C++ Class

Tuesday, August 25, 2009
3:30 PM

interface: in the .h file

```
#ifndef FILE_H
#define "
    ;
```

name of object

```
class Timer {
    public:
    // constructor
    Timer(double s=0.0, double e=0.0, double t=0.0):\
        ts(s), \
        te(e), \
        tt(t)    { }

    private:
    double ts;  // time start
    double te;  // time end
    double tt;  // total time (te - ts)
};
#endif
```

Time(Time rhs) ① copy constructor

```
Time(const Time& rhs):
    ts(rhs.ts), \
    te(rhs.te), \
    tt(rhs.tt)    { }
```

from

~~Time(Time *rhs): this(*rhs)~~
~~ts(rhs->ts), \~~

② destructor

```
~Timer()    { }  // destructor
```

Normally, would use this to delete any dynamically allocated data members (i.e., free memory don't leak memory)

(3)

```
// assignment operator

const Timer& operator=(const Timer& rhs)
{   // ptr to this obj
    if (this != &rhs) { // standard alias test (a=a)
        ts = rhs.ts;   ≡ this->ts ≡ (*this).ts
        te = rhs.te;
        tt = rhs.tt;
    }
    return *this;
}
```

---

e.g. usage

```
int main( )
{
    Timer t1;          // Timer(s=0.0, e=0.0, t=0.0) gets called
    Timer t2 = t1;     // { assignment or? or copy constructor?
    Timer t3(t1);

    t3 = t2;  ≡  t3.operator=(t2)
}                      t1.start(); t1.end();
```

// member functions (in .h)

```
void start();
void end();

double elapsed_us()   { return tt; }
double elapsed_ms()   { return tt/1000.0; }
double elapsed_s()    { return tt/1000000.0; }
double stamp_us();
```

to be written in the .cpp file.

still in the header file

// friends

```
friend ostream& operator<<(ostream& s, const Timer& rhs);
friend ostream& operator<<(ostream& s, Timer *rhs)
                        { return (s << (*rhs)); }
```

Usage: `std::cout << t3;`

= `std::cout.operator<<(t3)`

```
Timer *pt;

pt = new Timer;

cout << pt;
```