|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operator Precedence     |  |  | | --- | --- | |  | | | **Operators** | **Associativity** | |  | | | () [] | left to right   HIGH | | ++(postfix) --(postfix) | right to left | |  | | | ++(prefix) --(prefix) | left to right | | ! +(unary) -(unary) \* & | right to left | |  | | | \* / % | left to right   MEDIUM | |  | | | +(binary) -(binary) | left to right | |  | | | < <= > >= | left to right | |  | | | == != | left to right | |  | | | && | left to right | |  | | | || | left to right | |  | | | = += -= \*= /= %= | right to left | |  | | | ?: | right to left   LOW | |  | | | , | left to right | |  | | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Unary +, - and \* have higher precedence than the binary forms. The operator () refers to function call.  **Precedence** determines the order in which operands are bound to operators. Operators on the same line have the same precedence; rows are in order of decreasing precedence.  C does *not* specify the order in which the operands of an operator are evaluated. Similarly, the order in which function arguments are evaluated is not specified. Examples:   |  | | --- | | x = f() + g(); a[i] = i++;  printf("%d %d\n",++n,power(2,n)); z = x / ++x; |   Programs should not depend upon the order of evaluation of expressions, except as guaranteed by ANSI C for the following operators:   |  | | --- | | 1. a, b comma operator (not the comma between arguments)  2. a && b logical and  3. a || b logical or  4. a ? b : c conditional |   All of these guarantee that expression a will be computed before expression b (or c).  In addition, when a function-call takes place all arguments are evaluated before control transfers to the function.   |  | | --- | | 5. a(b) function call |   ANSI C++ guarantees that each full expression will be evaluted before going on.   |  | | --- | | 6. each full expression | | |