Add 1 to a given number

Write a program to add one to a given number. You are not allowed to use operators like '+', '-', '*', '/', '++', '--' ...etc.

Examples:
Input: 12
Output: 13

Input: 6
Output: 7

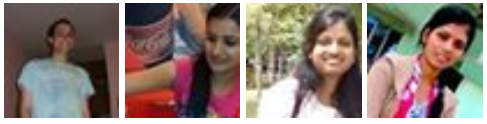
Yes, you guessed it right, we can use bitwise operators to achieve this. Following are different methods to achieve same using bitwise operators.

Method 1
To add 1 to a number x (say 0011000111), we need to flip all the bits after the rightmost 0 bit (we get 0011000000). Finally, flip the rightmost 0 bit also (we get 0011001000) and we are done.

```
#include<stdio.h>

int addOne(int x)
{
  int m = 1;

  /* Flip all the set bits until we find a 0 */
  while( x & m )
  {
    x = x^m;
    m <<= 1;
  }
```

```
  /* flip the rightmost 0 bit */
  x = x^m;
  return x;
}

/* Driver program to test above functions*/
int main()
{
  printf("%d", addOne(13));
  getchar();
  return 0;
}
```

Method 2
We know that the negative number is represented in 2′s complement form on most of the architectures. We have the following lemma hold for 2′s complement representation of signed numbers.

Say, x is numerical value of a number, then

~x = -(x+1) [~ is for bitwise complement]

(x + 1) is due to addition of 1 in 2′s complement conversion

To get (x + 1) apply negation once again. So, the final expression becomes (-(~x)).

```
int addOne(int x)
{
   return (-(~x));
}

/* Driver program to test above functions*/
int main()
{
  printf("%d", addOne(13));
  getchar();
  return 0;
}
```

Example, assume the machine word length is one *nibble* for simplicity.
And x = 2 (0010),
~x = ~2 = 1101 (13 numerical)
-~x = -1101
Interpreting bits 1101 in 2′s complement form yields numerical value as -(2^4 – 13) = -3. Applying

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

‘-’ on the result leaves 3. Same analogy holds for decrement. See this comment for implementation of decrement.
Note that this method works only if the numbers are stored in 2′s complement form.

Thanks to Venki for suggesting this method.

Please write comments if you find the above code/algorithm incorrect, or find better ways to solve the same problem

Related Tpoics:

- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)
- Find position of the only set bit
- Swap all odd and even bits
- Add two bit strings
- Write your own strcmp that ignores cases
- Binary representation of a given number

Writing code in comment? Please use **ideone.com** and share the link here.

34 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

neelabhsingh • 6 months ago
suppose size of int is 2 bytes. if N= 11111111 then method -1 will not work. If i
• Reply • Share ›

anon neelabhsingh • 16 days ago
It is called an overflow..
• Reply • Share ›

anonymous • 11 months ago
i think it will work:

let number be x=a0111...1//a be bit pattern containing 1 & 0
y=~x;//y=(~a)1000...0

z=(y&(~y+1));//z=01000...0
x^=(z-1);//a0000...0

x|=z;//a1000...0
• Reply • Share ›

Recent Comments

Abhi You live US or India?
Google (Mountain View) interview · 28 minutes ago

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...
Find subarray with given sum · 2 hours ago

Dhruv Balhara • 11 months ago

In method 2 :
but we are not allowed to use &#039-&#039 (minus) operator? -> -(~x)

bill nicholson Dhruv Balhara • 7 months ago

yes man u r awesome.... see those people doesnt know

Hanish • a year ago

Proof of decrement expression :

~x + 1 = -x

Put x = ~x //since the above equation holds for every number

=> ~(~x) + 1 = -(~x)
=> (x + 1) = -(~x)

Aashish • 2 years ago

Another approach

```
 int add(int num)
{
        int n=1;
        while(num&n)
        {
                num&=~n;
                n<<=1;
        }
        return num|n;
```

```
}

int main()
{
        int n;
        scanf("%d",&n);
        printf("%d ",add(n));
        return 0;
}
```

http://ideone.com/pArq1

Reply • Share ›

PsychoCoder • 3 years ago

Question says that without using '-' sign. The method 2 uses '-' sign. Otherwis

Reply • Share ›

prakhar PsychoCoder • 2 years ago

ques. is not to use binary '-'.
Here unary '-' is used.

Reply • Share ›

Ernesto • 3 years ago

A recursive method:

```
int plusOne(int x) {
   if ((x & 1) == 0) {
     return x | 1;
   } else {
     return plusOne(x >> 1) << 1;
   }
 }
```

```
}
```

And with recursion removed:

```
int plusOne2(int x) {
   int accum = 1;
   while ((x & 1) != 0) {
     x >>= 1;
     accum <<= 1;
   }
```

see more

• Reply • Share ›

Ernesto Ernesto • 11 months ago

After a second review at this code, and some testing, I've found that it l resulting value would be zero. This happens both in the recursive and i solution would be to add a conditional at the top of the function, like this

```
if (x == -1) return 0;
```

But this could be considered as breaking the rule of not using the minu "-1" is a literal value, and not a use of the minus operator per se).

In any case, does any one knows a workaround on how to handle this

• Reply • Share ›

Ernesto Ernesto • 11 months ago

I found the solution for the edge case without comparing with -1

```
int plus_one_recursive(int x) {
```

```
    if (x & 1 == 0) return x | 1;
    if (x >> 1 == x) return 0;
    return plusOne(x >> 1) << 1;
}

int plus_one_iterative(int x) {
    if (x != 0 && x >> 1 == x) return 0;
    int accum = 1;
    while ((x & 1) != 0) {
        x >>= 1;
        accum <<= 1;
    }
    x |= 1;
    while (accum > 1) {
        x <<= 1;
        accum >>= 1;
    }
    return x;
}
```


vikas • 3 years ago

my solution :
int x = 0;
int m = 1;
while (Convert.ToBoolean(x & m))
{
m = m << 1;
}
x = x | m;
while (m < x) { m = m|(m << 1); }

x = x & m;

Kanagaraj M • 3 years ago

```
int addone(short int x)
{
        int y = 1;
        if( 0 == (x & 1) )
                 x |= 1;
        else
        {
                for( y = 0; y < (sizeof(x) * 8 ) && (x & 1 << y ) ; y-
                {
                        x &= ~( 1 << y );
                }
                x |= ( 1 << y );
        }
        return x;
}

int main()
{
        int a = 16;
        printf(" Add One : %d", addone( a ));
}
```


Aakash Johari • 3 years ago

```
 int add_one(int x)
{
```

```
        int y = 1, temp = x;

        while ( temp & 1 != 0 ) {
                temp >>= 1;
                y <<= 1;
                y |= 1;
        }

        return x^y;
}
```

Reply • Share ›

Virus • 3 years ago

```
int addOne(int x)
{
    x=x<<1;
    x=x|1;
    return x;
}
```

Reply • Share ›

balloon Virus • 3 years ago

This is wrong. addOne(3) = 7

Reply • Share ›

a • 4 years ago

int add1(int x)
{
return -~-x;

}

 • Reply • Share ›

kartik a • 4 years ago

It doesn't seem to be working. See below program prints -10.

```
 #include<stdio.h>
#include<stdlib.h>

int add1(int x)
{
   return -~-x;
}


int main()
{
  printf("%d", add1(11));
  getchar();
  return 0;
}
```

 • Reply • Share ›

Venki kartik • 4 years ago

@Karthik, it should work. I guess there is some typo.

We know that the negative number is represented in 2's compl
architectures. We have the following lemma hold for 2's compl
numbers.

Say, k is numerical value of a number, then

~k = -(k+1) [~ is for bitwise complement]

(k + 1) is due to addition of 1 in 2's complement conversion

To get (k + 1) apply negation once again. So, the final expressio

```
inline
int increment(int x)
{
    return (-(~x));
```

see more

Reply • Share ›

DreamNik • 4 years ago

```
&((char*)n)[1]
```

Only if "*" is not treated as not allowed symbol in type name.

Reply • Share ›

NKS • 4 years ago

```
#include<stdio.h>
int main()
{
   int a,b=1;
   scanf("%d",&a);
   char *p=a;
   printf("%d",&(p[b]));
}
```

Reply • Share ›

Rajendra ➚ NKS • 4 years ago

Line

```
char *p=a;
```

won't compile.

Even if you do

```
char *p = (char*)a;
```

It results in undefined behavior!

^ | v • Reply • Share ›

Sergio • 4 years ago

Nobody says that you can't use '++' :)

^ | v • Reply • Share ›

GeeksforGeeks ➚ Sergio • 4 years ago

OK, we have added it :)

^ | v • Reply • Share ›

chandruthala • 4 years ago

```
public int addOne(int n){
        int y=1;
// Find the position of zero from last
        while( (n&y) !=0 ){
            y<<=1;
        }
// Set the zero as one
        n |=y;
// Set the rest of one after first occurrence of zero
```

```
        n &= y;
        return n;
    }

    int main() {
        printf("%d",addOne(3));
        return 0;
    }
```

Reply • Share ›

sankalp srivastava chandruthala • 4 years ago

```
int addno(int no)
{
        if(no%2)
                return no&~1;
        return no|1;

}
int main()
{
        int number=0;
        printf("%d\n",addno(number));
}
```

the above code works fine for integers as far as adding one is concern

Reply • Share ›

neo sankalp srivastava • 4 years ago

neo ➚ Sankalp Srivastava • 4 years ago

i don't think so.
try it on 4 and 5

▲ | ▼ • Reply • Share ›

Krunal Modi [IIT Kgp/IISc] ➚ chandruthala • 4 years ago

@chandruthala :
Your code will work only in the case of 3,7,15,...(all 1s)
rest will not work.

▲ | ▼ • Reply • Share ›

Shekhu • 4 years ago

how about adding any two given numbers without using '+', '-', '*', '/' …etc?

▲ | ▼ • Reply • Share ›

Venki ➚ Shekhu • 4 years ago

a + b = (a&b) + (a|b), try this code.

▲ | ▼ • Reply • Share ›

Anand ➚ Venki • 3 years ago

not working for all values..

▲ | ▼ • Reply • Share ›

Venki ➚ Shekhu • 4 years ago

Given below is logic based on Digital Circuits

```
// Based on Full Adder Logic Circuit
int AddTwoIntegers(int m, int n) {

    // Sum in 'result' (Exclusive OR as usual in Full Adder)
    int result = m ^ n;
```

```
    // Use n as bus to forward the carry to left
    // Use m as bus to forward previous intermediate addition

    // Iterate till there is no carry
    while ((m & n) != 0) {
        // Carry field
        n = (m & n) << 1;
        // Sum field
        m = result;

        // Save the result in 'result'
        result = m ^ n;
    }

    return result;
}
```


Venki Venki • 4 years ago

Driver run on http://ideone.com/gNi77