

# C vs. C++

Typically, C design pattern to create a re-usable module is performed like so:

```
typedef struct {
    int target_ms;
    int interval_ms;
} timer_t;
void timer_start(timer_t *t, uint32_t interval);
void timer_stop(timer_t *t);
bool timer_expired(timer_t *t);
```

On the other hand, C++ for the same module would be:

```
class timer
{
public:
    void start(uint32_t interval);
    void stop();
    bool expired();

private:
    int target_ms;
    int interval_ms;};
```

Code analysis

```
void main(void)
{
    /**
     * C convention:
```

```
*   object_method(&obj)
* Then you pass the object as the first parameter to the methods to "operate on" this object
*/
timer_t t;
timer_start(&t, 1000);
timer_stop (&t);

/**
* C++ convention:
*   obj.method()
*
* C++ automatically passes the object pointer, known as the "this" pointer to the method
* In reality, the language and the compiler is invoking the methods just like C:
*   timer::start(&obj, 1000);
*/
timer t;
t.start(1000);
t.stop();}
```

---

Revision #1

Created 7 years ago by [Preet Kang](#)

Updated 2 years ago by [Preet Kang](#)