

Re: operator <<

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.stl/2004-07/0145.html>

From: Carl Daniel [VC++ MVP] (cpdaniel_remove_this_and_nospam_at_mvps.org.nospam)

Date: 07/19/04

Date: Mon, 19 Jul 2004 07:11:24 -0700

"news.microsoft.com" <hovhannes.asatryan@epygilab.am> wrote in message news:%23gDyw9YbEHA.3792@TK2MSFTNGP09.phx.gbl...
> *Hi all.*
>
> *I have written the following code.*
>
> *void main()*
> {
> *int y = 1;*
> *cout << "a = " << y << endl << "b = " << y++ << endl << "c = " << y++ <<*
> *endl;*
>
> }
>
> *and get the following result*
>
> *a = 3*
> *b = 2*
> *c = 1*
>
> *who can explain what's going on?*

Simple: your code has unspecified behavior according to the C++ standard.

In your code, the call to cout translates to:

```
operator <<(  
  operator <<(  
    operator <<(  
      operator <<(  
        operator <<(  
          operator <<(cout,"a="),y),"b="),y++),"c="),y++),endl);
```

after resolution of user-defined operator overloading. Now, the order in which the argument to operator << are evaluated is not specified. Look at the second level nesting: it's not defined whether y++ or the third-level

microsoft.public.vc.stl: Re: operator <<

nested call to operator << is evaluated first. etc.

So technically, the 3 references to y could occur in any order, in could vary from compiler to compiler, id could vary from debug build to release build, it could vary from one compilation to the next. In practice, you'll either get the ordering you saw or the ordering you expected, but you're not guaranteed to get either: 2 1 3 is equally valid output.

-cd