Московский авиационный институт
(Национальный исследовательский университет)

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

Лабораторная работа № 2

Тема: Операторы и литералы

Студент: Суханов Егор Алексеевич
Группа: 80-206
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

1. Постановка задачи

Условие задачи: 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.
Операции сложения и вычитания TimePoint, а так же сравнения (больше, меньше и равно) необходимо реализовать в виде перегрузки операторов.
Необходимо реализовать пользовательский литерал для работы с константами типа **TimePoint**.

Выделим требования:

- Ввод вывод:
  - Стандартный поток ввода и вывода;
  - Простая обработка команд
- Класс TimePoint:
  - операция «-» -- разница между двумя TimePoint;
  - операция «+» -- сумма между двумя TimePoint;
  - сложение TimePoint и секунд;
  - вычитание секунд из TimePoint;
  - вычисление во сколько раз один момент времени больше (меньше) другого;
  - сравнение («>» «>=» «==» «<=» «<» «!=»)моментов времени;
  - перевод в секунды и обратно;
  - перевод в минуты и обратно;
  - пользовательский литерал;
  - Метка времени находится в интервале от 0:0:0 до maxInt:59:59.

2. Решение Задачи

Класс TimePoint имеет 3 приватных поля:

- hours - кол-во часов
- minuts - кол-во минут
- seconds - кол-во секунд

Для всех полей есть геттеры и сеттеры.

Для работы классом реализованы следующие методы:

- ToSeconds - Конвертирует TimePoint в секунды
  - используется следующая формула:
    (hours * 60 + minuts) * 60 + seconds
- FromSeconds - Из секунд устанавливает приватные поля

    - используется следующий алгоритм:
        this->seconds = seconds % 60;
        minuts = seconds / 60;
        hours = minuts / 60;
        minuts %= 60;
- ToMinuts и FromMinuts работают аналогично ToSeconds и FromSeconds
- AddSeconds - Добавляет кол-во секунд к экземпляру
    - используется следующий алгоритм:
        Перевести поля h m s в секунды
        Добавить к секундам аргумент функции
        Перевести полученное значение в поля h m s
- TimesMore - Возвращает отношение этой временной метки к другой
    - используется следующий алгоритм:
        (double)ToSeconds() / (double)tp.ToSeconds()

Согласно заданию, был реализован пользовательский литерал:
    TimePoint operator"" _tp(const char *str, size_t size);

Для ввода вывода, а также для операций сравнения, вычитания и сложения были перегружены соответствующие операторы.

Для интерактивной проверки класса TimePoint был организован ввод и выполнение команд.

Для сборки проекта нужно использовать CMake.

3. Набор тестов

Набор модульных тестов можно найти в файле tests.cpp.
Общее тесирование:
test_01.txt - тестирование работы команд:
    toSec 10 30 15
    toMin 10 30 15
    fromSec 37815
    fromMin 630
    add 10 30 15 10 30 15
    sub 10 30 15 1 0 0
    timesMore 6 0 0 1 30 0
    addSec 10 30 15 3600
    cmp 10 30 15   11 15 15

test_02.txt - тестирование неправильного ввода:

wrong_cmd
toSec 10 60 15
toSec 10 30 60
toSec -5 0 0
toSec awoeeggriuh
toSec 10 30 5fewlfk
fromSec -1246
fromSec waflk33
fromSec 123feeg

4. Результаты выполнения тестов

test_01.txt :

> 37815 сек.
> 630 мин.
> 10:30:15
> 10:30:0
> 21:0:30
> 9:30:15
> 4
> 11:30:15
> 10:30:15 != 11:15:15
10:30:15 < 11:15:15
10:30:15 <= 11:15:15
>

test_02.txt :

> Такой команды не существует.
> Ошибка ввода.
> Ошибка ввода.
> Ошибка ввода.
> Ошибка ввода.
> Ошибка ввода.
> Кол-во секунд не может быть меньше нуля.
> Ошибка ввода.
> Ошибка ввода.
>

5. Листинг программы

main.cpp :

```
/*
  Лабораторная работа: 2
```

```
  Вариант: 14
  Группа: М8О-206Б-19
  Автор: Суханов Егор Алексеевич

  Задание:
    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.
    Операции сложения и вычитания TimePoint,
    а так же сравнения (больше, меньше и равно) необходимо реализовать в
виде перегрузки операторов.
    Необходимо реализовать пользовательский литерал для работы с
константами типа TimePoint.
*/
#include <iostream>
#include <string>
#include "time_point.hpp"

using namespace std;

// Ощищает статус cin, а так же игнорирует остаток строки
void clearLine()
{
    cin.clear();
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}

// Считывает TimePoint с обработкой ошибок
bool readTimePoint(TimePoint &tp)
{
    if (cin >> tp && isspace(cin.peek()))
        return true;

    cout << "Ошибка ввода." << endl;
    clearLine();
    return false;
}

void help()
{
    cout <<
        "Ввод метки времени производится в формате: h m s\n"
        "Доступны следующие команды:\n"
        "\texit                                выход из программы\n"
        "\thelp                                вывод справочной информации о
программе\n"
        "\ttoSec     <TimePoint>              перевод в секунды\n"
        "\ttoMin     <TimePoint>              перевод в минуты\n"
        "\tfromSec   <seconds>                перевод из секунд в TimePoint\
n"
        "\tfromMin   <minuts>                 перевод из минут в TimePoint\
n"
        "\tadd       <TimePoint> <TimePoint>  сложение двух TimePoint\n"
        "\tsub       <TimePoint> <TimePoint>  вычитание двух TimePoint\n"
```

```
        "\ttimesMore <TimePoint> <TimePoint> нахождение отношения между
TimePoint\n"
        "\taddSec    <TimePoint> <seconds>   добавить секунды к TimePoint\
n"
        "\tcmp       <TimePoint> <TimePoint> сравнение двух TimePoint\n";
}

void toSec()
{
    TimePoint tp;
    if (readTimePoint(tp))
    {
        cout << tp.ToSeconds() << " сек." << endl;
    }
}

void toMin()
{
    TimePoint tp;
    if (readTimePoint(tp))
    {
        cout << tp.ToMinuts() << " мин." << endl;
    }
}

void fromSec()
{
    int sec;
    if (cin >> sec && isspace(cin.peek()))
    {
        if (sec < 0)
            cout << "Кол-во секунд не может быть меньше нуля." << endl;
        else
        {
            TimePoint tp;
            tp.FromSeconds(sec);
            cout << tp << endl;
        }
    }
    else
    {
        clearLine();
        cout << "Ошибка ввода." << endl;
    }
}

void fromMin()
{
    int minutes;
    if (cin >> minutes && isspace(cin.peek()))
    {
        if (minutes < 0)
            cout << "Кол-во минут не может быть меньше нуля." << endl;
        else
        {
            TimePoint tp;
            tp.FromMinuts(minutes);
            cout << tp << endl;
        }
    }
    else
```

```
    {
        clearLine();
        cout << "Ошибка ввода." << endl;
    }
}

void add()
{
    TimePoint a, b;
    if (readTimePoint(a) && readTimePoint(b))
        cout << a + b << endl;
}

void sub()
{
    TimePoint a, b;
    if (readTimePoint(a) && readTimePoint(b))
        cout << a - b << endl;
}

void timesMore()
{
    TimePoint a, b;
    if (readTimePoint(a) && readTimePoint(b))
        cout << a.TimesMore(b) << endl;
}

void addSec()
{
    TimePoint a;
    int sec;
    if (readTimePoint(a))
    {
        if (cin >> sec)
        {
            a.AddSeconds(sec);
            cout << a << endl;
        }
        else
            clearLine();
    }
}

void cmp()
{
    TimePoint a, b;
    if (readTimePoint(a) && readTimePoint(b))
    {
        if (a == b)
            cout << a << " == " << b << endl;
        if (a != b)
            cout << a << " != " << b << endl;
        if (a < b)
            cout << a << " < " << b << endl;
        if (a <= b)
            cout << a << " <= " << b << endl;
        if (a > b)
            cout << a << " > " << b << endl;
        if (a >= b)
            cout << a << " >= " << b << endl;
    }
```

```
}

int main()
{
    setlocale(LC_ALL, "Russian");

    string cmd;
    cout << "> ";
    while (cin >> cmd)
    {
        if (cmd == "exit")
            break;
        else if (cmd == "help")
            help();
        else if (cmd == "toSec")
            toSec();
        else if (cmd == "toMin")
            toMin();
        else if (cmd == "fromSec")
            fromSec();
        else if (cmd == "fromMin")
            fromMin();
        else if (cmd == "add")
            add();
        else if (cmd == "sub")
            sub();
        else if (cmd == "timesMore")
            timesMore();
        else if (cmd == "addSec")
            addSec();
        else if (cmd == "cmp")
            cmp();
        else
            cout << "Такой команды не существует." << endl;
        cout << "> ";
    }
}
```

time_point.hpp:

```
#include <iostream>

class TimePoint
{
public:
      // Создает метку "0:0:0"
      TimePoint();
      // Создает метку "h:m:s"
      TimePoint(int hours, int minuts, int seconds);

      // Конвертирует метку в секунды
      int ToSeconds() const;
      // Устанавливает метку равной seconds
      void FromSeconds(int seconds);
      // Конвертирует метку в минуты
      int ToMinuts() const;
      // Устанавливает метку равной minuts
      void FromMinuts(int minuts);
      // Добавляет секунды к метке. кол-во секунд может быть положительным
и отрицательным
      void AddSeconds(int seconds);
```

```
	// Возрващает отношение этой временной метки к tp (во сколько раз
больше this, чем tp)
	double TimesMore(const TimePoint tp) const;

	// Возвращает кол-во полных часов
	int GetHours() const;
	// Возвращает кол-во минут в текущем часу
	int GetMinuts() const;
	// Возвращает кол-во секунд в текущей минуте
	int GetSeconds() const;

	// Устанавливает кол-во часов
	void SetHours(int hours);
	// Устанавливает кол-во минут в текущем часу
	void SetMinuts(int minuts);
	// Устанавливает кол-во секнуд в текущей минуте
	void SetSeconds(int seconds);

	friend TimePoint operator-(const TimePoint a, const TimePoint b);
	friend TimePoint operator+(const TimePoint a, const TimePoint b);

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

	friend std::ostream &operator<<(std::ostream &os, const TimePoint
tp);
	friend std::istream &operator>>(std::istream &os, TimePoint &tp);

private:
	int hours;
	int minuts;
	int seconds;
};

TimePoint operator"" _tp(const char *str, size_t size);
```

time_point.cpp:

```
#include "time_point.hpp"
#include <iostream>

TimePoint::TimePoint()
		: hours(0), minuts(0), seconds(0)
{
}

TimePoint::TimePoint(int hours, int minuts, int seconds)
		: hours(hours), minuts(minuts), seconds(seconds)
{
	if (this->hours < 0)
		this->hours = 0;
	if (this->minuts < 0 || this->minuts >= 60)
		this->minuts = 0;
	if (this->seconds < 0 || this->seconds >= 60)
		this->seconds = 0;
}
```

```
int TimePoint::ToSeconds() const
{
	return seconds + (minuts + hours * 60) * 60;
}

void TimePoint::FromSeconds(int seconds)
{
	if (seconds < 0)
		seconds = 0;
	this->seconds = seconds % 60;
	minuts = seconds / 60;
	hours = minuts / 60;
	minuts %= 60;
}

int TimePoint::ToMinuts() const
{
	return minuts + hours * 60;
}

void TimePoint::FromMinuts(int minuts)
{
	if (minuts < 0)
		minuts = 0;

	this->seconds = 0;
	this->minuts = minuts % 60;
	this->hours = minuts / 60;
}

void TimePoint::AddSeconds(int seconds)
{
	int new_sec = ToSeconds() + seconds;
	if (new_sec < 0)
		new_sec = 0;

	FromSeconds(new_sec);
}

double TimePoint::TimesMore(const TimePoint tp) const
{
	return (double)ToSeconds() / (double)tp.ToSeconds();
}

int TimePoint::GetHours() const
{
	return seconds;
}

int TimePoint::GetMinuts() const
{
	return minuts;
}

int TimePoint::GetSeconds() const
{
	return seconds;
}

void TimePoint::SetHours(int hours)
{
```

```
	if (hours < 0)
		return;
	this->hours = hours;
}

void TimePoint::SetMinuts(int minuts)
{
	if (minuts < 0 || minuts >= 60)
		return;
	this->minuts = minuts;
}

void TimePoint::SetSeconds(int seconds)
{
	if (seconds < 0 || seconds >= 60)
		return;
	this->seconds = seconds;
}

bool TimePoint::operator==(TimePoint rhs)
{
	return this->hours == rhs.hours && this->minuts == rhs.minuts &&
this->seconds == rhs.seconds;
}

bool TimePoint::operator!=(TimePoint rhs)
{
	return this->seconds != rhs.seconds || this->minuts != rhs.minuts ||
this->hours != rhs.hours;
}

bool TimePoint::operator>(TimePoint rhs)
{
	return ToSeconds() > rhs.ToSeconds();
}

bool TimePoint::operator>=(TimePoint rhs)
{
	return ToSeconds() >= rhs.ToSeconds();
}

bool TimePoint::operator<(TimePoint rhs)
{
	return ToSeconds() < rhs.ToSeconds();
}

bool TimePoint::operator<=(TimePoint rhs)
{
	return ToSeconds() <= rhs.ToSeconds();
}

TimePoint operator-(const TimePoint a, const TimePoint b)
{
	int s = a.seconds - b.seconds;
	int m = a.minuts - b.minuts;
	int h = a.hours - b.hours;

	if (s < 0)
	{
		m--;
		s += 60;
```

```
	}
	if (m < 0)
	{
		h--;
		m += 60;
	}
	if (h < 0)
	{
		h = 0;
		m = 0;
		s = 0;
	}
	return TimePoint(h, m, s);
}

TimePoint operator+(const TimePoint a, const TimePoint b)
{
	int s = a.seconds + b.seconds;
	int m = a.minuts + b.minuts;
	int h = a.hours + b.hours;

	if (s >= 60)
	{
		m++;
		s -= 60;
	}
	if (m >= 60)
	{
		h++;
		m -= 60;
	}
	return TimePoint(h, m, s);
}

std::ostream &operator<<(std::ostream &os, const TimePoint tp)
{
	os << tp.hours << ':' << tp.minuts << ':' << tp.seconds;
	return os;
}

std::istream &operator>>(std::istream &os, TimePoint &tp)
{
	int s, m, h;
	os >> h >> m >> s;
	if (h < 0 || m < 0 || s < 0 || m >= 60 || s >= 60)
	{
		os.setstate(std::ios_base::failbit);
		h = 0;
		m = 0;
		s = 0;
	}
	tp = TimePoint(h, m, s);
	return os;
}

// возвращает число из строки
int getInt(const char *str)
{
	int res = 0;
	while (*str != ':' && *str != '\0')
	{
```

```
            res *= 10;
            res += *str - '0';
            str++;
      }
      return res;
}

// возвращает индекс следующего разделителя
int getNextSeparate(const char* str)
{
      int sep = 0;
      while (str[sep] != ':')
            sep++;
      return sep;
}

TimePoint operator""_tp(const char *str, size_t size)
{
      int beg = 0; // индекс элемента, находящегося после разделителя

      // Часы
      int h = getInt(str);
      beg = getNextSeparate(str) + 1;
      str += beg;

      // Минуты
      int m = getInt(str);
      beg = getNextSeparate(str) + 1;
      str += beg;

      // Секунды
      int s = getInt(str);

      return TimePoint(h, m, s);
}
```

tests.cpp:

```
// Тестирование отдельных функций
#include <iostream>
#include "time_point.hpp"

// Выводит в поток вывода занчение expr и текст msg
void test(bool expr, const char* msg)
{
      std::cout << (expr?"[TRUE]\t":"[FALSE]\t") << msg << std::endl;
}

void tTimePointEq()
{
      TimePoint a, b;
      test(a == b && !(a != b), "TimePoint eq 1");

      a = TimePoint(10, 30, 15);
      b = TimePoint(10, 30, 15);
      test(a == b && !(a != b), "TimePoint eq 2");

      a = TimePoint(10, 30, 0);
      test(!(a == b) && a != b, "TimePoint eq 3");

      a = TimePoint(10, 0, 15);
```

```
	test(!(a == b) && a != b, "TimePoint eq 4");

	a = TimePoint(10, 0, 15);
	test(!(a == b) && a != b, "TimePoint eq 5");

}

void tTimePointSub()
{
	test(TimePoint(10, 30, 15) - TimePoint(1, 1, 1) == TimePoint(9, 29,
14), "TimePoint Sub 1");
	test(TimePoint(10, 30, 15) - TimePoint(0, 40, 0) == TimePoint(9, 50,
15), "TimePoint Sub 2");
	test(TimePoint(10, 30, 15) - TimePoint(0, 0, 30) == TimePoint(10, 29,
45), "TimePoint Sub 3");
	test(TimePoint(10, 30, 15) - TimePoint(10, 30, 15) == TimePoint(),
"TimePoint Sub 4");
	test(TimePoint(10, 30, 15) - TimePoint(20, 0, 0) == TimePoint(),
"TimePoint Sub 5");
}

void tTimePointAdd()
{
	test(TimePoint(10, 30, 15) + TimePoint(1, 1, 1) == TimePoint(11, 31,
16), "TimePoint Add 1");
	test(TimePoint(10, 30, 15) + TimePoint(0, 40, 0) == TimePoint(11, 10,
15), "TimePoint Add 2");
	test(TimePoint(10, 30, 15) + TimePoint(0, 0, 45) == TimePoint(10, 31,
0), "TimePoint Add 3");
	test(TimePoint(10, 30, 15) + TimePoint(10, 30, 15) == TimePoint(21,
0, 30), "TimePoint Add 4");
	test(TimePoint(10, 30, 15) + TimePoint(20, 0, 0) == TimePoint(30, 30,
15), "TimePoint Add 5");
}

void tTimePointAddSeconds()
{
	TimePoint tp(10, 30, 15);

	tp.AddSeconds(15);
	test(tp == TimePoint(10, 30, 30), "TimePoint Add Seconds 1");

	tp = TimePoint(10, 30, 30);
	tp.AddSeconds(45);
	test(tp == TimePoint(10, 31, 15), "TimePoint Add Seconds 2");

	tp = TimePoint(10, 30, 30);
	tp.AddSeconds(60 * 60);
	test(tp == TimePoint(11, 30, 30), "TimePoint Add Seconds 3");

	tp = TimePoint(10, 30, 30);
	tp.AddSeconds(60 * 60 + 45);
	test(tp == TimePoint(11, 31, 15), "TimePoint Add Seconds 4");
}

void tTimePointTimesMore()
{
	test(TimePoint(3, 0, 0).TimesMore(TimePoint(1, 30, 0)) == 2.,
"TimePoint Times More 1");
	test(TimePoint(3, 20, 20).TimesMore(TimePoint(10, 1, 0)) == 1./3.,
"TimePoint Times More 2");
```

```
}

void tTimePointCmp()
{
	test(TimePoint(3, 15, 2) > TimePoint(2, 0, 0), "TimePoint Cmp 1");
	test(TimePoint(3, 15, 2) >= TimePoint(2, 0, 0)
		&& TimePoint(3, 15, 2) >= TimePoint(3, 15, 2), "TimePoint Cmp
2");

	test(TimePoint(3, 15, 2) < TimePoint(5, 0, 0), "TimePoint Cmp 3");
	test(TimePoint(3, 15, 2) <= TimePoint(5, 0, 0)
		&& TimePoint(3, 15, 2) <= TimePoint(3, 15, 2), "TimePoint Cmp
4");
}

void tTimePointConverts()
{
	TimePoint tp(3, 15, 2);
	TimePoint a;

	a.FromSeconds(tp.ToSeconds());
	test(a == tp, "TimePoint Converts 1");

	a.FromMinuts(tp.ToMinuts());
	test(a == TimePoint(3, 15, 0), "TimePoint Converts 2");
}

void tTimePointLiteral()
{
	test("12:23:12"_tp == TimePoint(12, 23, 12), "TimePoint Literal 1");
}

int main(void)
{
	tTimePointEq();
	std::cout << std::endl;
	tTimePointSub();
	std::cout << std::endl;
	tTimePointAdd();
	std::cout << std::endl;
	tTimePointAddSeconds();
	std::cout << std::endl;
	tTimePointTimesMore();
	std::cout << std::endl;
	tTimePointCmp();
	std::cout << std::endl;
	tTimePointConverts();
	std::cout << std::endl;
	tTimePointLiteral();
}
```

6. Выводы

Выполняя данную ЛР я познакомился с пользовательскими литералами, ключевым словом friend и перегрузкой операторов. С помощью перегруженных операторов можно намного удобнее работать с

переменными. А также это позволяет приблизить пользовательские типы к встроенным.

Нужно не забывать, что операторы имеют свой смысл (например оператор сложения ассоциируется как сущность, которая складывает операнды, а оператор сравнения - сущность, которая сравнивает). Поэтому желательно, что бы перегружаемый оператор подходил по смыслу той операции, которую хочется определить.

Я узнал о существовании функтора. Например, с его помощью можно реализовать Класс, который будет вычислять сумму: int sum = Add(1)(2)(3);

7. Список литературы

1. Страуструп, Бьярне. Язык программирования C++. Краткий курс, 2-е изд. : Пер. с англ. - СПб.: ООО "Диалектика", 2019. - 320 с.: ил. - Парал. тит. англ.