Rule of Three

The **rule of three** is a very important concept when creating classes if you have any of the following criteria:

1 Copy Constructor

2 Destructor

3 Assignment Statement

1) **Pointers**
2) **Dynamically Allocated Data**
3) **Class Member Variables**

The **rule of three** states that if you need to have any three of the following that you must define all of them to ensure that your program will run properly:

1) **Copy Constructor**
2) **Destructor**
3) **Assignment Statement (must be a member function)**

In order to fully comprehend why this is, there are two terms to know, **shallow copy** and **deep copy**.

Shallow copy of a class object means to copy over all member variables and copy over all pointers to a new object. However, the pieces of data that the pointers point to are not copied over. Therefore, the original object and the new object will point to the same data creating disastrous consequences.

Deep copy of a class object means to copy over all member variables and copy all dynamically allocated data such that the new object now has a unique pointer pointing to the copied data.

If you omit the copy constructor, destructor, and/or assignment statement, the compiler will create one for you, but it may not behave as you intend.

Dynamic Array Example

Let's say that you have two objects with dynamically allocated arrays and you want to add the two arrays together and put the result into a new object. This example makes use of the **overloaded + operator** as well. When you come to the statement, such as:

AddedArraysObject = Object1 + Object2;

The **overloaded + operator** function will have to create a temporary object. Once the function terminates, the destructor will destroy the temporary object's dynamic array. Without the **overloaded assignment operator**, **AddedArraysObject** will have its array pointer point to the same data set as the temporary object array pointer, which is no longer valid.

See **Rule.cpp** for a correct implementation and **RuleBroken.cpp** for an incorrect example. Also see the **MultiArray** program after this section.