[](http://www.s2d6.com/x/?x=c&z=s&v=6896031&r=%5bRANDOM%5d&k=%5bNETWORKID%5d)

**Decision Making and Looping**

Execution of a statement or set of statement repeatedly is called as looping.

The loop may be executed a specified number of times and this depends on the satisfaction of a test condition.

A program loop is made up of two parts one part is known as body of the loop and the other is known as control condition.

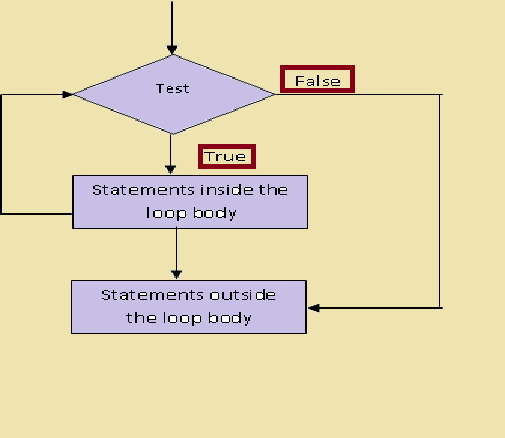
Depending on the control condition statement the statements within the loop may be executed repeatedly.

Depending on the position of the control statement in the loop, a control structure may be classified either as an entry controlled loop or as an exit controlled loop.

**Entry Controlled Loop:**

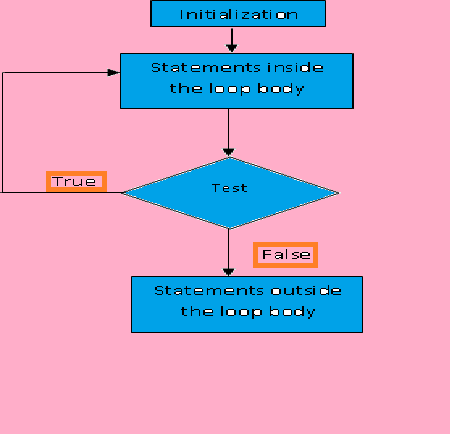
When the control statement is placed before the body of the loop then such loops are called as entry controlled loops.

If the test condition in the control statement is true then only the body of the loop is executed.

If the test condition in the control statement is not true then the body of the loop will not be executed. If the test condition fails in the first checking itself the body of the loop will never be executed.  
  
  
  
**Exit Controlled Loop:**

When the control statement is placed after the body of the loop then such loops are called as exit controlled loop.   
In this the body of the loop is executed first then the test condition in the control statement is checked.   
If it is true then the body of the loop is executed again.

If the test condition is false, the body of the loop will not be executed again. In exit controlled loops even if the test condition fails in the first attempt itself the body of the loop is executed at least once.



**Following are the steps in a looping process:**

1. Initialization of loop control variable
2. Test of a specific condition for the execution of the loop
3. Execution of the statements in the body of the loop
4. Altering the value of the loop control variable

In the beginning the loop control variable is initialized after that step2, step3 and step4 are carried out till specified test condition becomes false.  
The test may be either to determine whether the loop has been repeated the specified number of times or to determine whether a particular condition has been met. The C language provides 3 loop structures.

1. while loop.
2. do loop.
3. for loop.

**Control Variables:**  
These are the variables which control the number of times a loop will be executed.

**The while statement:** The basic format of the while statement is as given below

 **while (test condition)  
{  
body of the loop;  
}**

This is an entry controlled loop. In this the test condition is placed before the body of the loop. The test condition is evaluated first and if it is true, then only the body of the loop will be executed.

In the body of the loop there will be statement, which will alter the value of the loop control variable. After the execution of the body of the loop the test condition is again evaluated and if it is true then only the body of the loop will be executed. Then the body of the loop will be executed till the test condition becomes false.

Once the test condition fails the statement following the body of the loop will be executed. The body of the loop will begin with a opening brace and ends with a closing brace. The body of the loop may have one or more statements.  
The loop control variable is used in the test condition to check whether the body of the loop is to be executed or not.   
If the test condition fails in the beginning itself the body of the loop will never be executed.

Segment of a program is given below:  
**i = 0;**   
**while (i<=10)  
{  
printf ("%d\n", i);  
i++;  
}**

In the above example, the body of while loop is executed 11 times. Since the test of a condition is carried out before the loop is executed, the body of the loop may not be executed at all if the condition is not satisfied at the very first instance.

 **The do statement:**

 In some occasions it might be necessary to execute the body of the loop before the test is performed. Such situations can be handled with the help of do statement.  
**Syntax:**

**do  
{  
body of the loop;  
}   
while (test condition);**

When the do statement is encountered, the body of the loop is executed. The test condition is present at the end of the body of the loop. The test condition is evaluated and if it is true, the body of the loop is executed again. This process will continue till the test condition becomes false. Now the statement next to the test condition will be executed. Since the test condition is placed at the end of the body of the loop, this do loop is called as exit controlled loop. Therefore, even though the test condition fails at the first attempt itself, the body of the loop is executed at least once.

**Example**:

i=0;  
do  
{  
printf ("%d\n", i);  
i++;  
} while (i< 10);

The above loop is executed 11 times because the value of I is initialized to zero.  
  
Consider the following case where i is initialized to 11, still the body of the loop is executed at least once although the test condition fails at the first attempt itself.

i=11;  
do  
{  
printf ("%d\n",i);  
i++;  
} while (i< 10);

**/\* Program to test do loop\*/**  
# include <stdio.h>  
# include <conio.h>  
void main(){  
char c;  
c=' ';  
do  
{  
clrscr();  
printf("Welcome to MGM\n");  
c=getch();  
} while(c!= 'q');  
}  
  
**Note:**

Test conditions in loop constructs can also have compound statements as shown below.

Example:  
while (i>0 && k<5)  
{  
printf ("%d\n", i);  
i --;  
}

**for loop:**

 This is an entry controlled loop which provides compact loop control structure. In this, the initialization of loop control variable, the test condition and change in the value of loop control variable is done in the single statement separated by semicolon.

**Syntax:**

for (initialization; test condition; increment)  
{  
body of the loop;  
}

***The for loop is executed as given below:***

 The initialization of the loop control variable is done first with the help of the assignment statement such as i = 1 or j = 0. In this assignment statement, i and j are known as loop control variables. This is the first part of for statement.

 The second part of the for statement is the test condition. The test condition is evaluated and if it is true, the body of the for loop is executed. If the test condition is false, the body of the loop will not be executed and the control will be transferred to the statement immediately after the body of the loop.

 After the execution of the body of the loop the control is transferred to the third part of the for statement where the value of the loop control variable is changed, say incremented or decremented depending on the statement in this part. Now again the test condition is evaluated and if it is true, then, the body of the loop is executed. Otherwise the loop is terminated and the control will be transferred to the statement that follows the body of the loop.

**Example 1:**  
Output of this loop construct

for (i=0; i<5; i++) 0  
{ 1  
printf ("%d\n", i); 2  
} 3  
4  
for (i=4; i>=0; i--) 4  
{ 3  
printf ("%d\n", i); 2  
} 1  
0

In Example 1 the output is printed from 0 to 4 because the value of i is incremented by one and the loop is executed five times.  
The for loop in example 2 is also executed 5 times but it prints 4 to 0.  
Semicolons must separate the 3 sections enclosed within the paranthesis. Note that there is no semicolon at the end of increment section (i.e., i++, i --). Since the conditional test is always performed at the beginning of the loop, the body of the loop may not be executed at all if the condition fails at the beginning itself. So it is entry controlled loop.  
**Example:**  
for (i=6; i>7; i++)  
{  
printf ("%d\n", i);  
}

The above loop will not be executed at all as the condition fails in the first attempt itself.  
One of the important points about the for loop is that all the 3 sections, namely initialization, testing and assigning (incrementing & decrementing) are placed in the for statement itself.  
  
**Additional Features of for loop:**

 The for loop in C has several capabilities that are not found in other loop constructs. More than 1 variable can be initialized at a time in the for statement.  
  
**Example:**

p=1  
for (i=0; i<5; i++)

The above statement can be written as,

for (p=1, i=0; i<5; i++)

In the above for statement the initialization section has 2 parts i.e., p=1 and i=0 separated by, (comma). Like the initialization section, the incremenation section may also have more that one part.  
  
**Example:**

for (n=1, m=10; n<=m; n++, m--)

The multiple arguments in the increment section are separated by commas. Another feature of the for loop is that the test condition may have any compound relation and the testing need not be limited to the loop control variable.  
  
**Example:**

for (i=0; i<10 && k==2; i++)

In for loop it is also possible to use expression in the assignment statement of initialization and increment statement.  
  
**Example:**

for (i=m+n; i<10; i=m++)

Another unique aspect of for loop is that one or more sections can be omitted if necessary.  
i=0;  
for (; i<5;)  
{  
printf ("%d\n", i);  
i++;  
}  
  
Consider the for loop given above, both the initialization and increment section is omitted in the for loop. The initialization is done before the for loop and the control variable is incremented inside the loop. In such cases the sections are kept blank however the semicolons separating the sections must remain.  
  
**Null statement or Empty statement:**

When loop does not contain any statement instead if it contains just a semicolon then such statement is called as Null statement or empty statement. A time delay can be set up using a null statement in the for loop.  
  
**Example:**

for (i=0; i<100000; i++)

 The above loop is executed 100000 times without producing any output. It simply causes time delay. The body of the loop contains only semicolon and it is known as null statement.  
  
**Nesting of loop:**  
Placing one loop within another is called as Nesting of loops. This nesting can be achieved with any of the loop constructs namely for, while, or do statements. You can even mix different loop constructs.  
Example:

 for (i=5; i<20; i++)  
{  
for (j=5; j<20; j++)   
{   
statements;   
}   
statements;   
}

When the value of i is equal to 5 then the for with j will take the value from j=5 to j=19. Again for the i value of 6, j will take the value from j=5 to j=19 this continues till the value of i becomes 19.

i=0;   
while (i<10)   
{   
j=1;   
while (j<20)   
{   
statements;   
j++;   
}   
i++;   
}

For each value of I starting from 0 till 9 j will take the value from 1 to 19.  
  
**Jumps in loops:**  
Transferring the execution control from one statement to another statement within the loop or from within the loop to outside the loop is possible in C. This kind of jumps can be achieved by break, goto and continue statements.  
  
**break:**  
Using this statement an early exit from a loop can be achieved. That is when the break statement is encountered within a loop then the loop is exited immediately and the statements following the loop will be executed. When a break is encountered within a nested loop, the loop in which this statement is present only that loop will be exited.  
That means the break will exit only one loop.  
Syntax:  
break;  
(a) i=0;  
while (i<10>  
{  
:  
:  
if (condition)  
break;  
:  
:  
}  
n=7;  
(b) i=0;  
do  
{  
:  
:  
if (condition)  
break;  
:  
:  
} while (i<10);  
n=7;  
(c) while (i<15)  
{  
:  
for (j=0; j<10; j++)  
{  
:  
if (condition)  
break;  
:  
}  
n=7; (d) for (i=0; i<10; i++)  
{  
:  
if(condition)  
break;  
}  
n=7;  
In all the above cases the break statement will make the control exit the loop in which it is residing. Thus it executes the statement n=7 in all the cases.  
Since goto can be used to transfer the control to any place in a program it can be used to provide a branching within a loop or exit from deeply nested loop. A simple break statement could not work here.  
Example 1:  
while (i<15)  
{  
:  
:  
for (j=0; j<10; j++)  
{  
:  
:  
if (condition)   
goto xyz;  
:  
:  
xyz:  
r=4  
}  
n=7;  
:  
}  
m=8;  
Example 2:  
while (i<15)  
{  
.  
.  
.  
for(j=0; j<10; j++)  
{  
. while(p< 6)  
{  
.  
if(condition)   
goto xyz;  
.  
.}  
.  
k=4;  
}  
}  
n=7;  
xyz:  
m=8;  
In example1 the goto statement will transfer the control to lable xyz within the same loop skipping the part of the loop.  
  
  
**continue:**

 In a loop on some occasions depending on some condition we may have to skip a part of the body of the loop and proceed with the next iteration.   
This can be achieved by the statement continue.   
  
**Syntax:**

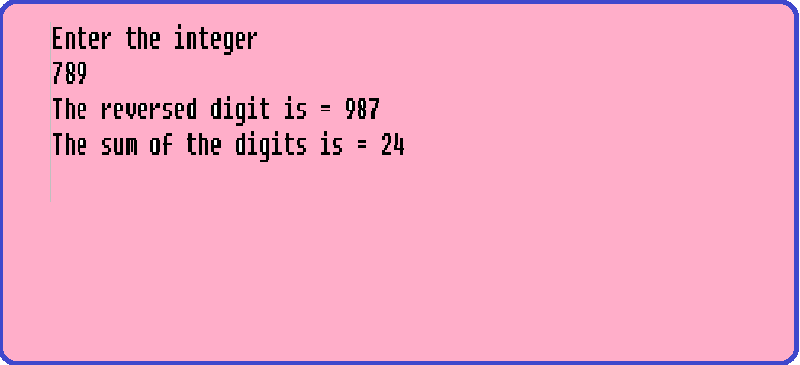
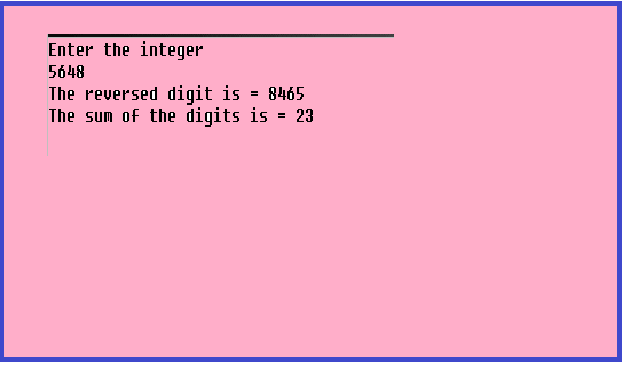
*continue;*

**Example 1:**

while (i<10)  
{  
:  
if (condition)  
continue;  
:  
}  
n=7;  
Example 2:  
do  
{   
:   
if (condition)  
continue;  
:   
} while (i<10);   
while (i<15)  
{  
:  
for (j=0; j<10; j++)  
{  
:  
if (condition)  
continue;  
:  
}  
:  
}

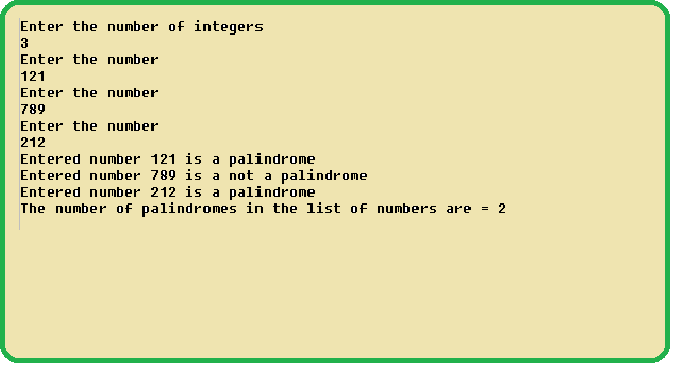
**/\*C Program to reverse the digit of a number and to find the sum of the digits\*/**

#include<stdio.h>   
#include <conio.h>   
void main()  
{  
int n,m,sum,rev;  
clrscr();  
printf("Enter the integer \n");  
scanf("%d",&n);  
m=0;  
rev=0;  
sum=0;  
while (n>0)  
{  
m=n%10;  
sum=sum+m;  
n=n/10;  
rev= rev\*10+m;  
}  
printf("The reversed digit is = %d \n",rev);  
printf("The sum of the digits is = %d \n", sum);  
getch();  
}

**output:**  
  
  
  
  
**/\*C Program to count the number of palindromes in a given list of n numbers\*/**

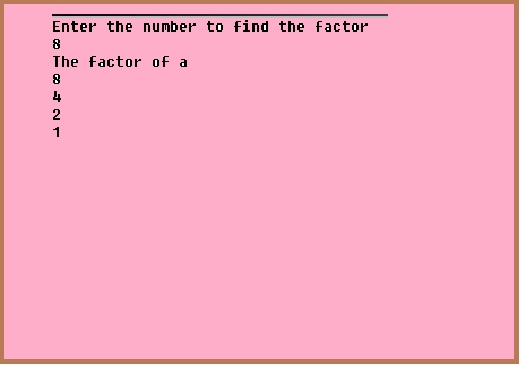
#include<stdio.h>   
#include <conio.h>  
void main()  
{  
int a[50], m, i, md, rev, count, sum, item;  
count = 0;  
clrscr();  
printf("Enter the number of integers \n");  
scanf("%d",&item);  
for (i=0; i<item;i++)  
{  
printf("Enter the number\n");  
scanf("%d",&a[i]);  
}  
for(i =0; i<item;i++)  
{  
m = a[i];  
rev = 0;  
sum = 0;  
while (m>0)  
{  
md = m%10;  
sum =sum+md;  
m = m/10;  
rev=rev\* 10+md;  
}  
if(rev == a[i])  
{  
count=count+1;  
printf("Entered number %d is a palindrome \n",a[i]);  
}  
else  
{  
printf("Entered number %d is a not a palindrome \n",a[i]);  
}  
}  
printf("The number of palindromes in the list of numbers are = %d\n",count);  
getch( );  
}

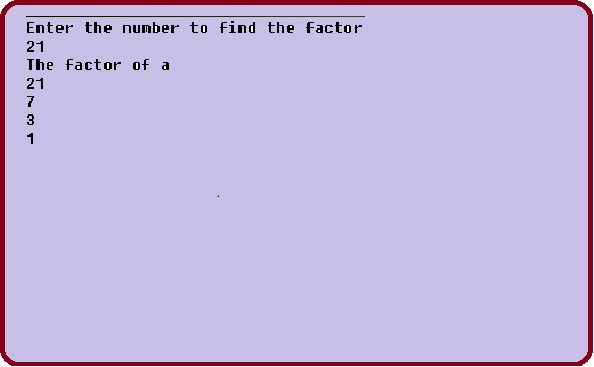
**Output:**



**/\*C Program to find the factors of the given number\*/**

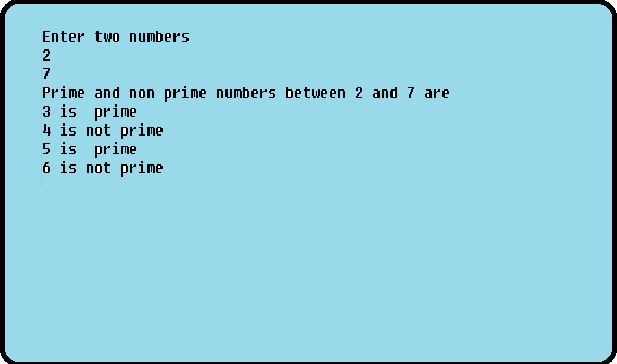
#include <stdio.h>   
#include <conio.h>  
void main()  
{  
int a, b, n;  
clrscr();  
printf("Enter the number to find the factor \n");  
scanf("%d",&n);  
a = n;  
printf("The factor of a\n");  
while (a>0)  
{  
b = n%a;  
if (b == 0)  
{  
printf("%d\n",a);  
}  
a--;  
}  
getch( );  
}

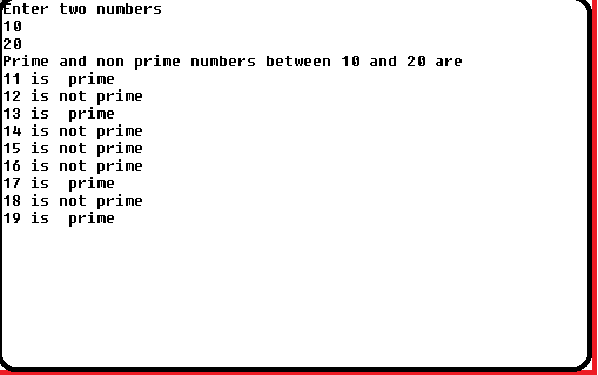
**Output:**  


  
  
  
**/\*C Program to find all the prime numbers between a & b\*/**

#include <stdio.h>  
#include <conio.h>  
void main( )  
{  
int a, b, i, temp, count;  
count = 0;  
clrscr();  
printf("Enter two numbers \n");  
scanf("%d%d", &a, &b);  
printf("Prime and non prime numbers between %d and %d are \n",a,b);  
if(a>b)  
{  
temp=a;  
a=b;  
b=temp;  
}  
a++;  
while(a<b)  
{  
for(i=1; i<=a; i++)  
{  
if(a%i==0)  
count++;  
}  
if(count == 2)  
printf("%d is prime \n",a);  
else  
printf("%d is not prime \n",a);  
a++;  
count=0;  
}  
getch( );  
}

**Output:**

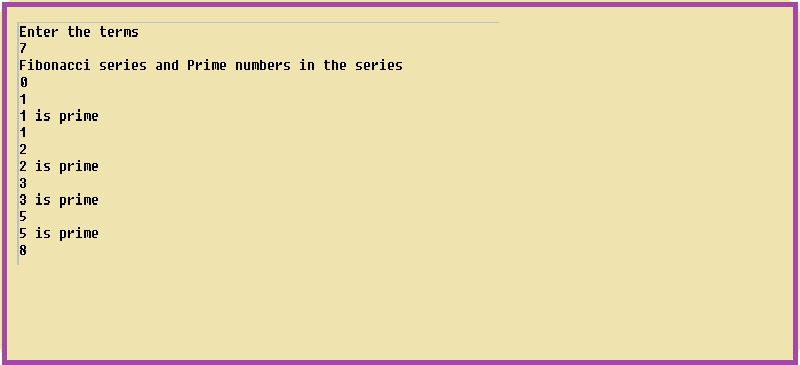


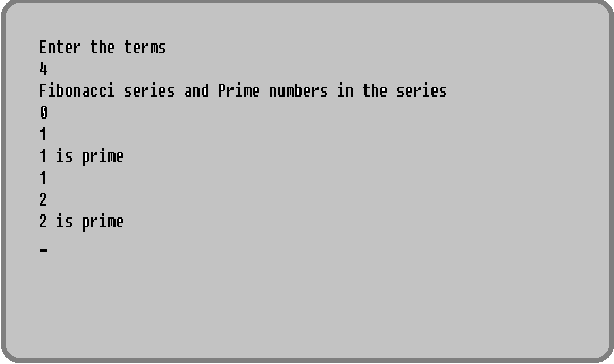


**/\*C Program to generate the fibonacci series and also to find the prime numbers in it\*/**

#include<stdio.h>   
#include <conio.h>  
void main()  
{  
int a, b, c, n, m, i, count;  
count = 0;  
clrscr();  
printf("Enter the terms \n");  
scanf("%d", &n);  
a=0;  
b=1;  
if(n == 0)  
{  
printf("no series \n");  
}  
else  
{  
if( n == 1)  
{  
printf("fibonacci series is \n%d \n",a);  
}  
else  
{  
printf("Fibonacci series and");  
printf(" Prime numbers in the series\n");  
printf("%d\n%d\n", a, b);  
printf("%d is prime\n",b);  
m=2;  
while(m<n)  
{  
c = a+b;  
printf("%d\n",c);  
for (i=1; i<c;i++)  
{  
if(c%i == 0)  
count++;  
}  
if(count == 2|| count == 1)  
printf("%d is prime\n",c);  
a=b;  
b=c;  
m++;  
count=0;  
}  
}  
getch( );

**Output**

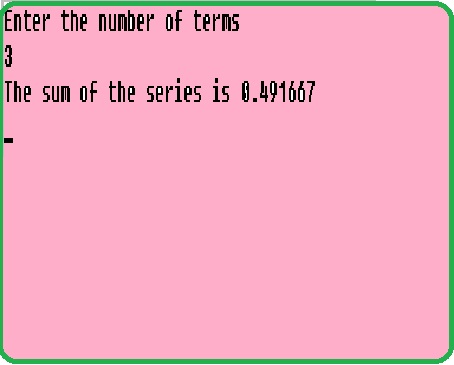


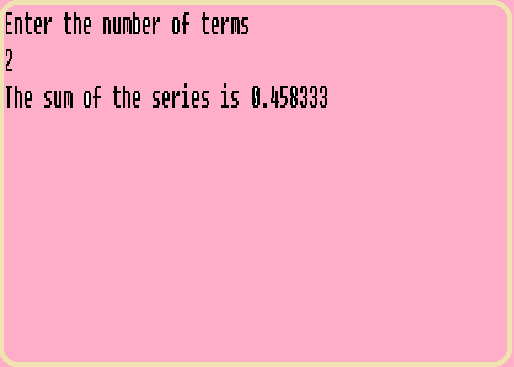


**/\*C Program to calculate the sum of series 2/3!+3/4!+4/5!+......\*/**

#include <stdio.h>  
#include <conio.h>  
void main()  
{  
int j,i,n,m,p;  
float sum, fact;  
sum=0.0;  
clrscr( );  
printf("Enter the number of terms \n");  
scanf("%d", &n);  
if(n == 0)  
printf("Cannot calculate the series \n");  
m=2;  
p=3;  
for(i=1; i<=n; i++)  
{  
fact=1.0;  
for (j=p;j>=1;j--)  
{fact=fact\*j;  
}  
sum=sum+m/fact;  
m++;  
p++;  
}  
printf("The sum of the series is %f\n",sum);  
getch( );  
}

**Output**

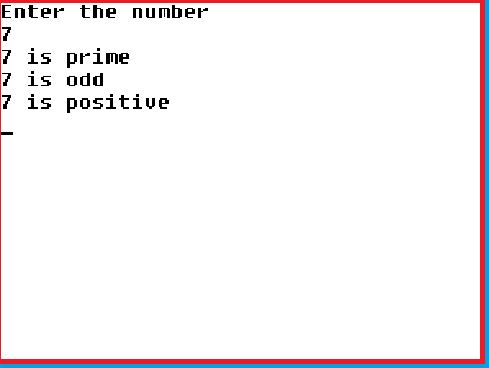


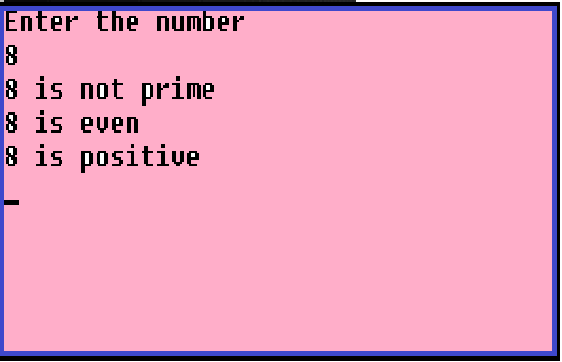


**/\*C Program to check whether a number is odd or even, prime or unprime\*/**

#include <stdio.h>   
#include <conio.h>  
void main( )  
{  
int i, n, r, count;  
count = 0;  
clrscr();  
printf("Enter the number \n");  
scanf("%d", &n);  
for(i=1; i<=n; i++)  
{  
if(n%i == 0)  
count++;  
}  
if (count == 2)  
printf("%d is prime \n",n);  
else  
printf("%d is not prime \n",n);  
r = n%2;  
r ==0? printf("%d is even \n",n): printf("%d is odd \n",n);  
n >0? printf("%d is positive \n",n): printf("%d is negative \n",n);  
getch();  
}

output:





[**C Decision Making & Branching<< Previous**](http://www.lessons2all.com/c_decision_making_branching.php) [**Next >> Array**](http://www.lessons2all.com/c_arrays.php)

Our aim is to provide information to the knowledge seekers.

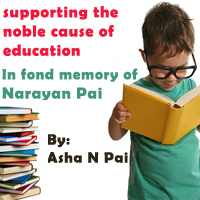
**Support us generously**

 5

 0  456 Google +0  0

[](http://www.s2d6.com/x/?x=c&z=s&v=6895973&k=%5bNETWORKID%5d)

[](http://www.s2d6.com/x/?x=c&z=s&v=6896024&r=%5bRANDOM%5d&k=%5bNETWORKID%5d)



[](http://www.s2d6.com/x/?x=c&z=s&v=6895990&r=%5bRANDOM%5d&k=%5bNETWORKID%5d)

[](http://www.s2d6.com/x/?x=c&z=s&v=6896016&r=%5bRANDOM%5d&k=%5bNETWORKID%5d)

[](http://www.s2d6.com/x/?x=c&z=s&v=6895976&k=%5bNETWORKID%5d)

**About us**

We are here to provide help to the self learners. All data and information provided on this site is for informational purposes only. Please look at Disclaimer too.

[See More](http://www.lessons2all.com/about.php)

**Disclaimer**

All data and information provided on this site is for informational purposes only. The information is provided to help the self learners. http://www.lessons2all.com makes no representations as to accuracy, completeness, suitability, or validity of any information on this site and will not be liable for any errors, omissions, or delays in this information or any losses, injuries, or damages arising from its display or use.

**Follow us**

[http://www.lessons2all.com/images/googleplus-48circle.png](https://plus.google.com/102058705154811980847)

[http://www.lessons2all.com/images/twitter-48circle.png](https://twitter.com/lessons2all)

[http://www.lessons2all.com/images/facebook-48circle.png](https://www.facebook.com/Lessons2allcom)

* [Home](http://www.lessons2all.com/index.php)

- See more at: http://www.lessons2all.com/c\_decision\_making\_looping.php#sthash.sM7bafap.dpuf