PARAMETERIzED CONSTRUCTORS

Parameterized Constructors

It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object. For example, here is a simple class that includes a parameterized constructor:

```
#include <iostream>
using namespace std;
class myclass {
int a, b;
public:
myclass(int i, int j) {a=i; b=j;}
void show() {cout << a << " " << b;}
};
int main()
{
myclass ob(3, 5);
ob.show();
return 0;
}
```

Notice that in the definition of **myclass( ),** the parameters **i** and **j** are used to give initial values to **a** and **b**.

The program illustrates the most common way to specify arguments when you declare an object that uses a parameterized constructor. Specifically, this statement myclass ob(3, 4); causes an object called **ob** to be created and passes the arguments **3** and **4** to the **i** and **j** parameters of **myclass( )**. You may also pass arguments using this type of declaration statement:

```
myclass ob = myclass(3, 4);
```

However, the first method is the one generally used, and this is the approach taken by most of the examples in this book. Actually, there is a small technical difference between the two types of declarations that relates to copy constructors. (Copy constructors are discussed later) Here is another example that uses a parameterized constructor. It creates a class that stores information about library books.

```
#include <iostream>
#include <cstring>
using namespace std;
const int IN = 1;
const int CHECKED_OUT = 0;
class book {
char author[40];
char title[40];
int status;
public:
book(char *n, char *t, int s);
int get_status() {return status;}
void set_status(int s) {status = s;}
void show();
};
book::book(char *n, char *t, int s)
{
strcpy(author, n);
strcpy(title, t);
status = s;
}
void book::show()
{
cout << title << " by " << author;
cout << " is ";
if(status==IN) cout << "in.\n";
```

```
else cout << "out.\n";
}
int main()
{
book b1("Twain", "Tom Sawyer", IN);
book b2("Melville", "Moby Dick", CHECKED_OUT);
b1.show();
b2.show();
return 0;
}
```

Parameterized constructors are very useful because they allow you to avoid having to make an additional function call simply to initialize one or more variables in an object. Each function call you can avoid makes your program more efficient. Also, notice that the short **get_status( )** and **set_status( )** functions are defined in line, within the **book** class. This is a common practice when writing C++ programs.

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iii-object-oriented-programming-with-c-10cs36-notes.pdf