

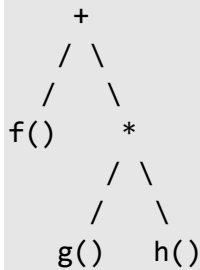
C/C++ Tip 0002 — Order of Operator Evaluation vs. Precedence

© 2009 Rex Jaeschke. All rights reserved.

Issue: Consider the following expression:

```
f() + g() * h()
```

The precedence is quite clear; multiplication wins out over addition, as demonstrated by the corresponding parse tree:



However, in what order are the three terms themselves evaluated? That is, in what order are the three functions called?

Response: The order of evaluation is undefined. Remember, the precedence table only tells us how terms are grouped, *not* the order in which they are evaluated. The only way to make the order predictable is to introduce what the C Standard calls *sequence points*. For example, the steps:

```
x = g();  
x = x * h();  
x = x + f();
```

result in the same precedence as before, but with the functions being called in the guaranteed order `g()`, `h()`, and `f()`.

The following expressions also exhibit undefined behavior because the order of evaluation across the assignment operator is undefined: `a[i] = b[i++]` and `a[i++] = b[i]`. Similarly, `x[i][i++]` and `x[++i][i]` exhibit undefined behavior because the order of evaluation across the subscript operator is undefined.

Do not rely on the order of evaluation of terms across a given operator unless that order is guaranteed.