

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

ЛАБОРАТОРНАЯ РАБОТА №2 по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент Морозов Артем Борисович, группа М80-208Б-20
Преподаватель Дорохов Евгений Павлович

Цель:

- Изучение основ работы с классами в С++;
- Перегрузка операций и создание литералов

Требования к программе

Разработать программу на языке С++ согласно варианту задания. Программа на С++ должна собираться с помощью системы сборки CMake. Программа должна

получать данные из стандартного ввода и выводить данные в стандартный вывод.

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 - исполняемый код.
2. TimePoint.h - специальный файл .h, содержащий прототипы используемых мною функций.
3. TimePoint.cpp - реализация функций для моего задания.
4. CMakeLists.txt - специальный дополнительный файл типа CMakeLists.

Дневник отладки

Во время выполнения лабораторной работы программа не нуждалась в отладке, все ошибки компиляции были исправлены с первой попытки. После их исправления программа работала так, как было задумано изначально.

Недочёты

Недочётов не было обнаружено.

Выводы

Лабораторная работа №2 - это, по сути, та же самая лабораторная №1, только предусматривающая возможность перегрузки операторов. Лабораторная была выполнена успешно, в ее процессе были еще раз осознаны основные принципы ООП и перегрузки операторов.

Исходный код

TimePoint.h

```
#ifndef TIMEPOINT_H
#define TIMEPOINT_H
#include <iostream>
class TimePoint {
public:
    TimePoint();
    TimePoint(int h, int m, int s);
    TimePoint(const TimePoint &other);
    void AddSeconds(int s);
    void RemoveSeconds(int s);
    int IsBigger(const TimePoint &other);
    int ToSeconds();
    int ToMinutes();
    TimePoint operator + (const TimePoint &object);
    TimePoint operator - (const TimePoint &object);
    bool operator == (const TimePoint &object);
    bool operator > (const TimePoint &other);
    bool operator < (const TimePoint &other);
    friend std::istream& operator >> (std::istream& is, TimePoint &object);
    friend std::ostream& operator <<(std::ostream& os, TimePoint &object);
    ~TimePoint();
private:
    int hours;
    int minutes;
    int seconds;
};
#endif
```

TimePoint.cpp

```
#include "TimePoint.h"
```

```
TimePoint::TimePoint() {  
    hours = 0;  
    minutes = 0;  
    seconds = 0;  
    std::cout << "The default time-object has been created" << std::endl;  
}
```

```
TimePoint::TimePoint(int h, int m, int s) {  
    if (h >= 0 && m >= 0 && s >= 0) {  
        hours = h;  
        minutes = m;  
        seconds = s;  
    }  
    else {  
        std::cout << "Please enter positive numbers!" << std::endl;  
    }  
    std::cout << "The time-object according to your parameters has been created" << std::endl;  
}
```

```
TimePoint::TimePoint(const TimePoint& other) {  
    hours = other.hours;  
    minutes = other.minutes;  
    seconds = other.seconds;  
    std::cout << "The copy of your time-object has been created" << std::endl;  
}
```

```
void TimePoint::AddSeconds(int s) {  
    if (s < 0) {  
        std::cout << "Please enter positive number!" << std::endl;  
    }  
    else {  
        int x = hours * 3600 + minutes * 60 + seconds + s;  
        hours = x / 3600;  
        minutes = ((x % 3600) / 60);  
        seconds = (x % 3600) - (((x % 3600) / 60) * 60);  
        std::cout << "After adding seconds your time is: " << hours << ":" << minutes << ":" <<  
seconds << std::endl;  
    }  
}
```

```

    }
}

void TimePoint::RemoveSeconds(int s) {
    if (s < 0) {
        std::cout << "Please enter positive number!" << std::endl;
    }
    else {
        int x = hours * 3600 + minutes * 60 + seconds - s;
        hours = x / 3600;
        minutes = ((x % 3600) / 60);
        seconds = (x % 3600) - (((x % 3600) / 60) * 60);
        std::cout << "After removing seconds your time is: " << hours << ":" << minutes << ":" <<
seconds << std::endl;
    }
}

int TimePoint::IsBigger(const TimePoint &other) {
    int x = hours * 3600 + minutes * 60 + seconds;
    int y = other.hours * 3600 + other.minutes * 60 + other.seconds;
    if ((hours > other.hours) || (hours == other.hours && minutes > other.minutes) || (hours ==
other.hours && minutes == other.minutes && seconds > other.seconds)) {
        return x / y;
    }
    return y / x;
}

int TimePoint::ToSeconds() {
    return hours * 3600 + minutes * 60 + seconds;
}

int TimePoint::ToMinutes() {
    int z = hours * 3600 + minutes * 60 + seconds;
    int m = z / 60;
    if (z % 60 == 0) {
        return m;
    }
    else {
        if (z % 60 >= 30) {
            return m + 1;
        }
    }
    return m;
}

TimePoint TimePoint::operator + (const TimePoint &object) {
    int x = hours * 3600 + minutes * 60 + seconds;
    int y = object.hours * 3600 + object.minutes * 60 + object.seconds;
    int z = x + y;
    int dhours = z / 3600;
    int dminutes = (z % 3600) / 60;

```

```

    int dseconds = (z % 3600) - (dminutes * 60);
    this->hours = dhours;
    this->minutes = dminutes;
    this->seconds = dseconds;
    return *this;
}

TimePoint TimePoint::operator - (const TimePoint &object) {
    int x = hours * 3600 + minutes * 60 + seconds;
    int y = object.hours * 3600 + object.minutes * 60 + object.seconds;
    int dhours, dminutes, dseconds;
    if ((hours > object.hours) || (hours == object.hours && minutes > object.minutes) || (hours ==
object.hours && minutes == object.minutes && seconds > object.seconds)) {
        int z = x - y;
        dhours = z / 3600;
        dminutes = (z % 3600) / 60;
        dseconds = (z % 3600) - (dminutes * 60);
    }
    else {
        int z = y - x;
        dhours = z / 3600;
        dminutes = (z % 3600) / 60;
        dseconds = (z % 3600) - (dminutes * 60);
    }
    this->hours = dhours;
    this->minutes = dminutes;
    this->seconds = dseconds;
    return *this;
}

bool TimePoint::operator == (const TimePoint &object) {
    if (hours == object.hours && minutes == object.minutes && seconds == object.seconds) {
        return true;
    }
    return false;
}

bool TimePoint::operator > (const TimePoint &other) {
    if ((hours > other.hours) || (hours == other.hours && minutes > other.minutes) || (hours ==
other.hours && minutes == other.minutes && seconds > other.seconds)) {
        return true;
    }
    else {
        return false;
    }
}

bool TimePoint::operator < (const TimePoint &other) {
    if ((hours < other.hours) || (hours == other.hours && minutes < other.minutes) || (hours ==
other.hours && minutes == other.minutes && seconds < other.seconds)) {
        return true;
    }
}

```

```

    }
    else {
        return false;
    }
}

std::istream& operator >> (std::istream& is, TimePoint &object) {
    std::cout << "Please enter your time-object data: " << std::endl;
    is >> object.hours >> object.minutes >> object.seconds;
    if ((object.hours < 0 || object.hours > 23) || (object.minutes < 0 || object.minutes > 59) ||
        (object.seconds < 0 || object.seconds > 59)) {
        std::cout << "Invalid input. Enter again!" << std::endl;
        is >> object.hours >> object.minutes >> object.seconds;
    }
    return is;
}

std::ostream& operator << (std::ostream& os, TimePoint &object) {
    os << object.hours << ":" << object.minutes << ":" << object.seconds << std::endl;
    return os;
}

TimePoint::~TimePoint() {
    std::cout << "FROM DESTRUCTOR: Your time-object has been deleted" << std::endl;
}

```

main.cpp

```

#include "TimePoint.h"

//literals
unsigned long long operator "" _tohours(unsigned long long sec) {
    unsigned long long hours_lit = sec / 3600;
    return hours_lit;
}

unsigned long long operator "" _tominutes(unsigned long long sec) {
    unsigned long long minutes_lit = sec / 60;
    return minutes_lit;
}

int main () {
    TimePoint a(15,0,5);
    TimePoint b(15,0,5);
}

```



```

std::cout << (b == a) << std::endl;
std::cout << (a > b) << std::endl;
std::cout << (b < a) << std::endl;
TimePoint c = a + b;
std::cout << c;
TimePoint d;
std::cin >> d;
TimePoint e = d - b;
std::cout << e;
TimePoint x(23,59,59);
TimePoint y = x - a;
TimePoint j, h, u;
std::cin >> j >> h >> u;
std::cout << j << h << u;
std::cout << "The example of using to-hours literal is: " << 3600_tohours <<
std::endl;
    std::cout << "The example of using to-minutes literal is: " << 50505_tominutes
<< std::endl;
    return 0;
}

```