Order of Construction

A constructor performs its work in this order:

1. It calls base class and member constructors in the order of declaration.
2. If the class is derived from virtual base classes, it initializes the object's virtual base pointers.
3. If the class has or inherits virtual functions, it initializes the object's virtual function pointers. Virtual function pointers point to the class's virtual function table to enable correct binding of virtual function calls to code.
4. It executes any code in its function body.

The following example shows the order in which base class and member constructors are called in the constructor for a derived class. First, the base constructor is called, then the base-class members are initialized in the order in which they appear in the class declaration, and then the derived constructor is called.

```
#include <iostream>
using namespace std;

class Contained1 {
public:
    Contained1() {
        cout << "Contained1 constructor." << endl;
    }
};

class Contained2 {
public:
    Contained2() {
        cout << "Contained2 constructor." << endl;
    }
};

class Contained3 {
public:
    Contained3() {
        cout << "Contained3 constructor." << endl;
    }
};

class BaseContainer {
public:
    BaseContainer() {
        cout << "BaseContainer constructor." << endl;
    }
private:
    Contained1 c1;
    Contained2 c2;
};

class DerivedContainer : public BaseContainer {
public:
    DerivedContainer() : BaseContainer() {
        cout << "DerivedContainer constructor." << endl;
    }
private:
    Contained3 c3;
};
```

```
int main() {
    DerivedContainer dc;
    int x = 3;
}
```

Here's the output:

```
Contained1 constructor.
Contained2 constructor.
BaseContainer constructor.
Contained3 constructor.
DerivedContainer constructor.
```