The Interface or Header File

A library may contain several definitions: functions, types, and constants. In C++, the interface and implementation are in two files: a **header (or interface) file** and an **implementation** file. The **interface** file usually ends with the extension `.h`.

Add `#include "digits.h"` in your client file right after the `using namespace std` line. Then, create and open **`digits.h`** and let's look at header guards.

Preprocessor Header Guards

It is possible for one header file to include another. You must do **something** to **make sure that the compiler doesn't include the same interface twice**. You do that by adding three lines to every header file that are known as the **interface boilerplate**, or **header guards**. They look like this:

```
#ifndef FILE_IDENTIFIER
#define FILE_IDENTIFIER
    // Entire contents of the header file
#endif
```

This pattern appears **in every interface**. These are **instructions to the preprocessor**, a program that examines and modifies your code before it is sent to the compiler. The boilerplate consists of the **`#ifndef`** and **`#define`** at the beginning and the **`#endif`** at the end.

1. The **`#ifndef`** preprocessor directive checks whether the **`FILE_IDENTIFIER`** symbol has been defined in the **current translation unit**. When the preprocessor reads this interface file **for the first time**, the answer is no.
2. The next line **defines the symbol**, using **`#define`**. Thus, if asked later to **`#include`** the same interface, the **`FILE_IDENTIFIER`** **will already be defined**, and the preprocessor will **skip over the contents** of the interface this time around, not including them a second time.

A common convention to create **`FILE_IDENTIFIER`** is to simply **capitalize the name of the file itself**, replacing the dot with an underscore. You may use another convention if you like, but make sure that the name will be unique when you build your project.

Go ahead and add the header guards to **`digits.h`** now.