Subject: Re: Order of member initialization
Posted by jjacksonRIAB on Tue, 20 Sep 2022 16:27:50 GMT
View Forum Message <> Reply to Message

Quote:
I cannot discuss this, multiple inheritance is too advanced for me.

I hear you. :lol:

It got even crazier than that though. I realized I could also combine it with variadic templates and came up with this monstrosity:

```
#include <iostream>

using namespace std;

struct BaseA {
   int a { 100 };
   int b { 200 };
   int c { 300 };

   auto& BaseARef() { return *this; }
};

struct BaseB {
   BaseA& baseA;

   BaseB(BaseA& baseA) : baseA(baseA) {
      std::cout << baseA.a << "\n";
   }
};

struct BaseC {
   BaseA& baseA;

   BaseC(BaseA& baseA) : baseA(baseA) {
      std::cout << baseA.b << "\n";
   }
};

struct BaseD {
   BaseA& baseA;

   BaseD(BaseA& baseA) : baseA(baseA) {
      std::cout << baseA.c << "\n";
   }
```

```
};

template<typename ...Args>
struct Whatever : BaseA, Args... {
  Whatever() : Args(BaseARef())... {}
};

using Test = Whatever<BaseB, BaseC, BaseD>;

int main(void) {
    Test whatever;
}
```

which prints:

```
100
200
300
```

I mean it's kind of neat because you can use one struct as a data holder for the other ones that all of them have access to but I'm unsure whether I'd use it in production code. The other thing that's cool about it is you can kind of change the initialization order by swapping their positions around in the using statement:

```
using Test = Whatever<BaseD, BaseC, BaseB>;
```

prints

```
300
200
100
```

instead