Bitwise Operators.

C++ provides bitwise operators, which provide bit-level control. The following table describes these operations:

Expression	Produces the result of:
x & *y*	ANDing the bits of *x* with those of *y* Example: 21 & 7 = 10101 & 00111 = 00101 = 5
x | *y*	ORing the bits of *x* with those of *y* Example: 21 | 7 = 10101 | 00111 = 10111 = 23
x ^ *y*	XORing the bits of *x* with those of *y* Example: 21 ^ 7 = 10101 ^ 00111 = 10010 = 18
~*x*	Inverting (complementing) the bits of *x* (0 → 1 and 1 → 0) Example (assuming 16-bit representation): For short unsigned value: ~7 = 1111111111111000 = 65528 For short int value: ~7 = 1111111111111000 = -8
x << *y*	Shifting the bits of *x* to the *left* *y* positions Example: 25 << 3 = 10101 << 3 = 10101000 = 168
x >> *y*	Shifting the bits of *x* to the *right* *y* positions Example: 25 >> 3 = 10101 >> 3 = 00010 = 2

Shortcut versions (with assignment) are also provided: `&=, |=, ^=, !=, <<=, >>=`.
Example: x &= y; is equivalent to x = x & y;

Note: **<<** and **>>** are classic examples of *overloaded* operators.