

# Re: Function call evaluation order

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2006-03/msg00994.html>

---

- *From:* "Frederico Pissarra" <[frederico@xxxxxxxxxxx](mailto:frederico@xxxxxxxxxxx)>
  - *Date:* Thu, 23 Mar 2006 16:49:35 -0300
- 

"Cheng" <[chengwuchew@xxxxxxxxxxx](mailto:chengwuchew@xxxxxxxxxxx)> escreveu na mensagem [news:u8X%23alhTGHA.4340@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:u8X%23alhTGHA.4340@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Hi folks,

I need help to explain the behavior with the code below. I would like to chain up function call into a single line, while the order of the function call is correct, I cannot be certain with order the expression within the function call is evaluated.

Would it be better, then, to write all the function call into separate lines?

With regards,  
Cheng Wu

Code:

```
#include <stdio.h>
#include <string>

using namespace std;

void main()
{
    char* data[] = { " zero ", " one ", " two ", " three " };
    string s;
    int i;

    i = 0;
    s.assign(data[i++]).append(data[i++]).append(data[i++]);
    printf("%s\n", s.c_str());

    i = 0;
    s.assign(data[++i]).append(data[++i]).append(data[++i]);
    printf("%s\n", s.c_str());
}
```

## Re: Function call evaluation order

```
i = 0;
s.assign(data[i++]);
s.append(data[i++]);
s.append(data[i++]);
printf("%s\n", s.c_str());
}
```

Output:

```
zero zero zero
three three three
zero one two
```

It seems obvious to me that `assign()` will be performed before the first `append()`, and this one before the second one. Obvious because member selection operator `.` has the higher precedence and it's evaluated left to right.

At the same time, in the first `s.assign()`... the routine uses post-increment operators, so `"i"` will be incremented AFTER the expression is evaluated. The same happens in the second `s.assign()`... but with pre-increment.

The last sentences are evaluated separately, so `"s.assign(data[i++])"` is the same as `"s.assign(data[i]); i++;"`.

Take a look at the precedence table in any good C book (or MSDN)... you'll notice that `.` has the higher precedence... followed by `->`, followed by `[]`, followed by function call `()`...

Hope it helped!

[]s  
Fred

.