**1.C program to check Least Significant Bit (LSB) of a number using bitwise operator**

#include <stdio.h>

int main()

{

    int num;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    //If (num & 1) evaluates to 1

    if(num & 1)

        printf("Least Significant Bit (LSB) of %d is set (1).", num);

    else

        printf("Least Significant Bit (LSB) of %d is unset (0).", num);

    return 0;

}

**2.C program to get the nth bit of a number**

#include <stdio.h>

int main()

{

    int num, n, bitStatus;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    //Reads the bit number you want to check

    printf("Enter nth bit to check (0-31): ");

    scanf("%d", &n);

    //Shifting bits to right n times and ANDing with 1

    bitStatus = (num >> n) & 1;

    printf("The %d bit is set to %d", n, bitStatus);

    return 0;

}

**3.C program to set the nth bit of a number**

#include <stdio.h>

int main()

{

    int num, n, newNum;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    //Reads the bit number you want to set

    printf("Enter nth bit to set (0-31): ");

    scanf("%d", &n);

    //Right shift 1 to n times and perform bitwise OR with number

    newNum = (1 << n) | num;

    printf("Bit set successfully.\n\n");

    printf("Number before setting %d bit: %d (in decimal)\n", n, num);

    printf("Number after setting %d bit: %d (in decimal)\n", n, newNum);

    return 0;

}

**4.C program to clear the nth bit of a number**

#include <stdio.h>

int main()

{

    int num, n, newNum;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    //Reads the bit number you want to clear

    printf("Enter nth bit to clear (0-31): ");

    scanf("%d", &n);

    /\*

     \* Left shifts 1 to n times

     \* Perform complement of above

     \* then perform bitwise AND with number and result of above

     \*/

    newNum = num & (~(1 << n));

    printf("Bit cleared successfully.\n\n");

    printf("Number before clearing %d bit: %d (in decimal)\n", n, num);

    printf("Number after clearing %d bit: %d (in decimal)\n", n, newNum);

    return 0;

}

**5.C program to toggle nth bit of a number**

#include <stdio.h>

int main()

{

    int num, n, newNum;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    //Reads the bit number you want to toggle

    printf("Enter nth bit to toggle (0-31): ");

    scanf("%d", &n);

    /\*

     \* Left shifts 1 to n times

     \* then perform bitwise XOR with number and result of above

     \*/

    newNum = num ^ (1 << n);

    printf("Bit toggled successfully.\n\n");

    printf("Number before toggling %d bit: %d (in decimal)\n", n, num);

    printf("Number after toggling %d bit: %d (in decimal)\n", n, newNum);

    return 0;

}

**6.C program to find highest order set bit in a number**

#include <stdio.h>

#define INT\_SIZE sizeof(int) \* 8 //Integer size in bits

int main()

{

    int num, order=0, i;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    //Loops over each bit of the integer

    for(i=0; i<INT\_SIZE; i++)

    {

        //If the current bit is set

        if((num>>i) & 1)

            order = i;

    }

    printf("Highest order set bit in %d is %d", num, order);

    return 0;

}

**7.C program to find lowest order set bit in a number**

#include <stdio.h>

#define INT\_SIZE sizeof(int) \* 8 //Integer size in bits

 int main()

{int num, order, i;

 //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

  //Initially sets the order to max size of integer

    order = INT\_SIZE - 1;

 //Loops over each bit of the integer

    for(i=0; i<INT\_SIZE; i++)

    {

        //If the current bit is set

        if((num>>i) & 1)

        {

            order = i;

            break; //No need to check further

        }

    }

    printf("Lowest order set bit in %d is %d", num, order);

  return 0;

}

**8.C program to count trailing zeros in a binary number using bitwise operator**

#include <stdio.h>

#define INT\_SIZE sizeof(int) \* 8

int main()

{

    int num, count=0, i;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    //Iterate over each bit of the number

    for(i=0; i<INT\_SIZE; i++)

    {

        //If a set bit is found

        if((num >> i ) & 1)

        {

            //No need to run further

            break;

        }

        count++;

    }

    printf("Total number of trailing zeros in %d is %d.", num, count);

    return 0;

}

**ANOTHE APPROACH**

#include <stdio.h>

int main()

{

    int num, count=0;

 //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    while(!(num & 1))

    {

        count++;

        num >>= 1;

    }

    printf("Total number of trailing zeros = %d.", count);

 return 0;

}

**9.C program to count leading zeros in a binary number using bitwise operator**

#include <stdio.h>

#define INT\_SIZE sizeof(int) \* 8

int main()

{

    int num, count, msb, i;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    count = 0;

    //Equivalent to

    //10000000 00000000 00000000 00000000

    msb = 1 << (INT\_SIZE - 1);

    //Iterate over each bit

    for(i=0; i<INT\_SIZE; i++)

    {

        //If a leading set bit is found

        if((num << i) & msb)

        {

            //No need to run further

            break;

        }

        count++;

    }

    printf("Total number of leading zeros in %d is %d", num, count);

    return 0;

}

**10.C program to count flip all bits of a binary number using bitwise operator**

#include <stdio.h>

int main()

{

    int num, flippedNumber;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    flippedNumber = ~num;

    printf("Original number = %d (in decimal)\n", num);

    printf("Number after bits are flipped = %d (in decimal)", flippedNumber);

    return 0;

}

**11.C program to count total number of zeros and ones in a binary number using bitwise operator**

#include <stdio.h>

#define INT\_SIZE sizeof(int) \* 8 //Total number of bits in integer

int main()

{

    int num, temp, zeros, ones, i;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    temp = num;

    zeros = 0;

    ones = 0;

    for(i=0; i<INT\_SIZE; i++)

    {

        //If LSB is set then increment ones otherwise increment zeros

        if(temp & 1)

            ones++;

        else

            zeros++;

        //Right shift bits of temp to one position

        temp >>= 1;

    }

    printf("Total number of zeros bits is %d (in %d byte integer representation)\n", zeros, sizeof(int));

    printf("Total number of ones bits is %d (in %d byte integer representation)", ones, sizeof(int));

    return 0;

}

**12.C program to convert decimal to binary number system using bitwise operator**

#include <stdio.h>

#define INT\_SIZE sizeof(int) \* 8 //Size of int in bits

int main()

{

    int num, index, i;

    int bin[INT\_SIZE];

    printf("Enter any number: ");

    scanf("%d", &num);

    index = INT\_SIZE;

    //Converts decimal to binary

    while(index!=0)

    {

        index--;

        //Store 1 if LSB is set otherwise 0

        bin[index] = num & 1;

        //Shift bits of num to one position right

        num >>= 1;

    }

    //Prints the converted binary

    printf("Converted binary (in %d byte integer representation): ", sizeof(int));

    for(i=0; i<INT\_SIZE; i++)

    {

        printf("%d", bin[i]);

    }

    return 0;

}

**13.C program to swap two numbers using bitwise operator**

#include <stdio.h>

int main()

{

    int num1, num2;

    //Reads two numbers from user

    printf("Enter any two numbers: ");

    scanf("%d%d", &num1, &num2);

    printf("Original value of num1 = %d\n", num1);

    printf("Original value of num2 = %d\n", num2);

    num1 ^= num2;

    num2 ^= num1;

    num1 ^= num2;

    printf("Num1 after swapping = %d\n", num1);

    printf("Num2 after swapping = %d\n", num2);

    return 0;

}

**14.C program to check even or odd number using bitwise operator**

#include <stdio.h>

int main()

{

    int num;

    //Reads a number from user

    printf("Enter any number: ");

    scanf("%d", &num);

    if(num & 1)

    {

        printf("%d is odd.", num);

    }

    else

    {

        printf("%d is even.", num);

    }

    return 0;

}

**15.C program to find 2's complement of a binary number**

#include <stdio.h>

#include <string.h>

#define SIZE 8

int main()

{

    char binary[SIZE + 1], onesComp[SIZE + 1], twosComp[SIZE + 1];

    int i, carry=1;

    /\*

     \* Reads binary number from user

     \*/

    printf("Enter any %d bit binary value: ", SIZE);

    gets(binary);

    /\*

     \* Finds the 1's complement of the binary number

     \*/

    for(i=0; i<SIZE; i++)

    {

        if(binary[i]=='1')

        {

            onesComp[i] = '0';

        }

        else if(binary[i]=='0')

        {

            onesComp[i] = '1';

        }

    }

    onesComp[SIZE] = '\0';

    /\*

     \* Adds 1 to the 1's complement of the binary number to get 2's complement

     \*/

    for(i=SIZE-1; i>=0; i--)

    {

        if(onesComp[i]=='1' && carry==1)

        {

            twosComp[i] = '0';

        }

        else if(onesComp[i]=='0' && carry==1)

        {

            twosComp[i] = '1';

            carry = 0;

        }

        else

        {

            twosComp[i] = onesComp[i];

        }

    }

    twosComp[SIZE] = '\0';

    printf("\nOriginal binary value = %s\n", binary);

    printf("One's complement = %s\n", onesComp);

    printf("Two's complement = %s", twosComp);

    return 0;

}

**16.C program to find 1's complement of a binary number**

#include <stdio.h>

#include <string.h>

#define SIZE 8

int main()

{

    char binary[SIZE + 1], onesComp[SIZE + 1];

    int i, error=0;

    /\*

     \* Reads binary value from user

     \*/

    printf("Enter any %d bit binary value: ", SIZE);

    gets(binary);

    /\*

     \* Stores all inverted bits of binary value to onesComp

     \*/

    for(i=0; i<SIZE; i++)

    {

        if(binary[i]=='1')

        {

            onesComp[i] = '0';

        }

        else if(binary[i]=='0')

        {

            onesComp[i] = '1';

        }

        else

        {

            printf("Invalid Input");

            error = 1;

            break;

        }

    }

    onesComp[SIZE] = '\0';

    if(error==0)

    {

        printf("\nOriginal binary = %s\n", binary);

        printf("Ones complement = %s", onesComp);

    }

    return 0;

}

**17.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;

}

**18.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("\nTotal 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;

}