# Binary from Decimal (Integer) number using C program

#include <stdio.h>

/\*function declaration

\* name : getBinary

\* Desc : to get binary value of decimal number

\* Parameter : int -integer number

\* return : void

\*/

**void** getBinary(**int**);

**int** main()

{

**int** num=0;

printf("Enter an integer number :");

scanf("%d",&num);

printf("\nBinary value of %d is =",num);

getBinary(num);

**return** 0;

}

/\*Function definition : getBinary()\*/

**void** getBinary(**int** n)

{

**int** loop;

/\*loop=15 , for 16 bits value, 15th bit to 0th bit\*/

**for**(loop=15; loop>=0; loop--)

{

**if**( (1 << loop) & n)

printf("1");

**else**

printf("0");

}

}

Output

Enter an integer number :13

Binary value of 13 is =0000000000001101

# [C program to get minimum number of bits to store an integer number.](https://www.includehelp.com/c-programs/c-program-to-get-minimum-bits-to-store-integer.aspx)

/\*Program to get minimum number of bits to store an integer number.\*/

#include <stdio.h>

/\*function declaration

\* name : countBit

\* Desc : to get bits to store an int number

\* Parameter : int

\* return : int

\*/

**int** countBit(**int**);

**int** main()

{

**int** num;

printf("Enter an integer number :");

scanf("%d",&num);

printf("Total number of bits required = %d\n",countBit(num));

**return** 0;

}

**int** countBit(**int** n)

{

**int** count=0,i;

**if**(n==0) **return** 0;

**for**(i=0; i< 32; i++)

{

**if**( (1 << i) & n)

count=i;

}

**return** ++count;

}

Output

First run:

Enter an integer number :127

Total number of bits required = 7

Second run:

Enter an integer number :13

Total number of bits required = 4

# C program to swap two bits of a byte

**Given a byte (an integer number of 8 bits) and we have to swap its any two bits using C program.**

In this program, we declared an unsigned char type variable to read 8 bits number (byte) and we are swapping two bits (1 and 2) of given number.

**Example:**

Input number: **0x0A** (Hexadecimal)

Binary of input number: **0000 1010**

After swapping of bit 1 and 2

Binary will be: **0000 1100**

Output number will be: **0x0C** (Hexadecimal)

Swapping two bits of a byte using C program

/\*C program to swap two bits of a byte.\*/

    #include <stdio.h>

**int** main()

{

**unsigned** **char** data=0x0A;

// swaping 1st bit to 2nd bit (bit counting 7-0).

// binary of 0x0A is : 0000 1010

data^=(1<<1);

data^=(1<<2);

// data will be : 0000 1100 (0x0C)

printf("\ndata after swap bits : %02X",data);

**return** 0;

}

Output

data after swap bits : 0C

﻿

# C program to check whether all bits of a number are UNSET/LOW?

﻿

Learn: how we can **check that whether all bits of a one byte (8 bits) number are UNSET/LOW using C program**? Here, we are implemented the program for this using **Bitwise AND (&) operator**.

**Give a number of one byte (8 bits) and we have to check whether all bits are UNSET/LOW.**

**Example -1**

Input number: **0**

Binary value: **00000000**

Output: **Yes, all bits are unset**

**Example -2**

Input number: **50**

Binary value: **00110011**

Output: **No, all bits are not unset**

To solved this program, we will use **BITWISE AND (&) operator**, we will traverse bits from 7 to 0 (in case of two bytes number, it would be 15 to 0 and so on...) and check if there is any SET/HIGH bit, then we will break the loop and output will be false that means **"all bits are not UNSET/LOW"**.

**Consider the program:**

﻿

#include <stdio.h>

//function to check whether all bits are

//UNSET/LOW or not?

**int** isAllBitsUnset(**unsigned** **int** num)

{

**int** loop, cnt=0;

**for**(loop=7; loop>=0; loop--)

{

//check, if there is any SET/HIGH bit

**if**( num & (1<<loop))

{

cnt =1;

**break**;

}

}

**if**(cnt==0)

**return** 1; //true

**else**

**return** 0; //false

}

//main function

**int** main()

{

**unsigned** **int** number;

//read number

printf("Enter an integer number (between 0-255): ");

scanf("%d",&number);

**if**(isAllBitsUnset(number))

printf("All bits are UNSET/LOW.\n");

**else**

printf("All bits are not UNSET/LOW.\n");

**return** 0;

}

Output

First Run:

Enter an integer number (between 0-255): 0

All bits are UNSET/LOW.

Second Run:

Enter an integer number (between 0-255): 50

All bits are not UNSET/LOW.

We wrote a function int isAllBitsUnset(unsigned int num), it is taking an unsigned integer type number as an argument and returning 0 if all bits are not UNSET/LOW and 1 if all bits are UNSET/LOW.

**function:**

**int** isAllBitsUnset(**unsigned** **int** num)

{

**int** loop, cnt=0;

**for**(loop=7; loop>=0; loop--)

{

//check, if there is any SET/HIGH bit

**if**( num & (1<<loop))

{

cnt =1;

**break**;

}

}

**if**(cnt==0)

**return** 1; //true

**else**

**return** 0; //false

}

To check the bits, we are traversing loop from 7 to 0 (in case of 8 bits) and checking each bit, whether it is HIGH/SET or not, if any bit is HIGH/SET then value of cnt will be 1 (which is 0 initially, cnt is using as a flag variable) and loop will break.

# C program to swap two words/bytes.

﻿

This program will **swap two bytes/words of an integer number**, here this operation is implemented using bitwise shifting and bit masking.

**Swapping two Bytes/Words using C program**

/\* C program to swap bytes/words of integer number.\*/

#include <stdio.h>

int main()

{

    unsigned int data=0x1234;

    printf("\ndata before swapping : %04X",data);

    data= ((data<<8)&0xff00)|((data>>8)&0x00ff);

    printf("\ndata after swapping  : %04X",data);

    return 0;

}

data before swapping : 1234

data after swapping : 3412

# C program to reverse bits of an integer number.

﻿

This program will **reverse all bits of an integer number**, we will implement this program by creating a User Define Function, that will return an integer number by reversing all bits of passed actual parameter (integer number).

Reversing bits of a number using C program

/\*C program to reverse bits of a number \*/

#include <stdio.h>

unsigned int revBits**(**unsigned int data**)**

**{**

unsigned char totalBits **=** **sizeof(**data**)** **\*** 8**;**

unsigned int revNum **=** 0**,** i**,** temp**;**

**for** **(**i **=** 0**;** i **<** totalBits**;** i**++)**

**{**

temp **=** **(**data **&** **(**1 **<<** i**));**

**if(**temp**)**

revNum **|=** **(**1 **<<** **((**totalBits **-** 1**)** **-** i**));**

**}**

**return** revNum**;**

**}**

int main**()**

**{**

unsigned int num **=** 0x4**;**

printf**(**"\n%u"**,** revBits**(**num**));**

**return** 0**;**

**}**

8912

# C program to counter number of 1’s in an integer number.

﻿

This program will **count total number of 1’s in an integer number**. Here we are counting total number of 1’s in an integer number using a User Define Function.

Counting number of 1’s using C program

/\*C program to count number of 1's in a number \*/

#include <stdio.h>

**int** count1s(**unsigned** **int** num)

{

**unsigned** **char** i;

**int** count=0;

**unsigned** **char** totalBits=**sizeof**(num)\*8;

**for**(i=0;i< totalBits;i++)

{

**if**( num & (1<< i) )

count++;

}

**return** count;

}

**int** main()

{

**unsigned** **int** data=0x58;

printf("\nTotal number of 1's are : %d\n",count1s(data));

**return** 0;

}

**Output**

Total number of 1's are : 3

# C program to swap two nibbles of a byte.

This program will **swap two nibbles of a byte**, as we know that one byte has 8 bits or 2 nibbles. Hence one nibble has 4 bits, by shifting 4, 4 bits we can swap nibbles of a byte.

Swapping nibbles of a byte using C program

/\*C program to swap two nibbles of a given byte\*/

#include <stdio.h>

/\* function : swapTwoNibbles, to swap two nibbles of a given byte.\*/

unsigned char swapTwoNibbles**(**unsigned char n**)**

**{**

unsigned char num**;**

num**=** **(** **(**n **&** 0x0F**)<<**4 **|** **(**n **&** 0xF0**)>>**4 **);**

**return** num**;**

**}**

int main**()**

**{**

unsigned char number**;**

unsigned char revNumber**;**

printf**(**"Enter an integer number (One byte number):"**);**

scanf**(**"%u"**,&**number**);**

revNumber**=**swapTwoNibbles**(**number**);**

printf**(**"\nEntered Number was : %u \nNumber after swapping nibbles : %u"**,**number**,**revNumber**);**

**return** 0**;**

**}**

Enter an integer number (One byte number):200

Entered Number was : 200

Number after swapping nibbles : 140

# C program to demonstrate example of left shift (<<) operator.

﻿

This program will demonstrate **example of Left Shift (<<) Operator in C programming language**. Using this program we will show how to perform left shift operation using C program.

Example of Left Shift (<<) Operator in C program

/\* C Program to demonstrate example of left shift (<<) operator.\*/

#include <stdio.h>

int main**()**

**{**

unsigned int num**=** 0xff**;**

printf**(**"\nValue of num = %04X before left shift."**,**num**);**

/\*shifting 2 bytes left\*/

num **=** **(**num**<<**2**);**

printf**(**"\nValue of num = %04X after left shift."**,**num**);**

**return** 0**;**

**}**

Value of num = 00FF before left shift.

Value of num = 03FC after left shift.

Left Shift Operator (<<) is a bitwise operator, which perform operation on bits. It is used to shift given number of bytes in the left and inserts 0’s in the right.   
Binary of 0xFF in (in 4 bytes format) - 0000 0000 1111 1111.  
After 2 bytes left shift (in 4 bytes format) – 0000 0011 1111 1100, which is equivalent of 0x03FC.

# C program to demonstrate example of right shift (>>) operator.

﻿

This program will demonstrate **example of Right Shift (>>) Operator in C programming language**. Using this program we will show how to perform right shift operation using C program.

Example of Right Shift (>>) Operator in C program

/\* C Program to demonstrate example right shift (>>) operator.\*/

#include <stdio.h>

int main**()**

**{**

unsigned int num**=** 0xff**;**

printf**(**"\nValue of num = %04X before right shift."**,**num**);**

/\*shifting 2 bytes right\*/

num **=** **(**num**>>**2**);**

printf**(**"\nValue of num = %04X after right shift."**,**num**);**

**return** 0**;**

**}**

Value of num = 00FF before right shift.

Value of num = 003F after right shift.

Right Shift Operator (>>) is a bitwise operator, which perform operation on bits. It is used to shift given number of bytes in the right and inserts 0’s in the left.   
Binary of 0xFF in (in 4 bytes format) - 0000 0000 1111 1111.  
After 2 bytes right shift (in 4 bytes format) – 0000 0000 0011 1111, which is equivalent of 0x003F.

# C program to set/clear (high/low) bits of a number.

﻿

This program will **set or clear (high or low) bits of a number**, this operation can be performed using Bitwise OR (|) and Bitwise AND (&) operators.

High/Low (Set/Clear) bits of a number using C program

/\* C program to set and clear bits of a number.\*/

#include <stdio.h>

int main**()**

**{**

unsigned int num**=** 0x0C**;**

/\*set 0th and 1st bits\*/

num **|=** **(**1 **<<** 0**);** /\*set 0th bit\*/

num **|=** **(**1 **<<** 1**);** /\*set 1st bit\*/

printf**(**"\nValue of num = %04X after setting 0th and 1st bits."**,**num**);**

/\*clear 0th and 1st bits\*/

num **&=** **~(**1 **<<** 0**);** /\*set 0th bit\*/

num **&=** **~(**1 **<<** 1**);** /\*set 1st bit\*/

printf**(**"\nValue of num = %04X after clearing 0th and 1st bits."**,**num**);**

**return** 0**;**

**}**

Value of num = 000F after setting 0th and 1st bits.

Value of num = 000C after clearing 0th and 1st bits.

# C program to swap two numbers using bitwise XOR operator.

﻿

This program will **swap two integer numbers using Bitwise XOR Operators**. Numbers are swapping in a User Define Function with the help of Call by Pointers.

Swap two numbers using Bitwise XOR Operator in C

/\*C program to swap two numbers using bitwise operator.\*/

#include <stdio.h>

void swap**(**int **\***a**,** int **\***b**);** //function declaration

int main**()**

**{**

int a**,**b**;**

printf**(**"Enter first number: "**);**

scanf**(**"%d"**,&**a**);**

printf**(**"Enter second number: "**);**

scanf**(**"%d"**,&**b**);**

printf**(**"Before swapping: a=%d, b=%d\n"**,**a**,**b**);**

swap**(&**a**,&**b**);**

printf**(**"After swapping: a=%d, b=%d\n"**,**a**,**b**);**

**return** 0**;**

**}**

//function definition

void swap**(**int **\***a**,**int **\***b**)**

**{**

**\***a **=** **\***a **^** **\***b**;**

**\***b **=** **\***a **^** **\***b**;**

**\***a **=** **\***a **^** **\***b**;**

**}**

Enter first number: 10

Enter second number: 20

Before swapping: a=10, b=20

After swapping: a=20, b=10