D

Operator Precedence

D

Operator Precedence

The C operators are executed in a predetermined order. Parentheses are used to change the order in which operators are executed. The innermost parentheses are given the highest precedence and are, therefore, implemented first.

Several operators can share a level of precedence. In such cases, the operators are generally executed from left to right; however, you should never assume this. If the order of execution for operators at the same level of precedence is important, use parentheses to guarantee the execution order.

Table D.1. Level of precedence for operators.

<table>
<tr><th>Highest Precedence</th></tr>
<tr><td>()(function) [](array) -> . :></td></tr>
<tr><td>++(postfix) --(postfix)</td></tr>
<tr><td>++(prefix) --(prefix) ! ~ *(indirection) &(address of)
+(unary) -(unary) sizeof</td></tr>
<tr><td>(typecast)</td></tr>
<tr><td>*(multiplication) / %</td></tr>
<tr><td>+ -</td></tr>
<tr><td><< >></td></tr>
<tr><td>< <= => ></td></tr>
<tr><td>== !=</td></tr>
<tr><td>&(bitwise)</td></tr>
<tr><td>^</td></tr>
<tr><td>|</td></tr>
<tr><td>&&</td></tr>
<tr><td>||</td></tr>
<tr><td>?:</td></tr>
<tr><td>= += -= *= /= %= &= ^= |= <<= >>=</td></tr>
<tr><td>,</td></tr>
<tr><td><b>Lowest Precedence</b></td></tr>
</table>

A few examples:

1.
```
5 * 3 + 4 * 5
= 15 + 20
= 30
```

2.
```
5 * (3 + 4) * 5
= 5 * 12 * 5
= 300
```

3.
```
x *= 5 + 4 * 3      (assume x = 2)
x = x * (5 + 4 * 3)
x = 2 * (5 + 12)
x = 2 * 17
x = 34
```