averege(Tlist& l)

{

Tlist buf;

int size = l.size();

Time sum;

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

{

sum = sum + l.front();

buf.push\_back(l.front());

l.pop\_front();

}

l = buf;

return (sum / size);

}

//óìíîæèòü ýëåìåíò

void multi(Time& t)

{

t = t \* s;

}

int main()

{

int n;

cout << "Enter n >> "; cin >> n;

Tvector v = make\_vector(n);

print\_vector(v);

cout << "\nEnter position for add min >> "; cin >> n;

Tlist temp = copy\_vector\_to\_list(v);

add\_min(temp, n);

v = copy\_list\_to\_vector(temp);

cout << "Add min: "; print\_vector(v);

temp = copy\_vector\_to\_list(v);

s = averege(temp);

cout << "\nAvereage: " << s << "\n";

temp.erase(remove\_if(temp.begin(), temp.end(), Greater\_s()), temp.end());

v = copy\_list\_to\_vector(temp);

cout << "Del elements more avereage: "; print\_vector(v);

temp = copy\_vector\_to\_list(v);

s = \*max\_element(temp.begin(), temp.end());

for\_each(temp.begin(), temp.end(), multi);

v = copy\_list\_to\_vector(temp);

cout << "\nMultiple all elements: "; print\_vector(v);

return 0;

}

Результат к программе 13.2:

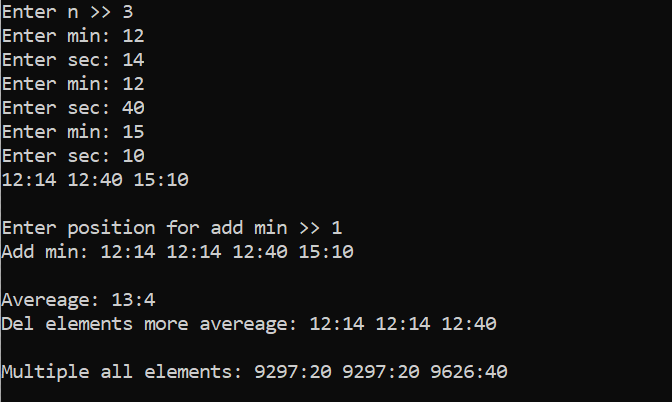
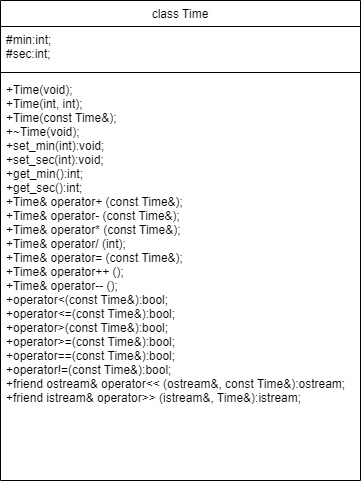


Диаграмма к программе 13.2:



В реализации (13.3) : Беру реализацию класса Time – из лаб.р. номер 3. Контейнером в моем варианте будет – список, но добавляется адаптер – вектор. Перегружаю все операторы, которые будут нужны в задаче, а также операторы ввода и вывода. Имеются методы создания списка, добавления минимального элемента в список, вывода списка, подсчета среднего элемента в списке, умножение каждого элемента списка на максимальный элемент из этого же списка.

Также, эти все действия, методы, взаимодействуют с функционалом библиотеки STL (push\_back, pop\_front, front, min\_element, max\_element, for\_each, insert, begin, end). Также за счет появление ассоциативного контейнера – множество, появляются методы (greater\_s, min\_pred, max\_pred, del\_more\_aver)

Код программы 13.3:

(Аналогично ситуация с кодом, как и в прошлых задачах).

#pragma once

#include <iostream>

using namespace std;

class Time

{

protected:

//àòòðèáóòû

int min;

int sec;

public:

//êîíñòðóêòîðû

Time(void);//áåç ïàðàìåòðîâ

Time(int, int);//ñ ïàðàìåòðàìè

Time(const Time&);//êîïèðîâàíèÿ

//äåñòðóêòîð

~Time(void);

//ìîäèôèêàòîðû

void set\_min(int);

void set\_sec(int);

//ñåëåêòîðû

int get\_min();

int get\_sec();

//ïåðåãðóçêà áèíàðíûõ îïåðàòîðîâ

Time& operator+ (const Time&);

Time& operator- (const Time&);

Time& operator\* (const Time&);

Time& operator/ (int);

Time& operator= (const Time&);

//ïåðåãðóçêà óíàðíûõ îïåðàòîðîâ

Time& operator++ ();

Time& operator-- ();

//ïåðåãðóçêà îïåðàòîðîâ ñðàâíåíèÿ

bool operator<(const Time&);

bool operator<=(const Time&);

bool operator>(const Time&);

bool operator>=(const Time&);

bool operator==(const Time&);

bool operator!=(const Time&);

//ïåðåãðóçêà îïåðàòîðîâ äëÿ ðàáîòû ñ âõîäíûì è âûõîäíûì ïîòîêàìè

friend ostream& operator<< (ostream&, const Time&);

friend istream& operator>> (istream&, Time&);

};

#include "Time.h"

//êîíñòðóêòîðû

Time::Time(void)//áåç ïàðàìåòðîâ

{

min = 0;

sec = 0;

}

Time::Time(int mn, int sc)//ñ ïàðàìåòðàìè

{

int buf = mn \* 60 + sc;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

}

Time::Time(const Time& t)//êîïèðîâàíèÿ

{

min = t.min;

sec = t.sec;

}

//äåñòðóêòîð

Time::~Time(void) = default;

//ìîäèôèêàòîðû

void Time::set\_min(int v\_min) { min = v\_min; }

void Time::set\_sec(int v\_sec)

{

int buf = min \* 60 + v\_sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

}

//ñåëåêòîðû

int Time::get\_min() { return min; }

int Time::get\_sec() { return sec; }

//ïåðåãðóçêà áèíàðíûõ îïåðàöèé

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

{

int buf = (min \* 60) + (t.min \* 60) + sec + t.sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator- (const Time & t)

{

int buf = (min \* 60 + sec) - (t.min \* 60 + t.sec);//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

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

{

int buf = (min \* 60 + sec) \* (t.min \* 60 + t.sec);

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator/ (int n)

{

int buf = (min \* 60 + sec) / n;

min = buf / 60;

sec = buf % 60;

return \*this;

}

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

{

min = t.min;

sec = t.sec;

return \*this;

}

//ïåðåãðóçêà óíàðíûõ îïåðàöèé

Time& Time::operator++ ()

{

int buf = min \* 60 + sec + 1;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

Time& Time::operator-- ()

{

int buf = min \* 60 + sec - 1;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

min = buf / 60;

sec = buf % 60;

return \*this;

}

//ïåðåãðóçêà îïåðàòîðîâ ñðàâíåíèÿ

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

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

{

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

return false;

}

bool Time::operator!= (const Time & t)

{

if (min != t.min || sec != t.sec) return true;

return false;

}

//ïåðåãðóçêà îïåðàòîðîâ äëÿ ðàáîòû ñ âûõîäíûì è âõîäíûì ïîòîêàìè

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

{

out << t.min << ":" << t.sec;

return out;

}

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

{

int buf\_min;

int buf\_sec;

cout << "Enter min: "; in >> buf\_min;

cout << "Enter sec: "; in >> buf\_sec;

buf\_min = buf\_min \* 60 + buf\_sec;//åñëè ñåêóíä áîëüøå 60, òî ìû èõ äîëæíû ïåðåâåñòè â ìèíóòû

t.set\_min(buf\_min / 60);

t.set\_sec(buf\_min % 60);

return in;

}

#include <iostream>

#include <map>

#include <algorithm>

#include "Time.h"

using namespace std;

typedef map<int, Time> Tmap;

typedef map<int, Time>::iterator it;

Time s;//äîáàâèì ãëîáàëüíóþ ïåðåìåííóþ äëÿ ñðàâíåíèÿ ñ çàäàííûì çíà÷åíèåì

bool greater\_s(Time a)//äîáàâèì ïðåäèêàò äëÿ ñðàâíåíèÿ çàäàííîãî çíà÷åíèÿ ñ îáúåòîì òèïà Time

{

return (a > s);

}

bool min\_pred(pair<int, Time> a, pair<int, Time> b)//ïðåäèêàò äëÿ ïîèñêà ìèíèìàëüíîãî ýëåìåíòà

{

return (a.second < b.second);

}

bool max\_pred(pair<int, Time> a, pair<int, Time> b)//ïðåäèêàò äëÿ ïîèñêà ìàêñèìàëüíîãî ýëåìåíòà

{

return (a.second > b.second);

}

//ñîçäàòü ñëîâàðü

Tmap make\_map(int n)

{

Tmap m;

Time t;

for (int i = 1; i <= n; i++)

{

cin >> t;

m.insert(make\_pair(i, t));

}

return m;

}

//âûâîä ñëîâàðÿ

void print\_map(Tmap m)

{

it i = m.begin();

while (i != m.end())

{

cout << i->first << ") " << i->second << ", ";

i++;

}

cout << "\n";

}

//äîáàâèòü ìèíèìàëüíûé ýëåìåíò â ñëîâàðü

void add\_min(Tmap& m, int n)

{

it i = m.begin();

Tmap buf;

Time min = min\_element(m.begin(), m.end(), min\_pred)->second;//ïîèñê ìèí ýëåìåíòà

for (int j = 1; j < n; j++)//çàïîëíÿåì áóô ñëîâàðá äî êëþ÷à n

{

buf.insert(make\_pair(j, i->second));

i++;

}

buf.insert(make\_pair(n, i->second));//äîáàâëÿåì ïîä êëþ÷¸ì n íîâûé ýëåìåíò

i++;

int k = n + 1;

while (i != m.end())//ïðîäîëæàåì çàïîëíÿòü áóô ñëîâàðü

{

buf.insert(make\_pair(k, i->second));

k++;

i++;

}

m = buf;

}

//âû÷èñëåíèå ñðåäíåãî

Time averege(Tmap m)

{

it i = m.begin();

Time sum;

int size = m.size();

while (i != m.end())

{

sum = sum + i->second;

i++;

}

return (sum / size);

}

//óäàëèòü ýëåìåíòû áîëüøå ñðåäíåãî

void del\_more\_aver(Tmap& m)

{

Tmap buf;

int size = m.size();

int k = 1;

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

if (!greater\_s(m[i]))

{

buf.insert(make\_pair(k, m[i]));

k++;

}

m = buf;

}

//óìíîæèòü ýëåìåíò

void multi(pair<const int, Time>& a)

{

a.second = a.second \* s;

}

int main()

{

int n;

cout << "Enter n >> "; cin >> n;

Tmap m = make\_map(n);

cout << "Your map: "; print\_map(m);

cout << "\nEnter position for add min >> "; cin >> n;

add\_min(m, n);

cout << "Add min: "; print\_map(m);

s = averege(m);

cout << "\nAvereage: " << s << "\n";

del\_more\_aver(m);

cout << "Del elements more avereage: "; print\_map(m);

s = max\_element(m.begin(), m.end(), max\_pred)->second;

for\_each(m.begin(), m.end(), multi);

cout << "\nMultiple all elements: "; print\_map(m);

return 0;

}

Результат программы 13.3:

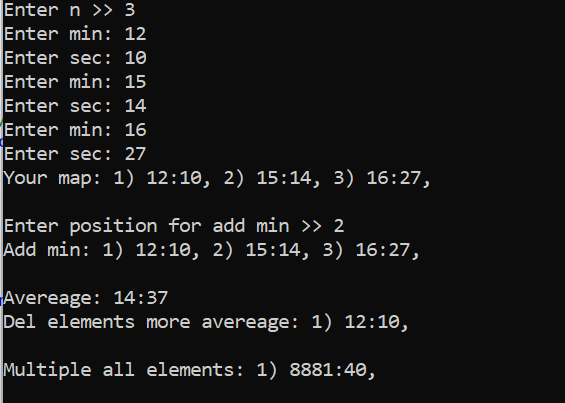


Диаграмма к задаче 13.3:

