Boolean Expressions

Boolean expressions evaluate either true (**non-zero int**) or false (**zero**)

Note that C will treat any non-zero value in such a condition as true

Some C programmers write `if( x )` as a shorthand for `if( x != 0 )`

I recommend the longer form

Operator	Meaning
x > y	x greater than y
x >= y	x greater than or equal to y
x < y	x less than y
x <= y	x less than or equal to y
x == y	x equal to y
x != y	x not equal to y

Boolean Expressions can be Assignments

In C, any expression may cause side-effects, most obviously by containing an assignment - which then delivers it's result as the expression value

The following: `if( x = 3 ) statement;` sets `x` to 3, evaluates `x` (3, so non-zero, so true) and then executes the statement unconditionally

It's nearly always a typo for `if( x == 3 ) statement;`

Most C compilers nowadays issue a warning about `if( x = 3 )`

Compound Boolean Expressions

- The && Operator (and)

Operand1	Operand2	operand1 && operand2
non-zero (true)	non-zero (true)	non-zero (true)
non-zero (true)	zero (false)	zero (false)
zero (false)	non-zero (true)	zero (false)
zero (false)	zero (false)	zero (false)

- The | | Operator (or)

Operand1	Operand2	operand1 | | operand2
non-zero (true)	non-zero (true)	non-zero (true)
non-zero (true)	zero (false)	non-zero (true)
zero (false)	non-zero (true)	non-zero (true)
zero (false)	zero (false)	zero (false)

- The ! Operator (not)

Operand	!Operand
non-zero (true)	zero (false)
zero (false)	non-zero (true)