Compute the integer absolute value (abs) without branching

We need not to do anything if a number is positive. We want to change only negative numbers. Since negative numbers are stored in 2's complement form, to get the absolute value of a negative number we have to toggle bits of the number and add 1 to the result.

For example -2 in a 8 bit system is stored as follows 1 1 1 1 1 1 1 0 where leftmost bit is the sign bit. To get the absolute value of a negative number, we have to toggle all bits and add 1 to the toggled number i.e, 0 0 0 0 0 0 0 1 + 1 will give the absolute value of 1 1 1 1 1 1 1 0. Also remember, we need to do these operations only if the number is negative (sign bit is set).

Method 1
1) Set the mask as right shift of integer by 31 (assuming integers are stored using 32 bits).

```
mask = n>>31
```

2) For negative numbers, above step sets mask as 1 1 1 1 1 1 1 1 and 0 0 0 0 0 0 0 0 for positive numbers. Add the mask to the given number.

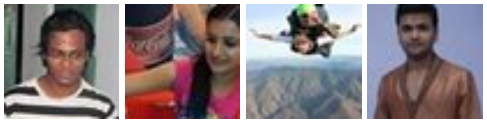
```
mask + n
```

3) XOR of mask +n and mask gives the absolute value.

```
(mask + n)^mask
```

Implementation:

```
#include <stdio.h>
#define CHAR BIT 8
```

```
/* This function will return absoulte value of n*/
unsigned int getAbs(int n)
{
  int const mask = n >> (sizeof(int) * CHAR_BIT - 1);
  return ((n + mask) ^ mask);
}

/* Driver program to test above function */
int main()
{
  int n = -6;
  printf("Absoute value of %d is %u", n, getAbs(n));

  getchar();
  return 0;
}
```

Method 2:

1) Set the mask as right shift of integer by 31 (assuming integers are stored using 32 bits).

```
mask = n>>31
```

2) XOR the mask with number

```
mask ^ n
```

3) Subtract mask from result of step 2 and return the result.

```
(mask^n) - mask
```

Implementation:

```
/* This function will return absoulte value of n*/
unsigned int getAbs(int n)
{
  int const mask = n >> (sizeof(int) * CHAR_BIT - 1);
  return ((n ^ mask) - mask);
}
```

On machines where branching is expensive, the above expression can be faster than the obvious approach, r = (v < 0) ? -(unsigned)v : v, even though the number of operations is the same.

Geometric Algorithms

Popular Posts

Please see this for more details about the above two methods.

Please write comments if you find any of the above explanations/algorithms incorrect, or a better ways to solve the same problem.

References:
http://graphics.stanford.edu/~seander/bithacks.html#IntegerAbs

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.

705 Subscribe

30 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Lohith Ravi • a month ago

return IntegerMax & n ; where n is an integer should also do rit ?

Reply • Share ›

Ashish Mishra • 9 months ago

int x=-2342;
cout<<~x+1;

is it wrong?????

Reply • Share ›

okay • 10 months ago

int k = n>>31;
return (n - (k*2*n));

1 Reply • Share ›

AMIT • 10 months ago

a smaller and much easier algo

```
/* Paste your code here (You may delete these lines if not writing c
#include<stdio.h>
```

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

```
int abs(int x)
{
    return x*((x>>(sizeof(int)*8-1)) | 1);
}
int main()
{
    printf("abs value=%d\n",abs(5));
    printf("abs value=%d\n",abs(-5));
    return 0;
}
```

1 ▲ | ▼ • Reply • Share ›

anonymous • 11 months ago

int fun(int x)
{//assuming int is 32 bit
((x>>(sizeof(int)*8-1))&&return (-x))||return (x)
}

i think it also work tell me if anythnik goes wrong

▲ | ▼ • Reply • Share ›

cooldude • a year ago

[sourcecode language="java"]
//On 64 bit machine
//first flip and then add 1 for negative to positive
public int absVal(int a)
{
return (((a>>63)^a)+((a>>63)&1));

}

Reply • Share ›

itreallyismE • a year ago

How about return (-2*(x<0)+1)*x?

Reply • Share ›

srinivas ➦ itreallyismE • a year ago

good and simple thinking

```
/* Paste your code here (You may delete these lines if not wri
```

Reply • Share ›

Jagat • a year ago

To those looking for an intuitive explanation to why abs(-ve num) = (mask + n)
Recall that in twos complement notation, a negative number is represented by
counterpart and adding 1 to it. Hence to get the positive counterpart of a negat

Observe that in case of negative numbers mask = 31 bits set; which is equiva
Adding 2^32 - 1 to num is equivalent to subtracting 1 from num, since adding 2
considering the bits has just 32 bits while 2^32 has 33 bits with all 0s except th
Hence (num + mask) == (num - 1)
Doing an xor with the mask is equivalent to flipping the bits.

I'm assuming that the positive scenario is obvious to everyone.

But why go through all these pain?
Simply because we want a unified way of handling both positive and negative r
use an "if" condition.

Admins, please add this explanation to the article. It'll help those who didn't rea

3 Reply • Share ›

v4group • a year ago

Referred http://www.geeksforgeeks.org/a...

```
#include
//#define CHAR_BIT 8

/* This function will return absoulte value of n*/
unsigned int getAbs(int n)
{
int a = n;
if((n >> 31)& 1)
{
a = ~n; //toggle all bit of negative number. we will get +(n-1)
a = -(~a); // add one to that so n-1 + 1 = n
}
return a;
}

/* Driver program to test above function */
int main()
{
int n = -14;
printf("Absoute value of %d is %u", n, getAbs(n));

getchar();
return 0;
}
```

 • Reply • Share ›

Arindam • a year ago

```
#include
#include
int abs(int);
```

```
void main(){
clrscr();
int n;
printf("enter a number to get the absolute value\n");
scanf("%d",&n);
printf("%d",abs(n));

getch();
}
int abs(int n){
if(n<0)
return (~n+1);
else
return n;
}
```


Pranshu • 2 years ago

n = (mask & -x) | (~mask & x)

Pranshu • 2 years ago

If mask = n>>31
then n = (mask & -n) | (~mask & n) Does it solves the problem ?

me • 3 years ago

```
# innclude

int absvalue(int *n)
{
```

```
*n=~*n+1;

if(*n<0)
return(*n);
else
return(**k);
}
int main()
{
int num;
absvalue(&num);
printf("%d",absvalue(&num));

return 0;
}
```

 • Reply • Share ›

PsychoCoder me • 3 years ago

Don't you see the question properly!!!!

```
/* Paste your code here (You may delete these lines if not wri
```

 • Reply • Share ›

rahul • 3 years ago

can any one explain me what its not working

```
 #include <stdio.h>
#define CHAR_BIT 8

/* This function will return absoulte value of n*/
unsigned int getAbs(int n)
```

```
{
  return ((n^1)+1);
}

/* Driver program to test above function */
int main()
{
  int n = -6;
  printf("Absoute value of %d is %u", n, getAbs(n));

  getchar();
  return 0;
}
```

 • Reply • Share ›

Manna rahul • 3 years ago

try -14.. it would not work !!

 • Reply • Share ›

adityaork • 3 years ago

n=(n>>31)*(-n)+(!n>>31)*n

 • Reply • Share ›

anonmous • 3 years ago

how about this 1:

```
int absvalue(int n)
{
int mask=n>>31;
if(mask)
```

```
  return ~n+1;
else
  return n;
}
```

Reply • Share ›

alternateSoln • 3 years ago

How about:

```
/* This function will return absoulte value of n*/
unsigned int getAbs(int n)
{
   int const mask = n &gt;&gt; (sizeof(int) * CHAR_BIT - 1);
   return (n^mask)+1;
}
```

This also seems easier to derive from the explanation in the blog post

Reply • Share ›

Tejas alternateSoln • 9 months ago

I thought the same thing. I don't know why they didn't post this.

Reply • Share ›

sureshpaldia22 • 3 years ago

Since a negative number is always stored as 2's compliment.
And 2's compliment is actually
((1's compliment of number) + (1))

Say, number is 2;

1's compliment of 2: 11111101
+ 00000001

2's compliement 11111110

So, just do 1's compliment and add 1 to it.

```
void abs(int num)
{
     return (num<0) ? (~num+1) : num;
}
```

  • Reply • Share ›

Venki sureshpaldia22 • 3 years ago

@Suresh, your code fails at one place. For example on 32 bit machine

```
int absolute(int num)
{
     return (num < 0) ? (~num+1) : num;
}

int main()
{
    int i = 0x80000000;
    printf("%d\n", absolute(-i));
    return 0;
}
```

Abhirup Ghosh • 3 years ago

```
int absolute (int x)
{
  return (-(x<0) & (-x)) | (-(x>0) & x);
}
```

• Reply • Share ›

Vinay Kumar • 4 years ago

Mask = X >> 31
Result = (~Mask & X) | (Mask & (-X))

• Reply • Share ›

kiran ➦ Vinay Kumar • 3 years ago

Mask = -(X>>31)

• Reply • Share ›

Mayank • 4 years ago

I think simple logic is to subtract number by its double
if number is negative and by zero is positive
so
(n-((n<>31)))
((n<>31)) this part is zero if n>0
and -2n is n<0

• Reply • Share ›

Himanshu Aggarwal • 4 years ago

Hi,

There is no need to define CHAR_BIT explicitly in the code examples above. T predefined in the limits.h header file.

Thanks,
Himanshu

Himanshu Aggarwal • 4 years ago

Hi,

I have found another method to calculate the absolute value, though I have not derivation for that.

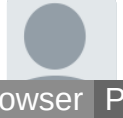
The method calculates absolute value as:
abs(x) = x – (2*x & mask);

where mask= x >> 31;

The following is a C program to get it.

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

/* This function will return absoulte value of n*/
unsigned int getAbs(int n)
{
```

Himanshu Aggarwal • 4 years ago

Thanks for the solution.
Can someone please explain, how this formula or solution has been derived?

Thanks,
Himanshu