14.3 Macro Definitions

The macros that we've been using since Chapter 2 are known as *simple* macros, because they have no parameters. The preprocessor also supports *parameterized* macros. We'll look first at simple macros, then at parameterized macros. After covering them separately, we'll examine properties shared by both.

Simple Macros

The definition of a ***simple macro*** (or ***object-like macro,*** as it's called in the C standard) has the form

#define directive (simple macro)

```
#define identifier replacement-list
```

replacement-list is any sequence of ***preprocessing tokens,*** which are similar to the tokens discussed in Section 2.8. Whenever we use the term "token" in this chapter, it means "preprocessing token."

A macro's replacement list may include identifiers, keywords, numeric constants, character constants, string literals, operators, and punctuation. When it encounters a macro definition, the preprocessor makes a note that *identifier* represents *replacement-list*; wherever *identifier* appears later in the file, the preprocessor substitutes *replacement-list.*

⚠ Don't put any extra symbols in a macro definition—they'll become part of the replacement list. Putting the = symbol in a macro definition is a common error:

```
#define N = 100     /*** WRONG ***/
...
int a[N];           /* becomes int a[= 100]; */
```

In this example, we've (incorrectly) defined `N` to be a pair of tokens (= and `100`). Ending a macro definition with a semicolon is another popular mistake:

```
#define N 100;      /*** WRONG ***/
...
int a[N];           /* becomes int a[100;]; */
```

Here `N` is defined to be the tokens `100` and `;`.

The compiler will detect most errors caused by extra symbols in a macro definition. Unfortunately, the compiler will flag each use of the macro as incorrect, rather than identifying the actual culprit—the macro's definition—which will have been removed by the preprocessor.

Simple macros are primarily used for defining what Kernighan and Ritchie call "manifest constants." Using macros, we can give names to numeric, character, and string values:

Q&A