What are header guards?

Header guards are designed to ensure that the contents of a given *header* file are not copied, more than once, into any single file to prevent duplicate definitions. This is a good thing because we often need to reference the contents of a given header from different project files.

Why We Need It

A function defined more than once returns an error. Take a look at the example below:

CASE : 1

main.cpp

```cpp
#include <iostream>

int foo() // this is a definition for function foo
{
    return 5;
}

int foo() // compile error: duplicate definition
{
    return 5;
}

int main()
{
    std::cout << foo();
    return 0;
}
```

input

Compilation failed due to following error(s).

```
main.cpp: In function 'int foo()':
main.cpp:8:5: error: redefinition of 'int foo()'
 int foo() // compile error: duplicate definition
     ^~~
main.cpp:3:5: note: 'int foo()' previously defined here
 int foo() // this is a definition for function foo
     ^~~
```

CASE 2 :

Similarly, header files, that get included more than once, also give a compilation error. Take a look at the example below:

File 1 : Letters.h

```
include <iostream>
using namespace std;

// We shouldn't be including function definitions in header files
// But for the sake of this example, we will
int GetNumLetters()
{
    return 26;
}
```

FILE 2 : alphabets.h

```
#include "letters.h"
```

File 3: Main.cpp

```
#include "letters.h"

#include "alphabets.h"

#include <iostream>

int main()
{
    return 0;
}
```

Here's what's happening:

First, `main.cpp` includes `letters.h`, which copies the definition for function `getNumLetters` into `main.cpp`.

Then, `main.cpp` `#includes` `alphabets.h`, which includes `letters.h` itself.

This copies contents of `letters.h` (including the definition for function `getNumLetters`) into `alphabets.h`,

which then get copied into `main.cpp`.

NOTE : Individually, each file is fine. However, since `main.cpp` ends up including the content of `letters.h` twice, we've run into problems.

If `alphabets.h` needs `getNumLetters()`, and `main.cpp` needs both `alphabets.h` and `letters.h`, how would you resolve this issue?

This is where ***header guards*** come in.

How To implement Header Guards

Header guards are conditional compilation directives that take the following form:

```
#ifndef FILE
#define FILE

// your declarations (and certain types of definitions) here

#endif
```

When this header is included, the preprocessor checks whether `FILE` has been previously defined.

If this is the first time the header is included, `FILE` will not have been defined. Consequently, it defines `FILE` and includes the contents of the file.

If the header is included again into the same file, `FILE` will already have been defined and thus, the contents of the header will be ignored (thanks to the `#ifndef`).

Now Takes Our Previous Example

File 1 : Letters.h

```
#ifndef LETTERS_H
#define LETTERS_H

#include <iostream>
using namespace std;

// We shouldn't be including function definitions in header files
// But for the sake of this example, we will
int GetNumLetters()
{
    return 26;
}

#endif
```

File 2 : alphabet.h

```
#ifndef ALPHABETS_H
#define ALPHABETS_H

#include "letters.h"

#endif
```

File 3 : main.cpp

```cpp
#include "letters.h"
#include "alphabets.h"
#include <iostream>

int main()
{
    return 0;
}
```

The second inclusion of the contents of `letters.h` (from `alphabets.h`) gets ignored because `LETTERS_H` was already defined from the first inclusion. Therefore, function `getNumLetters`

only gets included once, and the program compiles successfully.