... 
if (prize.type == 1) 
    cin >> prize.id_num;
else 
    cin >> prize.id_char;

Because the union is anonymous, `id_num` and `id_char` are treated as two members of `prize` that share the same address. The need for an intermediate identifier `id_val` is eliminated. It is up to the programmer to keep track of which choice is active.

**Enumerations**

The C++ `enum` facility provides an alternative means to `const` for creating symbolic constants. It also lets you define new types but in a fairly restricted fashion. The syntax for using `enum` resembles structure syntax. For example, consider the following statement:

```c++
enum spectrum {red, orange, yellow, green, blue, violet, indigo, ultraviolet};
```

This statement does two things:

- It makes `spectrum` the name of a new type; `spectrum` is termed an *enumeration*, much as a `struct` variable is called a structure.

- It establishes `red`, `orange`, `yellow`, and so on, as symbolic constants for the integer values 0—7. These constants are called *enumerators*.

By default, enumerators are assigned integer values starting with 0 for the first enumerator, 1 for the second enumerator, and so forth. You can override the default by explicitly assigning integer values. We'll show you how later.

You can use an enumeration name to declare a variable of that type:

```c++
spectrum band;
```

An enumeration variable has some special properties, which we'll examine now.

The only valid values that you can assign to an enumeration variable without a type cast are the enumerator values used in defining the type. Thus, we have the following: