06-06-24

Sum Of n-Integers =>

```
-> main ( )
   {
      int n, s;
      printf ("enter n");
      scanf ("%d", &n);
      s = n * (n+1) / 2;
      printf ("%d", s);
   }
```

I. WITHOUT ARGUMENTS WITHOUT RETURN:-

```
-> Void sum ( )
   {
     int n, s;
     printf ("enter n");
     scanf ("%d", &n);
       s = n *(n+1)/2 ;
     printf ("%d", s);
   }
   main ( )
   {
      sum ( );
   }
```

II. WITH ARGUMENTS WITHOUT RETURN;-

```
-> Void sum ( int n);
   int main ( )
   {
     int n;
     printf ("enter n");
     scanf ("%d", &n);
     sum (n);
     return 0;
   }
   Void sum (int n)
   {
     int s = n *(n+1)/2;
     printf ("%d", s);
   }
```

III. WITHOUT ARGUMENTS WITH RETURN :-

```
-> int sum (void)
   {
     int n, s;
     printf ("enter n: ");
     scanf ("%d", &n);
     s = n * (n+1) / 2;
     return s;
   }
   main ()
   {
     int S;
     S = sum ();
     printf ("%d", s);
   }
```

IV. WITH ARGUMENTS WITH RETURN :-

```
-> int sum (int n)
   {
     return n * (n+1) / 2;
   }
   main ()
   {
     int n, s;
     printf ("enter n: ");
     scanf ("%d", &n);
     s = sum (n);
     printf ("%d", s);
   }
```

$$\frac{n!}{(n-r)! * r!} = {}^{n}C_{r}$$

$${}^{5}C_{2} = \frac{5!}{3! \times 2!} = \frac{120}{12} = 10$$

→ FACTORIAL :-

```
int fact (int n)
{
   int i, f = 1;
   for (i = 1; i <= n; i++)
     f = f * i;
   return f;
}
```

→ (+)

```
# include "fact.c"
main ()
{
   int n, r, ncr;
   printf ("enter n, r");
   scanf ("%d %d", &n, &r);
   ncr = fact(n)/(fact(n-r)*fact(r));
   printf ("%d", ncr);
}
```

Q:- What is a function & what are the advantages:-

FUNCTION:- It is a block of code which performs particular task.

SYNTAX:-

```
returntype name (type arg1, ...)
{
   body
}
```

ADVANTAGES:-

→ Code is reduced
→ Repetative works can be done easy
→ Easy to Debug
→ A complex problem is divided into small parts called Modules / Functions.