Definitions Outside Class Definition Also Require Prefix

Both member and class **function definitions** are usually* **specified outside of the class definition**.

Again, the **compiler must be given the context**, so…

```
int32_t MyClass::memFunc
    (char x, double* y) {
}
```

*Short/simple functions can be written into the class definition, in which case the code is often inlined into calling code.

Within Functions, Class:: and `this->` are Often Inferred

Within a member or class **function definition**, **symbols** from the class can be **used without the namespace prefix**.

Fields and member functions used in a member function **implicitly add the prefix "`this->`"**

```
field++; // means this->field++;
memFunc ();
    // means this->memFunc ();
```


Class Function Definitions Do Not Include "`static`"

For **class function definitions, do not write "`static`"**. If a class definition includes

```
static void doSomething (int32_t q);
```

outside of the class definition, the **class function is defined as follows**:

```
void MyClass::doSomething
    (int32_t q) {
}
```


Class Functions Have No `this` Argument

Class functions have no `this` argument, so …

```
void MyClass::doSomething
    (int32_t q)
{
    field++;      // invalid
    method ();    // not allowed
    classFunc (q - 1);
      // ok -- MyClass:: is inferred
}
```

