Class & Operator Overloading

Oct 6

Class Time

```
class Time {
        int h;    // 0~23
        int m;   // 0~59
        int s;    // 0~59
public:
        // Constructor
        Time() { h=m=s=0; }
        Time( int hour, int min, int sec ) { h=hour; m=min; s=sec; }

        // accessor (getter)
        int getHour() { return h; }
        int getMin() { return m; }
        int getSec() { return s; }

        // accessor (setter)
        void setHour( int hour ) { if( 0 <= hour && hour <= 23 ) h = hour; }
        void setMin( int min ) { if( 0 <= min && min <= 59 ) m = min; }
        void setSec( int sec ) { if( 0 <= sec && sec <= 59 ) s = sec; }

        friend Time operator+( Time t1, Time t2 );
};
```

```
Time operator+( Time t1, Time t2 ) {
        Time t;
        t.h = t1.h + t2.h;
        t.m = t1.m + t2.m;
        t.s = t1.s + t2.s;
        return t;
}
```

HW4

- 생성자 Time(int hour, int min, int sec)가 불리면 주어진 값을 저장하기 이전에 hour, min, sec 의 범위를 체크하고, 범위에 맞지 않을 때에는 범위가 벗어난 값 대신에 -1을 저장한다.
- 예: Time(3, 100, 20) ⇒ h=3, m=-1, s=20
- 새로운 생성자 Time(int sec)을 추가하여라. 그리고 주어진 sec 값이 음수면 s=-1. 만약 sec 값이 60 이상이면 이를 시, 분, 초로 다시 계산해서 h, m, s값을 설정하도록 한다. 단, h가 24 이상이면 24로 나눈 나머지만 취한다. 예를 들어서 sec=3700 이면 h=1, m=1, s=40 으로 설정한다.
- operator + 를 다음과 같이 수정한다. 두 Time의 s, m, h 값을 각각 더한 후에도 s와 m이 0에서 59 사이에 있도록 값을 자동으로 조절한다. 단, h 값은 이 경우 24 이상이면 24로 나눈 나머지 값을 취한다.
- 예: Time(3, 30, 30) + Time(5, 40, 40) == Time(9, 11, 10)
- 예: Time(15, 30, 30) + Time(15, 20, 40) == Time(6, 51, 10)

HW4 Evaluation

```
void checkTime( string testname, Time t, int h,
int m, int s )
{
   if( t.getHour() == h && t.getMin() == m &&
t.getSec() == s )
      cout << testname << " passed" << endl;
   else
      cout << testname << "  failed" << endl;

}
```

```
nt main()
{
   Time t1(1, 2, 74);
   checkTime( "test 1", t1, 1, 2, -1 );

   Time t2(1, -3, 23);
   checkTime( "test 2", t2, 1, -1, 23 );

   Time t3(3700);
   checkTime( "test 3", t3, 1, 1, 40 );

   Time t4(3, 30, 30);
   Time t5(5, 40, 40);
   Time t6 = t4 + t5;

   checkTime( "test 4", t6, 9, 11, 10 );

   Time t7(15, 30, 30);
   Time t8(15, 20, 40);
   Time t9 = t7 + t8;

   checkTime( "test 5", t9, 6, 51, 10 );
}
```

HW4 Sol

```
Time::Time(int hour, int min, int sec)
{
    s = ( 0 <= sec && sec <= 59 ) ? sec : -1;
    m = ( 0 <= min && min <= 59 ) ? min : -1;
    h = ( 0 <= hour && hour <= 23 ) ? hour : -1;
}
```

HW4 Sol

```
Time::Time(int sec)
{
    h = (sec / 3600) % 24;
    m = (sec % 3600) / 60;
    s = sec % 60;
}
```

HW4 Sol

```
Time operator+( Time t1, Time t2 ) {
    return Time( ( t1.h + t2.h )*3600 + (t1.m + t2.m) * 60 + (t1.s + t2.s) );
}
```

More Time

- showTime()
- ==
- <
- <=
- >
- >=

- +=
- -=
- ++
- --
- <<
- >>

Reference vs. Pointer

- Pointer
  - Hold the memory address of a data
  - Need dereferencing (*)
  - Used for call by reference
- Reference
  - Hold a nick name of a data
  - No need dereferencing
  - Used for call by reference

Reference vs. Pointer

- int x;
- int *px = &x;
- (*px)++;

- int &rx = x;
- int rx++;

Call by Reference

```
void swap( int x, int y ) {
    int temp=x; x=y; y=temp;
}
void swap( int *px, int *py ) {
    int temp = *px; *px = *py; *py = temp;
}
void swap( int &x, int &y ) {
    int temp = x; x = y; y = temp;
}
```

Overloading

```
bool operator==( const Time d1, const Time d2 );
bool operator>( const Time d1, const Time d2 );
bool operator>=( const Time d1, const Time d2 );
Date &operator+=(int n);
Date &operator=(const Date &d);
Date &operator++(Time &d);
Date operator++(Time &d, int)
```

Overloading <<

```
ostream &operator<<(ostream &output, const Time &t) {
    output << …
    return output;
}
```

Overloading >>

```
istream &operator<<(istream &input, Time &t) {
    input >> …
    return input;
}
```