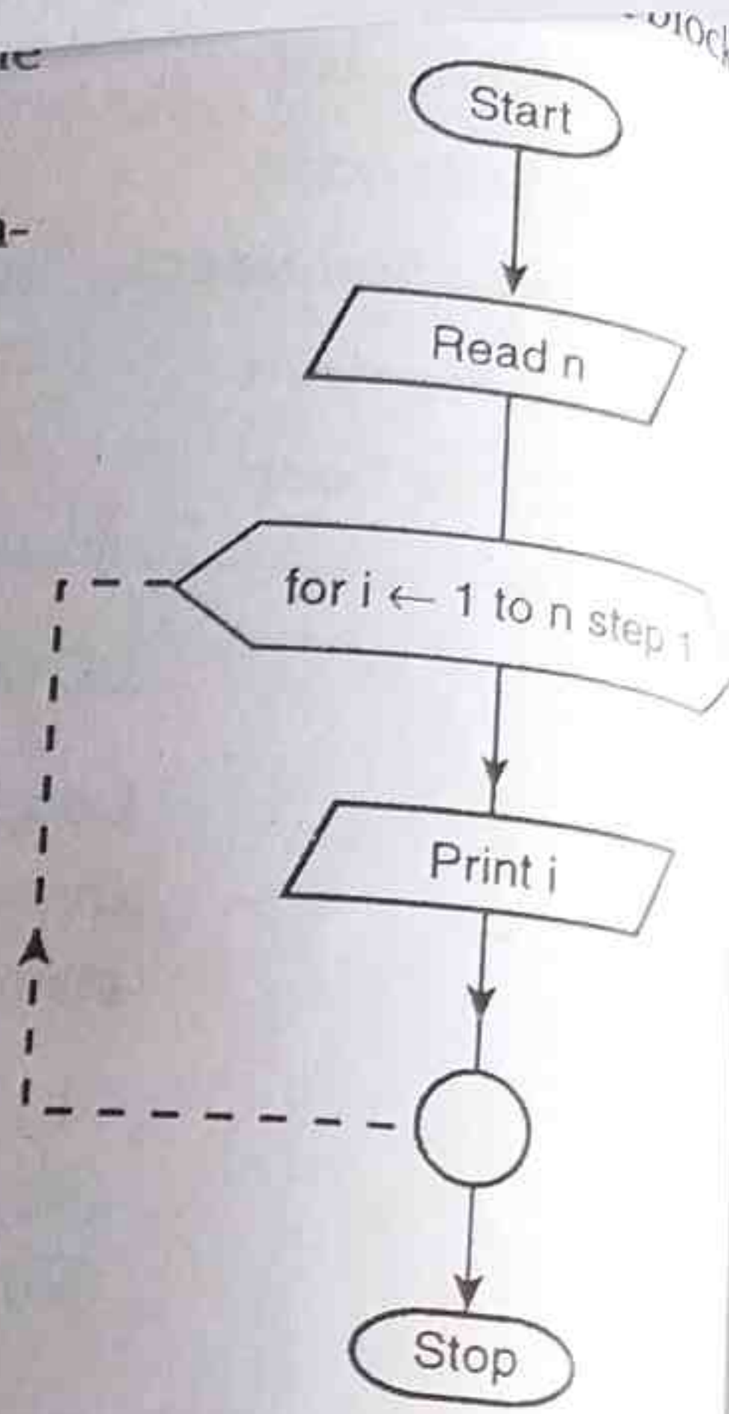


6) The <condition> and <increment> are executed on each iteration/repetition.

Example 1

Write a C program to print natural numbers from 1 to n.

Solution

A `for` loop can be used to generate and print numbers from 1 to n in steps of 1.

```
/* program to natural numbers upto N */
#include <stdio.h>
#include <conio.h>
main()
{   int n,i;
```

```
    clrscr();
    printf("\n Enter value to n : ");
    scanf("%d",&n);
 /* loop to generate and print natural numbers */
    for(i = 1; i <= n; i++)
       printf("%8d",i);
    getch();
}
```

When this program is executed, the user has to enter the value of n. The numbers are generated and printed using the **for** loop as shown below.

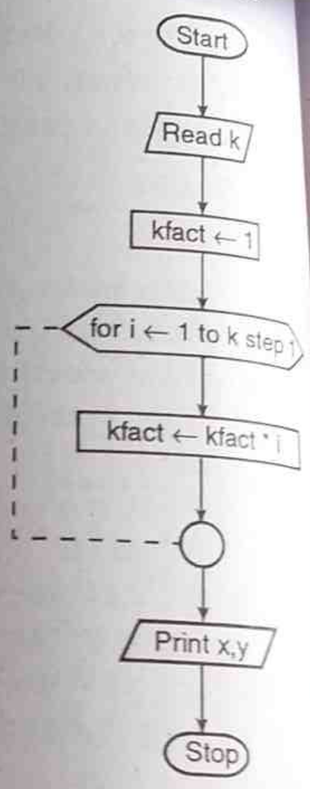
```
Enter value to n : 15
1   2   3   4   5   6   7   8   9   10
11  12  13  14  15
```

Example 3

Write a C program to find factorial of a given integer k.

Solution

We know that k ! = 1 × 2 × 3 × ...× k. The program is written to generate numbers from 1 to k in steps of 1 and multiply them to get the factorial of k.




```
/* program to find factorial of a positive integer */
#include <stdio.h>
#include <conio.h>
main()
{   int k,kfact,i;
    clrscr();
    printf("\n Enter the number : ");
    scanf("%d",&k);
    kfact = 1;
/* loop to generate numbers from 1 to n */
```

```
    for(i = 1; i <= k; i++)
        kfact = kfact*i;
    printf("\n %d factorial is %d",k,kfact);
    getch();
}
```

When this program is executed, the user has to enter a positive integer. A `for` loop is used to generate values of `i` from `1` to `k` in steps of `1`. Each value is multiplied with `kfact` every time to get factorial of given number `k`.

```
Enter the number : 4
4 factorial is 24
```

Example 4

Write a C program to sum the following series.

$$S = 1 + (1+2) + (1+2+3) + \ldots + (1+2+3+\ldots+N)$$

Solution

A nested `for` loop can be written to find the sum of the series. The inner loop is used to find the terms values 1, (1+2), (1+2+3) and so on. These are added in outer loop to get the sum of the series.

```
/* program to find sum of series */
#include <stdio.h>
#include <conio.h>
main()
{   int i,j,n,s,term;
    clrscr();
    printf("\n Enter value to N : ");
    scanf("%d",&n);
    s = 0;
    /* outer loop to find sum of series S */
    for(i = 1; i <= n; i++)
    { /* inner loop to find the terms */
        term = 0;
        for(j = 1; j <= i; j++)
        {
            term = term + j;
        }
        s = s + term;
    }
    printf("\n Sum of the series S = %d",s);
    getch();
}
```

When this program is executed, the user has to enter the value to N. The terms are now generated and added to print the sum of the series as shown below.

```
Enter value to N : 4
Sum of the series S = 20
```

compiler directive `#include <stdlib>` is used when the

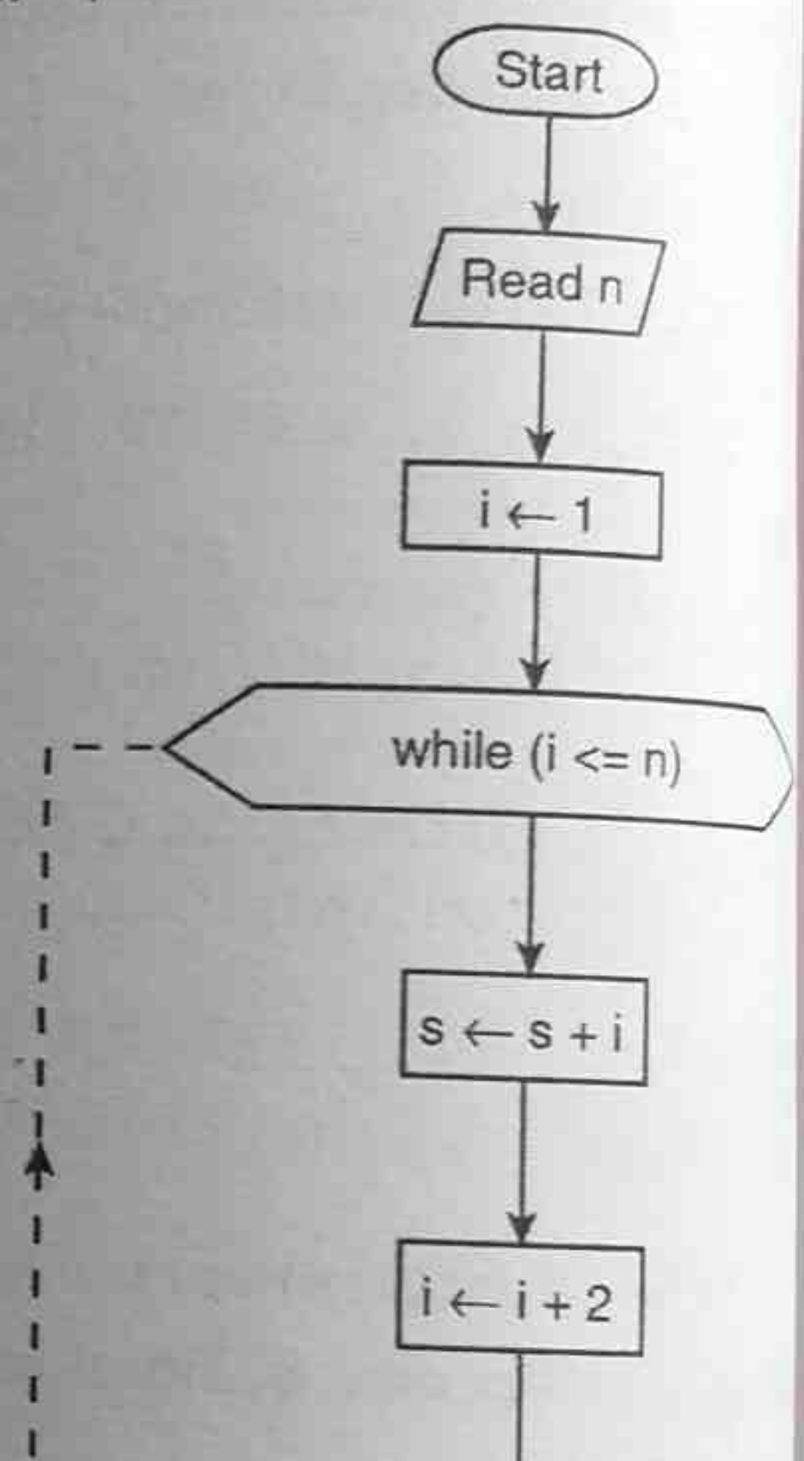
Example 6

Write a C program to find the sum of all odd integers between 1 and n.

Solution

A **while** loop can be used to generate integers from 1 to n in steps of 2 and add them to get the sum of all odd integers. S = 1 + 3 + 5 + 7 + ... + N

```
/* program to find sum of odd integer between 1 and N */
#include <stdio.h>
#include <conio.h>
main()
{   int s,i,n;
    clrscr();
    printf("\n Enter value to N : ");
    scanf("%d",&n);
    s = 0;
    i = 1;
    while(i <= n)
    {
        s = s + i;
        i = i + 2;
    }
    printf("\n Sum of odd integers = %d",s);
    getch();
}
```



When this program is executed, the user has to enter the value of n. The **while** loop used will generate odd numbers and add them to get the sum.

Example 13

Write a C program to check whether the given number n is a prime number.

(A prime number is an integer which is divisible by unity and the number itself. For example, 13 is a prime number because it is divisible by 1 and 13 only.)

Solution

A **for** loop can be used to find the divisor of the given number between 2 and n/2 (for example, when n = 23, it has divisors upto 11 [i.e. 23/2 =11]).

An identification variable `flag` is used to identify any divisors for the given number other than 1. It is assumed `flag = 0` initially; when a divisor is identified it will be changed to `flag = 1`.

Based on the value of `flag`, the loop will be terminated and the prime number is identified.

```
/* program to check prime number */
#include <stdio.h>
#include <conio.h>
main()
{   int n,k,flag=0,r;
    clrscr();
    printf("\n Enter a positive integer : ");
    scanf("%d",&n);
    for(k = 2; k <= n/2 && flag == 0; k++)
    {
        r = n % k;
        if(r == 0)
            flag = 1;
    }
    if(flag == 0)
        printf("\n %d is a prime number",n);
    else
        printf("\n %d is not a prime number",n);
    getch();
}
```

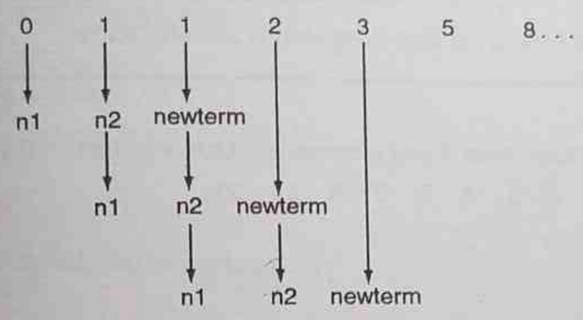
Example 14

Write a C program to generate the Fibonacci series

0 1 1 2 3 5 8...up to n.

Solution

Note that a term in the series is obtained by adding previous two terms. Various terms are generated and printed using the steps given below.



Step 1 : Read n.

Step 2 : Assume `n1=0; n2=1 ;`

Step 3 : Print `n1, n2`

Step 4 : `newterm =n1 + n2 ;`

Step 5 : Print `newterm`

Step 6 : `n1 ← n2`

Step 7 : `n2 ← newterm ;`

Step 8 : Repeat step 4 to step 7 until `newterm > n ;`

```
/* program to generate and print Fibonacci series */
#include <stdio.h>
#include <conio.h>
main()
{   int n,n1,n2,newterm;
    clrscr();
    printf("\n Enter the final term of the series : ");
    scanf("%d",&n);
    n1 = 0;
    n2 = 1;
    printf("%8d%8d",n1,n2);
    newterm = n1 + n2;
    while(newterm <= n)
    {
        printf("%8d",newterm);
        n1 = n2;
        n2 = newterm;
        newterm = n1 + n2;
    }
    getch();
}
```

When this program is executed, the user has to enter the value of n. The

Example 18

Write a C program to evaluate the series

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}$$

Solution

A **for** loop can be used to generate numbers between 1 and n. These numbers are used as denominators of terms to solve the series.

```
/* program to solve a series */
#include <stdio.h>
#include <conio.h>
main()
{  int i,n;
   float s;
   clrscr();
   printf("\n Enter value to N : ");
   scanf("%d",&n);
   s = 0;
/* loop to find summation of the series */
   for(i = 1; i <= n; i++)
      s = s + 1.0/i;
   printf("\n Sum of series = %8.4f",s);
   getch();
}
```

When this program is executed, the user has to enter the end value of the series. The series is evaluated and the result is printed as shown below.

```
Enter value to N : 15
Sum of series = 3.3182
```

43. Discuss the use of **break** and **continue** statements with examples.

SHORT QUESTIONS AND ANSWERS

1) Give differences between **while** and **do-while** statement.

while statement	**do-while** statement
1. The statement block in **while** statement is executed when the values of the condition is **true**.	1. The statement block is executed at least once irrespective of the value of the condition.
2. The condition is evaluated at the beginning of the loop. The statement block is executed if the value of the condition is **true**.	2. The condition is evaluated at the end of the loop. The statement block is executed again if the value of the condition is **true**.

2. Write a C program to find the sum of numbers between 1 and n.

```
#include <stdio.h>
main()
{   int n,i,s = 0 ;
    scanf("%d",&n) ;
    for(i = 1; i <= n ; i++)
            s = s + i ;
    printf("\n sum = %d", s);
}
```

3. Write a C program to find the square root of a given number.

```
#include <stdio.h>
#include <math.h>
main()
{   float x;
    scanf("%f",&x);
    printf("\n square root of %6.2f is %6.2f ",x,sqrt(x));
}
```

4. The loop in a `do...while` structure is executed _____ once.
   at least
5. The _____ statement causes the loop to be terminated.
   break
6. The **`continue`** statement is used to transfer the control to the _____ of the loop.
   beginning
7. Distinguish between the **`break`** and **`continue`** statements in C.

`break` statement	**`continue`** statement
1. A **`break`** statement is used to terminate the execution of a statement block. The next statement which follows this statement block will be executed. The keyword is **`break`**;	1. A **`continue`** statement is used to transfer the control to the beginning of a statement block. The keyword is **`continue`**;
2. When a **`break`** statement is executed in a loop, the repetition of the loop will be terminated.	2. When a **`continue`** statement is executed in a loop, the execution is transferred to the beginning of the loop.

8. What are the differences between **`break`** and **`exit()`** function.

`break` statement	**`exit()`** function
1. A **`break`** statement is used to terminate the execution of a statement block. The next statement which follows this statement block will be executed.	1. An **`exit()`** function is used to terminate the execution of a C program permanently.
2. It is used in a program as **`break;`**	2. It is a built-in function and is used/called with necessary argument as **`exit(0);`**

9. What happens if the condition in a **`while`** loop is initially false?
   When the condition in a **`while`** loop in initially false, the statement block in that loop will not be executed.

10. What is the use of **`break`** statement?
    A **`break`** statement is used to transfer the control to the end of a statement block. The next statement which follows this statement block will be executed. Consider the following example.

```
for(i = 1; i <= n; i++)
{   ------------
```

```
a[2][2] = 1
```

Example 1

Write a C program to find the sum of the given n integers using an array.

Solution

Consider the following steps to read the list of all integers and store them in an array. The integers are then added to get the sum.

Step 1: Read n

Step 2: Loop to read n values of the list.

Step 3: SUM ← 0

Step 4: Loop to add all values to get SUM of those values

Step 5: Print SUM

Step 6: Stop

```
/* Program to find sum of N integers */
#include <stdio.h>
#include <conio.h>
main()
{
    int x[50],n,i,sum;
    clrscr();
    printf("\n How many integers ? ");
    scanf("%d",&n);
    /* loop to read N integers */
    for(i = 0; i < n; i++)
    {
```

```
    printf("\n Enter the %dth value : ",i+1);
    scanf("%d",&x[i]);
  }
  sum = 0;
  /* loop to find sum of all integers */
  for(i = 0; i < n; i++)
     sum = sum + x[i];
  printf("\n Sum of all integers = %d",sum);
  getch();
}
```

Example 2

Write a C program to find the biggest of given n numbers.

Solution

Consider a **for** loop to read the list of n numbers and store them in an array. The first number in the list is assumed as big and is compared with the remaining numbers to get the biggest number. Steps in this program are written as follows.

Step 1 : Read n

Step 2 : Loop to read n numbers of the list

Step 3 : BIG = x[0]

Step 4 : Loop to find the biggest by comparing it with the remaining numbers

Step 5 : Print BIG

Step 6 : Stop

```
/*Program to find biggest of N values*/
#include <stdio.h>
#include <conio.h>
```

```
main()
{   int x[50],n,i,big;
    clrscr();
    printf("\n How many numbers ? ");
    scanf("%d",&n);
    printf("\n Enter all those numbers \n");
    for(i = 0; i < n; i++)
        scanf("%d",&x[i]);
    /* assume first value as big */
    big = x[0];
    /* loop to find the biggest by comparing from second number onwards */
    for(i = 1; i < n; i++)
        if(x[i] > big)
            big = x[i];
    printf("\n %d is the biggest number",big);
    getch();
}
```

When this program is executed, the user has to enter the number of values available and these values are read using **for** loop. The second **for** loop is used to find the biggest value. Finally, the biggest number is displayed as follows.

```
How many numbers ? 5
Enter all those numbers
25  -228  0  185  36
185 is the biggest number
```

Example 3

Write a C program to find the arithmetic mean, variance and standard deviation of any n values.

Solution

An array may be used to store the list of 'n' values. The arithmetic mean, variance and standard deviation are obtained using the following:

$$\text{Mean } \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \text{Variance } \sigma_x = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}$$

$$\text{Standard deviation} = \sqrt{\sigma_x}$$

Following steps are used to find the mean, variance and standard deviation.

Step 1: Read n

Step 2 : Loop to read n values in the list

Step 3 : SUM = 0

Step 4 : Loop to find SUM of all values

Step 5 : $\bar{x} = \frac{SUM}{n}$

Step 6 : VSUM = 0

Step 7 : Loop to find the numerator VSUM to find variance

Step 8 : $\sigma_x = \frac{VSUM}{n}$

Step 9 : Standard deviation = $\sqrt{\sigma_x}$

Step 10 : Print $\bar{x}$, σ_x, standard deviation

Step 11 : Stop

```
/* Program to find Mean,Variance and Standard Deviation */
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{   float x[50],sum,vsum,xbar,sigmax,sd;
    int n,i;
    clrscr();
    printf("\n How many values ? ");
    scanf("%d",&n);
    printf("\n Enter all values in the list \n");
    for(i = 0; i < n; i++)
        scanf("%f",&x[i]);
    sum = 0;
    /* loop to find sum of all values */
    for(i = 0; i < n; i++)
        sum = sum + x[i];
    xbar = sum/n;
    vsum = 0;
    /* loop to find the numerator vsum to find variance */
    for(i = 0; i < n; i++)
        vsum = vsum + (x[i] - xbar)*(x[i] - xbar);
    sigmax = vsum/n;
    sd = sqrt(sigmax);
    printf("\n Arithmetic mean = %0.3f",xbar);
```

```
    printf("\n Variance = %0.3f",sigmax);
    printf("\n Standard deviation = %0.3f",sd);
    getch();
}
```

Example 10

Write a C program to multiply A matrix of order m × n with B matrix of order n × l.