File Management is Orthogonal to Information Hiding

In most C++ code, since
- access rights can be managed explicitly
- all **class definitions** are **globally visible**
- (there's no reason to hide anything).

And **file management** is an orthogonal issue:
- **C++** class / module **code**
- can be **organized into files** in any way
- that **suits the needs of the team**.

`private` Restricts Access to within a Class' Code

C++ offers **three levels of access**:
`private`, **`protected`**, and **`public`**.

`private` is the default: access is allowed
- **only within the class' functions**:
- member and class functions,
- code in the class definition, and
- friend classes/functions.

`private` should be **used for** most **fields, class variables, and implementation functions**.

`protected` Allows Access within Derived Class' Code

`protected` extends access to **code within any derived class' functions**.

`protected`
- can also be **used for fields, class variables, and implementation functions**,
- but **`protected`**, inlined get/set methods are usually a better choice for fields.

`public` Allows Arbitrary Access

`public` removes access control, **allowing any code to use the fields/functions**.

`public` should generally
- **only** be used **for interface functions**,
- which can include get/set functions for fields.

