Review Modules and Explore How They Map into C++

Let's **review the contents of a module**.
- You've seen at least one: the memory allocator)
- Then we will **discuss how each piece maps into C++**.

Typically,
- a module is organized
- around one or more data structures.

Aspects of Data Structures and Modules

A **data structure** is
- a **`struct`** with fields (sometimes > 1)
- related **static data**
- **interface functions** that operate on one or more instances
- **internal / implementation functions**
- **initialization / teardown routines** for an instance

A **module** also **includes initialization / teardown routines for the module**.

Class Definition (Small Module) Contains All Information

In **C++**, the **smallest module is a class**.

Classes are
- usually **defined in header files**,
- as the definition includes
- specification of all interface functions.

In fact, declarations for **everything related to a class must appear in the class definition**.

Elements of a Class Definition

```
class MyClass : public ParentClass {
    // fields--same as a struct
    // associated functions (both
    //   interface & implementation)
    // static data related to class
    // initialization / teardown
    //   for an instance
}
```

Let's examine each in more detail.

