`#define`

Here's an example. If you write

```
#define PRICE_OF_CORN 0.99
```

when you want to, for example, print the price of corn, you use the word `PRICE_OF_CORN` instead of the number 0.99 – the preprocessor will replace all instances of `PRICE_OF_CORN` with 0.99, which the compiler will interpret as the literal `double` 0.99. The preprocessor performs substitution, that is, `PRICE_OF_CORN` is replaced by 0.99 so this means there is no need for a semicolon.

It is important to note that `#define` has basically the same functionality as the "find-and-replace" function in a lot of text editors/word processors.

For some purposes, `#define` can be harmfully used, and it is usually preferable to use `const` if `#define` is unnecessary. It is possible, for instance, to `#define`, say, a `DOG` as the number 3, but if you try to print it, thinking that `DOG` represents a string that you can show on the screen, the program will have an error. `#define` also has no regard for type. It disregards the structure of your program, replacing the text *everywhere* (in effect, disregarding scope), which could be advantageous in some circumstances, but can be the source of problematic bugs.

It is good convention to write `#define`d words in all capitals, so a programmer will know that this is not a variable that you have declared but a `#define`. It is not necessary to end a preprocessor directive such as `#define` with a semicolon; in fact, some compilers may warn you about unnecessary tokens in your code if you do.