These are *Bitwise Operators* ([reference](http://en.cppreference.com/w/cpp/language/operator_arithmetic)).

x & 1 produces a value that is either 1 or 0, depending on the least significant bit of x: if the last bit is 1, the result of x & 1 is 1; otherwise, it is 0. This is a bitwise AND operation.

x >>= 1 means "set x to itself shifted by one bit to the right". The expression evaluates to the new value of x after the shift.

**Note:** The value of the most significant bit after the shift is zero for values of unsigned type. For values of signed type the most significant bit is copied from the sign bit of the value prior to shifting as part of *sign extension*, so the loop will never finish if x is a signed type, and the initial value is negative.

|  |
| --- |
|  |
|  |

Code :

int bitCountBin(int n)

{

int res = 0 ;

while(n)

{

if(n&1)

{

res++ ;

}

n>>=1 ;

}

return res ;

}

int hexaBitCount(int n)

{

int sz =0 ;

int d[4] ;

while(n)

{

d[sz++] = n%10 ;

n/=10 ;

}

int NumHex = 0 ;

for(int i=0 ; i<sz ; i++)

{

NumHex = NumHex\*16 + d[i] ;

}

return bitCountBin(NumHex) ;

}