

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

ЛАБОРАТОРНАЯ РАБОТА

№1

**по курсу объектно-ориентированное программирование I семестр, 2021/22
уч. год**

Студент: Макаров Глеб Александрович, группа М8О-207Б-20

Преподаватель: Дорохов Евгений Павлович

Условие

Задание: Вариант 14: Создать класс TimePoint для работы с моментами времени в формате «час:минута:секунда». Обязательными операциями являются: вычисление разницы между двумя моментами времени, сумма моментов времени, сложение момента времени и заданного количества секунд, вычитание из момента времени заданного количества секунд, вычисление во раз сколько один момент времени больше (меньше) другого, сравнение моментов времени, перевод в секунды и обратно, перевод в минуты (с округлением до минуты) и обратно. Реализовать над объектами реализовать в виде перегрузки операторов. Реализовать пользовательский литерал для работы с константами объектов созданного класса.

Описание программы

Исходный код лежит в 3 файлах:

1. main.cpp: тестирование кода
2. time.h: описание класса TimePoint
3. time.cpp: реализация класса TimePoint

Дневник отладки

Ошибок не наблюдалось.

Недочёты

Недочётов не заметил.

Вывод

В данной лабораторной работе была написан код для понимания ООП, ничего сложного, но помогает вникнуть в данную парадигму.

Исходный код

main.cpp

```
#include <iostream>
```

```
#include "time.h"
```

```
#include <fstream>
```

```
int main() {
```

```
    std::ifstream file("test_01.txt");
```

```
    int size;
```

```
    TimePoint n = 1_sec;
```

```
    file >> size;
```

```
    int count;
```

```
    TimePoint m;
```

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

```
        file >> m;
```

```
        std::cout << "\nSource data"<< std::endl << m.sec();
```

```
        m = m + n;
```

```
        std::cout << "\n+ 00.00.1"<< std::endl << m.sec();
```

```
        m = m - n;
```

```
        std::cout << "\n- 00.00.1"<< std::endl << m.sec();
```

```
        int t = m / n;
```

```
        std::cout << "\n/00.00.1"<< std::endl << t;
```

```
    }
```

```
    return 0;
```

```
}
```

time.h

```

#ifndef TIME_H
#define TIME_H
#include <iostream>
#include <string>
#include <cstring>

class TimePoint
{
public:
    TimePoint();
    TimePoint(const TimePoint &m);
    TimePoint(const int sec, const int min, const int hour);

    TimePoint operator+(int sec);
    TimePoint operator-(int sec);
    void reFresh();
    TimePoint operator+(const TimePoint &number);
    TimePoint operator-(const TimePoint &number);

    int operator/(const TimePoint &number);
    TimePoint operator%(const TimePoint &number);

    bool operator==(const TimePoint &number);
    bool operator>(const TimePoint &number);
    bool operator<(const TimePoint &number);

    int sec();
    int min();

```

```
friend std::istream &operator>>(std::istream &is, TimePoint &object);
```

```
private:
```

```
int sec_;
```

```
int minutes_;
```

```
int hour_;
```

```
};
```

```
TimePoint operator "" _sec(unsigned long long time);
```

```
TimePoint operator "" _min(unsigned long long time);
```

```
TimePoint operator "" _hour(unsigned long long time);
```

```
#endif
```

time.cpp

```
#include "time.h"
```

```
TimePoint::TimePoint(): hour_(0), minutes_(0), sec_(0) {}
```

```
TimePoint::TimePoint(const TimePoint &m): hour_(m.hour_), minutes_(m.minutes_),  
sec_(m.sec_) {}
```

```
TimePoint::TimePoint(const int hour, const int min, const int sec): hour_(hour), minutes_(min),  
sec_(sec) {}
```

```
void TimePoint::reFresh(){
```

```
if (sec_>=60) {
```

```
minutes_ += sec_/60;
```

```
sec_ = sec_%60;
```

```

    if (minutes_>=60) {
        hour_+= minutes_/60;
        minutes_ = minutes_%60;
        if (hour_>=24) hour_ = hour_%24;
    }
}

TimePoint TimePoint::operator+(const int sec){
    sec_ = sec_ + sec;
    reFresh();
    return *this;
}

int TimePoint:: sec(){
    return hour_*3600+minutes_*60+sec_;
}

int TimePoint:: min (){
    return hour_*60+minutes_;
}

TimePoint TimePoint::operator-(int sec){

    if(sec < sec_+minutes_*60+hour_*3600){
        sec_+= minutes_*60+hour_*3600;
        sec_ = sec_ - sec;
        reFresh();
    }
    return *this;
}

```

```

}

int TimePoint::operator/(const TimePoint &m){
    int num = 0;
    int s = sec_+minutes_*60+hour_*3600;
    int ms = m.sec_+m.minutes_*60+m.hour_*3600;
    if((s >= ms) && (ms > 0)) {
        num = s / ms;
    }
    if((ms > s) && (s > 0)) {
        num = s / ms;
    }

    return num;
}

TimePoint TimePoint::operator-(const TimePoint &m){
    sec_ = sec_+minutes_*60+hour_*3600;
    int ms = m.sec_+m.minutes_*60+m.hour_*3600;
    if(sec_ < ms) sec_ = ms - sec_;
    else sec_ = sec_ - ms;
    hour_ = 0;
    minutes_ = 0;
    reFresh();
    return *this;
}

bool TimePoint::operator>(const TimePoint &m){
    return sec_+minutes_*60+hour_*3600 > m.sec_+m.minutes_*60+m.hour_*3600;
}

bool TimePoint::operator==(const TimePoint &m){
    return sec_+minutes_*60+hour_*3600 == m.sec_+m.minutes_*60+m.hour_*3600;
}

```

```

}

bool TimePoint::operator<(const TimePoint &m){
    return sec_+minutes_*60+hour_*3600 < m.sec_+m.minutes_*60+m.hour_*3600;
}

TimePoint TimePoint::operator+(const TimePoint &m){
    sec_ = sec_+minutes_*60+hour_*3600;
    int ms = 0;
    ms = m.sec_+m.minutes_*60+m.hour_*3600;
    sec_ = sec_ + ms;

    return *this;
}

std::istream &operator>>(std::istream &is, TimePoint &object){
    is >> object.sec_;
    return is;
}

TimePoint operator "" _sec(unsigned long long time){
    TimePoint a(0,0,time);
    a.reFresh();
    return a;
}

TimePoint operator "" _min(unsigned long long time){
    TimePoint a(0,time,0);
    a.reFresh();
    return a;
}

TimePoint operator "" _hour(unsigned long long time){
    TimePoint a(time,0,0);

```



```
a.refresh();  
return a;  
}
```