Constructor initializer lists

by Kent Reisdorph

With C++ classes you have the option of initializing the class's data members in the constructor's initializer list. Let's say, for example, that you had a class, SomeClass, with integer data members called x, y, and z. You could initialize those data members in a default constructor like this:

```
SomeClass::SomeClass()
{
    x = 0;
    y = 0;
    z = 0;
}
```

This is something you commonly see in C++ programming. Now, look at the same default constructor using an initializer list:

```
SomeClass::SomeClass() : x(0), y(0), z(0)
{
}
```

If you haven't seen an initializer list used before, this might look odd to you. The initializer list follows the colon in a constructor's definition. The initializer list is actually a more efficient way of initializing a class's data members. When member variables are constructed in an initializer list, they're constructed with the given argument. However, if the initializer list is omitted, the member variables are first constructed with their default constructors, and then explicitly assigned in the code. Thus, while it doesn't matter much for built-in data types, like int, initializer lists are more efficient for any other data type. Here's another example. Let's say that SomeClass had another constructor defined as follows:

```
SomeClass(int _x, int _y, int _z);
```

The definition of this constructor would look like this when an initializer list was implemented:

```
SomeClass::SomeClass(int _x, int _y, int _z) : x(_x), y(_y), z(_y)
{
}
```

If your class is derived from another class, then the initializer list typically follows the call to the base class constructor. Note that if the base class constructor is omitted, then the compiler will insert a call to the base class' default constructor. If SomeClass were derived from a class called Base, then the default constructor would look like this:

```
SomeClass::SomeClass() : Base(), x(0), y(0), z(0)
{
}
```

This syntax can be used to define local and global variables. Note: cannot be used for simple assignment.

```
int x(3);
double y(3.8);
```