The Need for "This"

Suppose that we wish to add a member function to the Fraction class that allows us to do assignments of the form:

"F = 12;" meaning "F = 12/1;"

Consider the following possible definition:

```
/************************ operator= ****************************
This Fraction function gives us the ability to assign an int to a
Fraction.  A typical call would be:

     F = 1234;  or
     F = N;          where N is an int.
-----------------------------------------------------------------*/
void Fraction::operator=(int N)
{
    Numer = N;
    Denom = 1;
}
```

Note that as it stands, the function does not support statements such as

```
F1 = F2 = 123456;
```

This can be remedied by first changing the prototype to:

```
Fraction operator=(int N);
```

Now we need to have some way of returning the Fraction value that has just been derived from N. C++ provides a simple solution by making available a pointer to the instance being referred to in the function definition. The name of the pointer is "**this**".

Now we simply add a "`return *this;`" statement to the end of the function and statements such as

```
F1 = F2 = 1234;
```

are now legal! The new function is identical to the old one except for the prototype and the addition of the return statement.

There is one problem with this simple solution; it requires that the compiler make extra copies of the value 1234. The reason for this is that in evaluating an assignment such as F2 = 1234; the call "F2 = 1234" first fills F2 with the correct value, then makes a *copy* of the value to return.

Fixing this inefficiency is easy; just change the prototype to

```
Fraction & operator=(int N);
```

Now *operator*= returns a reference to F2 instead of making a copy.