Московский авиационный институт
(Национальный исследовательский университет)

Институт: 8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 2

Тема: Перегрузка операторов в C++

Студент: Моисеенков Илья Павлович
Группа: М8О-208Б-19
Преподаватель: Чернышов Л.Н.
Дата: 12.10.2020
Оценка: 15/15

Москва, 2020

1. Постановка задачи

a. Ознакомиться с теоретическим материалом по перегрузке операторов в C++
b. Создать класс **TimePoint** для работы с моментами времени в формате "час:минута:секунда". Реализовать операции вычисления разницы между двумя моментами времени, суммы моментов времени, сложения момента времени и заданного количества секунд, вычисления во сколько раз один момент больше (меньше) другого, сравнения моментов времени, перевода в секунды и обратно, перевода в минуты и обратно. Операции сложения, вычитания и сравнения реализовать в виде перегрузки операторов. Реализовать пользовательский литерал для работы с константами типа TimePoint.
c. Настроить CMake файл для сборки программы
d. Продумать варианты тестирования работы методов класса и подготовить тестовые данные
e. Отправить файлы лабораторной работы в репозиторий Github
f. Подготовить отчет о лабораторной работе

2. Описание программы

Программа имеет многофайловую структуру. Описание класса и реализация его методов выделены в отдельные файлы.

TimePoint.h / TimePoint.cpp

TimePoint.h - это заголовочный файл, содержащий описание класса и определения всех методов класса. Файл TimePoint.cpp содержит реализацию всех методов.

Элементы класса:

- атрибуты `hours, min, sec,` показывающие текущий момент времени (часы, минуты, секунды)
- конструкторы: принимающий три аргумента - часы, минуты, секунды и не принимающий аргументов - зануляет все атрибуты.
- приватный метод `check_data,` проверяющий полученный момент времени. Если обнаружен отрицательный атрибут класса, то программа выводит сообщение об ошибке и завершает работу с кодом 1. Если число секунд больше 60, то лишняя часть переводится в минуты. Аналогично лишние минуты переводятся в часы.
- геттеры: `get_hours(), get_min(), get_sec().` Возвращают соответствующие приватные значения.
- сеттеры: `set_hours(), set_min(), set_sec().` Устанавливают соответствующие значения.
- метод `print_time,` выводящий момент времени в консоль в формате “часы:минуты:секунды”

- методы `to_minutes`, `to_seconds`, конвертирующие текущий момент времени в минуты и секунды соответственно
- методы `set_by_seconds`, `set_by_minutes`, конвертирующие секунды и минуты в полноценный объект класса.
- операторы сравнения. Два момента времени сравниваются сначала по часам, затем по минутам и по секундам.
- оператор сложения двух моментов времени. Каждый атрибут одного класса суммируется с соответствующим атрибутом второго класса. Затем производится проверка методом `check_data`.
- оператор сложения момента времени и целого числа секунд. Выполняется аналогично предыдущему методу.
- оператор разности двух моментов времени. Возвращает абсолютное значение разности моментов. Вычисляется аналогично методам выше
- оператор вычитания целого числа секунд из класса. Если вычитаемое больше уменьшаемого, то программа выводит сообщение об ошибке и завершает работу с кодом 1.
- оператор деления, показывающий, во сколько раз один момент времени больше другого. Оба аргумента переводятся в секунды, возвращается отношение полученных чисел.

main.cpp

Файл содержит определение пользовательского литерала и функцию main, тестирующую работу класса. Пользовательский литерал имеет вид “секунды_time” и создает объект TimePoint с заданным количеством секунд.

Функция main содержит заготовленный заранее набор операций, демонстрирующий работу всех методов класса на примере двух объектов. Для создания двух моментов времени пользователю предлагается ввести 6 неотрицательных чисел: часы, минуты, секунды для первого момента и для второго. Затем программа выполняет различные действия с этими объектами и выводит информацию о каждом выполненном действии на экран. Пользователь может не вводить данные, а указать название файла с заранее подготовленными данными.

3. Набор тестов

Один тестовый набор включает в себя 6 чисел: часы, минуты и секунды первого момента времени и аналогично для второго момента.

test1.txt

3 30 50

13 40 30

test2.txt

0 0 12842792484

12 23213 324234

test3.txt

10 34 12

0 0 -12

4. Результаты выполнения тестов

test1

```
Enter the name of test file (e.g. test1.txt) or type skip to enter
data yourself:test1.txt

We have time1: 3:30:50
And time2: 13:40:30

Sum of time moments: 17:11:20
Difference between times: 10:09:40
Time1 / Time2 is 0.256957

Convert time1 to seconds: 12650
Convert time2 to seconds: 49230
If we convert seconds back to time points we'll get:
New time1: 3:30:50
New time2: 13:40:30
We got same time points!

Now we'll add time2 converted to seconds to time 1
We got: 17:11:20
We got same time point as (time1 + time2)!

Now we'll subtract time1 converted to seconds from the time point above
We got: 13:40:30
We got same time point as time2!

Convert time1 to minutes: 12650
Convert time2 to minutes: 49230
If we convert minutes back to time points we'll get:
New time1: 3:31:00
New time2: 13:41:00

Finally, we'll try to create '3600_time' literal:
1:00:00

Process finished with exit code 0
```

test2

```
Enter the name of test file (e.g. test1.txt) or type skip to enter
data yourself:test2.txt

We have time1: 596523:14:07
And time2: 7162767:12:41

Sum of time moments: 7759290:26:48
Difference between times: 6566243:58:34
Time1 / Time2 is 0.0832811

Convert time1 to seconds: 2147483647
Convert time2 to seconds: 25785961961
If we convert seconds back to time points we'll get:
New time1: 596523:14:07
New time2: 7162767:12:41
We got same time points!

Now we'll add time2 converted to seconds to time 1
We got: 7759290:26:48
We got same time point as (time1 + time2)!

Now we'll subtract time1 converted to seconds from the time point above
We got: 7162767:12:41
We got same time point as time2!

Convert time1 to minutes: 2147483647
Convert time2 to minutes: 25785961961
If we convert minutes back to time points we'll get:
New time1: 596523:14:00
New time2: 7162767:13:00

Finally, we'll try to create '3600_time' literal:
1:00:00

Process finished with exit code 0
```

test3

```
Enter the name of test file (e.g. test1.txt) or type skip to enter
data yourself:test3.txt

TimePoint can't be negative!

Process finished with exit code 1
```

5. Листинг программы

TimePoint.h

```
/*
 *  Класс TimePoint для работы с моментами времени в формате
"час:минута:секунда". Реализованы операции
 * вычисления разницы между двумя моментами времени, суммы моментов времени,
сложения момента времени и
 * заданного количества секунд, вычисления во сколько раз один момент больше
(меньше) другого, сравнения
```

```
 * моментов времени, перевода в секунды и обратно, перевода в минуты и
обратно.
 *
 * Операции сложения, вычитания и сравнения реализованы в виде перегрузки
операторов. Реализован
 * пользовательский литерал для работы с константами типа TimePoint.
 */

#include <iostream>

#ifndef OOP_LAB2_TIMEPOINT_H
#define OOP_LAB2_TIMEPOINT_H


class TimePoint {
public:
    TimePoint();
    TimePoint(long long hh, long long mm, long long ss);

    long long get_hours() const;
    long long get_min() const;
    long long get_sec() const;
    void print_time() const;

    void set_hours(long long hh);
    void set_min(long long mm);
    void set_sec(long long ss);

    long long to_seconds() const; // converts the time moment to seconds
    long long to_minutes() const; // converts the time moment to minutes
    void set_by_seconds(long long seconds); // converts seconds to time
moment
    void set_by_minutes(long long minutes); // converts minutes to time
moment

    TimePoint& operator=(TimePoint rhs);
    bool operator==(TimePoint& rhs) const;
    bool operator!=(TimePoint& rhs) const;
    bool operator<(TimePoint& rhs) const;
    bool operator<=(TimePoint& rhs) const;
    bool operator>(TimePoint& rhs);
    bool operator>=(TimePoint& rhs);

    TimePoint operator+(TimePoint& rhs) const;
    TimePoint operator+(long long seconds) const;
    TimePoint operator-(TimePoint& rhs); // absolute difference between time
points
    TimePoint operator-(long long seconds);
    double operator/(TimePoint& rhs) const;

private:
    long long hours;
    long long min;
    long long sec;

    void check_data(); // removes overflows and checks if data is positive
};


#endif //OOP_LAB2_TIMEPOINT_H
```

TimePoint.cpp

```
#include "TimePoint.h"

#include <iostream>

TimePoint::TimePoint() : hours(0), min(0), sec(0) {}

TimePoint::TimePoint(long long hh, long long mm, long long ss) : hours(hh),
min(mm), sec(ss) {
    check_data();
}

long long TimePoint::get_hours() const {
    return hours;
}

long long TimePoint::get_min() const {
    return min;
}

long long TimePoint::get_sec() const {
    return sec;
}

void TimePoint::print_time() const {
    std::cout << hours << ":";
    std::cout.fill('0');
    std::cout.width(2);
    std::cout << min << ":";
    std::cout.fill('0');
    std::cout.width(2);
    std::cout << sec << std::endl;
}

void TimePoint::set_hours(long long hh) {
    hours = hh;
    check_data();
}

void TimePoint::set_min(long long mm) {
    min = mm;
    check_data();
}

void TimePoint::set_sec(long long ss) {
    sec = ss;
    check_data();
}

// converts the time moment to seconds
long long TimePoint::to_seconds() const {
    return hours * 3600 + min * 60 + sec;
}

// converts the time moment to minutes
long long TimePoint::to_minutes() const {
    return hours * 60 + min + (sec >= 30);
}
```

```
// converts seconds to time moment
void TimePoint::set_by_seconds(long long int seconds) {
    hours = seconds / 3600;
    seconds %= 3600;
    min = seconds / 60;
    seconds %= 60;
    sec = seconds;
    check_data();
}

// converts minutes to time moment
void TimePoint::set_by_minutes(long long int minutes) {
    hours = minutes / 60;
    min = minutes % 60;
    check_data();
}

// removes overflows and checks if data is positive
void TimePoint::check_data() {
    if (sec < 0 || min < 0 || hours < 0) {
        std::cerr << "TimePoint can't be negative!" << std::endl;
        exit(1);
    }
    if (sec > 59) {
        min += sec / 60;
        sec %= 60;
    }
    if (min > 59) {
        hours += min / 60;
        min %= 60;
    }
}

TimePoint &TimePoint::operator=(TimePoint rhs) {
    hours = rhs.hours;
    min = rhs.min;
    sec = rhs.sec;
    return *this;
}

bool TimePoint::operator==(TimePoint &rhs) const {
    return hours == rhs.hours && min == rhs.min && sec == rhs.sec;
}

bool TimePoint::operator!=(TimePoint &rhs) const {
    return !(*this == rhs);
}

bool TimePoint::operator<(TimePoint &rhs) const {
    if (hours != rhs.hours) {
        return hours < rhs.hours;
    }
    if (min != rhs.min) {
        return min < rhs.min;
    }
    return sec < rhs.sec;
}

bool TimePoint::operator<=(TimePoint &rhs) const {
    return *this < rhs || *this == rhs;
}
```

```
bool TimePoint::operator>(TimePoint &rhs) {
    return rhs < *this;
}

bool TimePoint::operator>=(TimePoint &rhs) {
    return rhs <= *this;
}

TimePoint TimePoint::operator+(TimePoint &rhs) const {
    return TimePoint(hours + rhs.hours, min + rhs.min, sec + rhs.sec);
}

TimePoint TimePoint::operator+(long long int seconds) const {
    return TimePoint(hours, min, sec + seconds);
}

// absolute difference between time points
TimePoint TimePoint::operator-(TimePoint &rhs) {
    if (rhs > *this) {
        return rhs - *this;
    }
    // *this <= rhs
    long long h = hours - rhs.hours;
    long long m = min - rhs.min;
    long long s = sec - rhs.sec;
    if (s < 0) { // if seconds are negative we should convert some minutes
to seconds
        m += s / 60 - 1;
        s = s % 60 + 60;
    }
    if (m < 0) { // same for minutes and hours
        h += m / 60 - 1;
        m = m % 60 + 60;
    }
    TimePoint result(h, m, s);
    result.check_data();
    return result;
}

TimePoint TimePoint::operator-(long long int seconds) {
    if (to_seconds() < seconds) { // we can't make subtraction then
        std::cerr << "Subtrahend is greater than minuend. Time moment can't
be negative!" << std::endl;
        exit(2);
    }
    TimePoint tmp(0, 0, seconds);
    return *this - tmp;
}

double TimePoint::operator/(TimePoint &rhs) const {
    return (double)to_seconds() / (double)rhs.to_seconds();
}
```

main.cpp

```
/* Моисеенков Илья М8О-208Б-19
 *
 * github: mosikk
 *
 *  Создать  класс  TimePoint  для  работы  с  моментами  времени  в  формате
```

```
"час:минута:секунда". Реализовать операции
 * вычисления разницы между двумя моментами времени, суммы моментов времени,
сложения момента времени и
 * заданного количества секунд, вычисления во сколько раз один момент больше
(меньше) другого, сравнения
 * моментов времени, перевода в секунды и обратно, перевода в минуты и
обратно.
 *
 * Операции сложения, вычитания и сравнения реализовать в виде перегрузки
операторов. Реализовать
 * пользовательский литерал для работы с константами типа TimePoint.
 */

#include <iostream>
#include <fstream>

#include "TimePoint.h"

TimePoint operator"" _time(unsigned long long seconds) {
    TimePoint time;
    time.set_by_seconds((long long)seconds);
    return time;
}

int main() {
    std::string files_name;
    std::cout << "Enter the name of test file (e.g. test1.txt) or type skip
to enter data yourself: ";
    std::cin >> files_name;
    std::cout << std::endl;
    int hh1, mm1, ss1, hh2, mm2, ss2;
    if (files_name != "skip") {
        std::ifstream input;
        input.open(files_name);
        input >> hh1 >> mm1 >> ss1 >> hh2 >> mm2 >> ss2;
    }
    else {
        std::cout << "Enter hours, minutes, seconds for time point 1: ";
        std::cin >> hh1 >> mm1 >> ss1;
        std::cout << std::endl;
        std::cout << "Enter hours, minutes, seconds for time point 2: ";
        std::cin >> hh2 >> mm2 >> ss2;
        std::cout << std::endl;
    }
    TimePoint time1(hh1, mm1, ss1);
    TimePoint time2;
    time2.set_hours(hh2);
    time2.set_min(mm2);
    time2.set_sec(ss2);
    std::cout << "We have time1: ";
    time1.print_time();
    std::cout << "And time2: ";
    time2.print_time();
    std::cout << std::endl;

    TimePoint sum = time1 + time2;
    std::cout << "Sum of time moments: ";
    sum.print_time();
    TimePoint diff = time1 - time2;
    std::cout << "Difference between times: ";
    diff.print_time();
```

```
    double quotient = time1 / time2;
    std::cout << "Time1 / Time2 is " << quotient << std::endl;
    std::cout << std::endl;

    long long seconds1 = time1.to_seconds();
    long long seconds2 = time2.to_seconds();
    std::cout << "Convert time1 to seconds: " << seconds1 << std::endl;
    std::cout << "Convert time2 to seconds: " << seconds2 << std::endl;

    TimePoint time1_seconds;
    TimePoint time2_seconds;
    time1_seconds.set_by_seconds(seconds1);
    time2_seconds.set_by_seconds(seconds2);
    std::cout << "If we convert seconds back to time points we'll get: " <<
std::endl;
    std::cout << "New time1: ";
    time1_seconds.print_time();
    std::cout << "New time2: ";
    time2_seconds.print_time();
    if (time1 == time1_seconds && time2 == time2_seconds) {
        std::cout << "We got same time points!" << std::endl;
    }
    else {
        std::cout << "The didn't get same time points. Error occurred!" <<
std::endl;
    }
    std::cout << std::endl;

    TimePoint time3 = time1 + seconds2;
    std::cout << "Now we'll add time2 converted to seconds to time 1" <<
std::endl;
    std::cout << "We got: ";
    time3.print_time();
    if (time3 == sum) {
        std::cout << "We got same time point as (time1 + time2)!" <<
std::endl;
    }
    else {
        std::cout << "The didn't get the same time point. Error occurred!"
<< std::endl;
    }
    std::cout << std::endl;

    TimePoint time4 = time3 - seconds1;
    std::cout << "Now we'll subtract time1 converted to seconds from the time
point above" << std::endl;
    std::cout << "We got: ";
    time4.print_time();
    if (time4 == time2) {
        std::cout << "We got same time point as time2!" << std::endl;
    }
    else {
        std::cout << "The didn't get the same time point. Error occurred!"
<< std::endl;
    }
    std::cout << std::endl;

    long long minutes1 = time1.to_minutes();
    long long minutes2 = time2.to_minutes();
    std::cout << "Convert time1 to minutes: " << seconds1 << std::endl;
    std::cout << "Convert time2 to minutes: " << seconds2 << std::endl;
```

```
    TimePoint time1_minutes;
    TimePoint time2_minutes;
    time1_minutes.set_by_minutes(minutes1);
    time2_minutes.set_by_minutes(minutes2);
    std::cout << "If we convert minutes back to time points we'll get: " <<
std::endl;
    std::cout << "New time1: ";
    time1_minutes.print_time();
    std::cout << "New time2: ";
    time2_minutes.print_time();
    std::cout << std::endl;

    std::cout << "Finally, we'll try to create '3600_time' literal:" <<
std::endl;
    TimePoint time5 = 3600_time;
    time5.print_time();

}
```

6. Выводы

Данная лабораторная работа способствовала изучению механизма перегрузки операторов внутри классов на языке C++ и созданию пользовательских литералов. Даже на примере простого класса можно убедиться, что перегрузка операторов - это мощный аппарат для организации взаимодействия различных объектов (возможно даже различных классов). При перегрузке операторов мы освобождаем себя от необходимости помнить, как называются методы для выполнения простейших действий над объектами, ведь для этого будут использоваться привычные всем операторы (перегрузить можно практически все стандартные операторы).

7. Список используемых источников

1. Руководство по языку C++ [Электронный ресурс]. URL: https://www.cplusplus.com/ (дата обращения 01.10.2020).
2. Построение классов C++ [Электронный ресурс]. URL: http://cppstudio.com/post/439/ (дата обращения 02.10.2020).
3. Перегрузка операторов в C++ [Электронный ресурс]. URL: https://metanit.com/cpp/tutorial/5.14.php (дата обращения 02.10.2020).