The Pointer this

Behind the scenes, the implicit parameter is a pointer to the calling object. Every member function has an implicit parameter. Thus the effective signature for the `hours()` function is as if you had declared it like this:

```
int hours(const Time* const this);
```

The keyword `this` is the **name which is automatically supplied** for the implicit parameter. The `const` following the `Time*` means that the value inside the pointer can never be changed; it always points to the block of data containing the object's data members. The `const` following the member function header means that the implicit parameter is a pointer to a `const Time` object.

If you wish, you can **explicitly** use the pointer when calling other member functions, or accessing data members:

```
int Time::hours() const
{
    return this->m_hours;
};
```

Initializing this

When you **call a member function** like this:

```
Time t;     // a Time object
cout << t.hours() << endl;           // value of t::m_hours
```

That call is **implicitly translated** into code that acts as if you had written:

```
Time t;     // a Time object
cout << hours(&t) << endl;           // value of t::m_hours
```

Because of this call, the `this` pointer is initialized to the **address of the calling object**.