

```
#define SUIT      int
#define CLUBS    0
#define DIAMONDS 1
#define HEARTS   2
#define SPADES   3
```

Our previous example now becomes easier to read:

```
SUIT s;
...
s = HEARTS;
```

This technique is an improvement, but it's still not the best solution. There's no indication to someone reading the program that the macros represent values of the same "type." If the number of possible values is more than a few, defining a separate macro for each will be tedious. Moreover, the names we've defined—CLUBS, DIAMONDS, HEARTS, and SPADES—will be removed by the preprocessor, so they won't be available during debugging.

C provides a special kind of type designed specifically for variables that have a small number of possible values. An *enumerated type* is a type whose values are listed ("enumerated") by the programmer, who must create a name (an *enumeration constant*) for each of the values. The following example enumerates the values (CLUBS, DIAMONDS, HEARTS, and SPADES) that can be assigned to the variables `s1` and `s2`:

```
enum {CLUBS, DIAMONDS, HEARTS, SPADES} s1, s2;
```

Although enumerations have little in common with structures and unions, they're declared in a similar way. Unlike the members of a structure or union, however, the names of enumeration constants must be different from other identifiers declared in the enclosing scope.

Enumeration constants are similar to constants created with the `#define` directive, but they're not equivalent. For one thing, enumeration constants are subject to C's scope rules: if an enumeration is declared inside a function, its constants won't be visible outside the function.

Enumeration Tags and Type Names

We'll often need to create names for enumerations, for the same reasons that we name structures and unions. As with structures and unions, there are two ways to name an enumeration: by declaring a tag or by using `typedef` to create a genuine type name.

Enumeration tags resemble structure and union tags. To define the tag `suit`, for example, we could write

```
enum suit {CLUBS, DIAMONDS, HEARTS, SPADES};
```

`suit` variables would be declared in the following way:

```
enum suit s1, s2;
```