Using #define

One capability of the preprocessor is to substitute one portion of text for another, before your code is compiled. You do this with the preprocessor directive `#define`. Here's an example:

```
#define PI 3.14159265358979323846
```

This is called a `#defined` constant. Note that you do **not** use an equals sign when creating these constants. By convention, constants and functions (called `MACRO`s) created by the preprocessor are named using all-caps.

With `PI` previously defined, if you write this fragment:

```
cout << PI << endl;
```

The preprocessor **replaces** the symbol `PI` with its predefined value before sending it to the compiler, which only sees this:

```
cout << 3.14159265358979323846 << endl;
```

Of course, in this case, a better solution is to just used `const`, which wasn't available in C. In C++, we discourage the use excessive use of the preprocessor because it can lead to unreadable, hard-to-maintain code, as well as hiding bugs beneath several layers of obfuscation. Of course, some of you may revel in that, so you'll want to look at the winners of the annual Obfuscated C Code Contest.

The one place where we still use `#define` in C++ is for the use of **source control** using conditional compilation. We'll look at that next.