Subject: Re: Order of member initialization
Posted by jjacksonRIAB on Wed, 21 Sep 2022 15:42:20 GMT
View Forum Message <> Reply to Message

Derp, I just realize I should have attempted to solve your problem instead of trying to just discuss order of member initialization.

OK as far as I know order of member initialization is done in the order it appears in the class, so

```
struct Test {
    int a;
    int b;
    int c;

    Test() : a(), b(a), c(b) {
    }
};
```

would be proper. What would be improper is if you had them appear in a different order:

```
struct Test {
    int c;
    int a;
    int b;

    Test() : a(), b(a), c(b) {
    }
};
```

so it's incumbent upon you once you use chains of dependencies in initializer lists that you don't go moving your member variables around without considering how they'll affect the initializers.

As for how to do the class you initially asked about, which I should have answered but got carried away, try this:

```
#include <Core/Core.h>

using namespace Upp;

class A {
public:
    A(Array<int>&& arr) : array { pick(arr) } {
        Cout() << "Array Initial item count in A = " << array.GetCount() << "\n";
        Cout() << array << EOL;
    }

protected:
    Array<int> array;
};

class B : public A {
public:
    B() : A { Array<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 } } {
        Cout() << "Array Initial item count in B = " << array.GetCount() << "\n";
        Cout() << array << EOL;
    }
};

CONSOLE_APP_MAIN {
    B();
}
```

Pick semantics with move constructor will transfer ownership of the contents of the initialized array into the base array. and they'll be available in the derived class because the base was initialized first.
