**Basic C Programs**

## Example 1: Displaying Hello World

#include <stdio.h>

int main()

{

/\* printf function displays the content that is

\* passed between the double quotes.

\*/

printf("Hello World");

return 0;

}

Output:

Hello World

1. #include <stdio.h> – This statement tells compiler to include this stdio.h file in the program. This is a standard input output file that contains the definitions of common input output functions such as scanf() and printf(). In the above program we are using printf() function.

2. int main() – Here main() is the function name and int is the return type of this function. Every C program must have this function because the execution of program begins with the main() function. The 0 return value of this function represents successful execution of program while the return value 1 represents the unsuccessful execution of program. This is the reason we have return 0; statement at the end of this main function.

## Example 2:

/\* Description: A program to check whether the input

\* integer number is positive or negative.

\*/

#include <stdio.h>

void main()

{

int num;

printf("Enter a number: \n");

scanf("%d", &num);

if (num > 0)

printf("%d is a positive number \n", num);

else if (num < 0)

printf("%d is a negative number \n", num);

else

printf("0 is neither positive nor negative");

}

Output 1:

Enter a number:

0

0 is neither positive nor negative

Output 2:

Enter a number:

-3

-3 is a negative number

## Example 3:

### Reverse a given number using Recursion:

In this program, we are calling a user defined function “reverse\_function”, the function is calling itself recursively.

#include<stdio.h>

int main(){

int num,reverse\_number;

//User would input the number

printf("\nEnter any number:");

scanf("%d",&num);

//Calling user defined function to perform reverse

reverse\_number=reverse\_function(num);

printf("\nAfter reverse the no is :%d",reverse\_number);

return 0;

}

int sum=0,rem;

reverse\_function(int num){

if(num){

rem=num%10;

sum=sum\*10+rem;

reverse\_function(num/10);

}

else

return sum;

return sum;

}

Output:

Enter any number: 23456

After reverse the no is :65432

## Example 4:

## Program to find largest of three input numbers

The program will prompt user to input three integer numbers and based on the input, it would compare and display the greatest number as output. In this program num1, num2 & num3 are three int variables that represents number1, number2 and number3 consecutively.

#include<stdio.h>

int main()

{

int num1,num2,num3;

//Ask user to input any three integer numbers

printf("\nEnter value of num1, num2 and num3:");

//Store input values in variables for comparsion

scanf("%d %d %d",&num1,&num2,&num3);

if((num1>num2)&&(num1>num3))

printf("\n Number1 is greatest");

else if((num2>num3)&&(num2>num1))

printf("\n Number2 is greatest");

else

printf("\n Number3 is greatest");

return 0;

}

Output:

Enter value of num1, num2 and num3: 15 200 101

Number2 is greatest

## Example 5:

### Fibonacci Series in C using loop

A simple for loop to display the series. Program prompts user for the number of terms and displays the series having the same number of terms.

#include<stdio.h>

int main()

{

int count, first\_term = 0, second\_term = 1, next\_term, i;

//Ask user to input number of terms

printf("Enter the number of terms:\n");

scanf("%d",&count);

printf("First %d terms of Fibonacci series:\n",count);

for ( i = 0 ; i < count ; i++ )

{

if ( i <= 1 )

next\_term = i;

else

{

next\_term = first\_term + second\_term;

first\_term = second\_term;

second\_term = next\_term;

}

printf("%d\n",next\_term);

}

return 0;

}

Output:

Enter the number of terms: 8

First 8 terms of Fibonacci series:

0

1

1

2

3

5

8

13

### Program to display Fibonacci series using recursion

Here we are using a user defined function fibonacci\_series() that calls itself recursively, in order to display series for the entered number of terms.

#include<stdio.h>

int fibonacci\_series(int);

int main()

{

int count, c = 0, i;

printf("Enter number of terms:");

scanf("%d",&count);

printf("\nFibonacci series:\n");

for ( i = 1 ; i <= count ; i++ )

{

printf("%d\n", fibonacci\_series(c));

c++;

}

return 0;

}

int fibonacci\_series(int num)

{

if ( num == 0 )

return 0;

else if ( num == 1 )

return 1;

else

return ( fibonacci\_series(num-1) + fibonacci\_series(num-2) );

}

Output:

Enter number of terms: 6

Fibonacci series:

0

1

1

2

3

5

## Example 6:

This Program prompts user for entering any integer number, finds the factorial of input number and displays the output on screen. We will use a recursive user defined function to perform the task. Here we have a function find\_factorial that calls itself in a recursive manner to find out the factorial of input number. We have involved the user interaction in the below program, however if you do not want that part then you can simply assign an integer value to variable num and ignore the scanf statement. In short you can tweak it in any way you want, the logic would be the same for each case.

### Program to find factorial

 /\* Program Name: Find Factorial

\*/

#include<stdio.h>

int find\_factorial(int);

int main()

{

int num, fact;

//Ask user for the input and store it in num

printf("\nEnter any integer number:");

scanf("%d",&num);

//Calling our user defined function

fact =find\_factorial(num);

//Displaying factorial of input number

printf("\nfactorial of %d is: %d",num, fact);

return 0;

}

int find\_factorial(int n)

{

//Factorial of 0 is 1

if(n==0)

return(1);

//Function calling itself: recursion

return(n\*find\_factorial(n-1));

}

**Output:**

Enter any integer number: 4

factorial of 4 is: 24

## Example 7:

Upon execution of below program, the user would be asked to provide the from & to range and then the program would display all the prime numbers in sequential manner for the provided range. Using this program you can find out the prime numbers between 1 to 100, 100 to 999 etc. You just need to input the range, for e.g. if you want the prime numbers from 100 to 999 then enter numbers 100 and 999 when program prompts for input.

### Program to find prime numbers

#include <stdio.h>

int main()

{

int num1, num2, flag\_var, i, j;

/\* Ask user to input the from/to range

\* like 1 to 100, 10 to 1000 etc.

\*/

printf("Enter two range(input integer numbers only):");

//Store the range in variables using scanf

scanf("%d %d", &num1, &num2);

//Display prime numbers for input range

printf("Prime numbers from %d and %d are:\n", num1, num2);

for(i=num1+1; i<num2; ++i)

{

flag\_var=0;

for(j=2; j<=i/2; ++j)

{

if(i%j==0)

{

flag\_var=1;

break;

}

}

if(flag\_var==0)

printf("%d\n",i);

}

return 0;

}

**Output:**

Enter two range(input integer numbers only):Prime numbers from 1 and 50 are: 1 50

2

3

5

7

11

13

17

19

23

29

31

37

41

43

47

## Example 8:

A number is called as Armstrong number if sum of cubes of digits of number is equal to the number itself. In the below C program, we are checking whether the input number is Armstrong or not.

#include<stdio.h>

int main()

{

int num,copy\_of\_num,sum=0,rem;

//Store input number in variable num

printf("\nEnter a number:");

scanf("%d",&num);

/\* Value of variable num would change in the

below while loop so we are storing it in

another variable to compare the results

at the end of program

\*/

copy\_of\_num = num;

/\* We are adding cubes of every digit

\* and storing the sum in variable sum

\*/

while (num != 0)

{

rem = num % 10;

sum = sum + (rem\*rem\*rem);

num = num / 10;

}

/\* If sum of cubes of every digit is equal to number

\* itself then the number is Armstrong

\*/

if(copy\_of\_num == sum)

printf("\n%d is an Armstrong Number",copy\_of\_num);

else

printf("\n%d is not an Armstrong Number",copy\_of\_num);

return(0);

}

**Output:**

Enter a number: 370

370 is an Armstrong Number

You can verify the result like this:

370 = 3\*3\*3 + 7\*7\*7 + 0\*0\*0

= 27 + 343 + 0

= 370

## Example 9:

If a number remains same, even if we reverse its digits then the number is known as palindrome number. For example 12321 is a palindrome number because it remains same if we reverse its digits. In this article we have shared two C programs to check if the input number is palindrome or not. 1) using while loop 2) using recursion.

#### Program 1: check palindrome using while loop

/\* Program to check if a number is palindrome or not

\* using while loop

\*/

#include <stdio.h>

int main()

{

int num, reverse\_num=0, remainder,temp;

printf("Enter an integer: ");

scanf("%d", &num);

/\* Here we are generating a new number (reverse\_num)

\* by reversing the digits of original input number

\*/

temp=num;

while(temp!=0)

{

remainder=temp%10;

reverse\_num=reverse\_num\*10+remainder;

temp/=10;

}

/\* If the original input number (num) is equal to

\* to its reverse (reverse\_num) then its palindrome

\* else it is not.

\*/

if(reverse\_num==num)

printf("%d is a palindrome number",num);

else

printf("%d is not a palindrome number",num);

return 0;

}

## Example 10:

If a number is exactly divisible by 2 then its an even number else it is an odd number. In this article we have shared two ways(Two C programs) to check whether the input number is even or odd. 1) Using Modulus operator(%) 2) Using Bitwise operator.

#### Program 1: Using Modulus operator

/\* Program to check whether the input integer number

\* is even or odd using the modulus operator (%)

\*/

#include<stdio.h>

int main()

{

// This variable is to store the input number

int num;

printf("Enter an integer: ");

scanf("%d",&num);

// Modulus (%) returns remainder

if ( num%2 == 0 )

printf("%d is an even number", num);

else

printf("%d is an odd number", num);

return 0;

}