★ Functions :-

```
int main ()  // A function
{
    Void RECT ();  // function declaration
    Void CIRCLE();  // function declaration
        RECT ();  // function Calling
        CIRCLE();
        CIRCLE();   > // Calling (User defined data types)
    printf ("In Back in india");
}

Void RECT ()    // function definition
{
    int L, B, A;
    printf ("Enter Length & Breadth : \n");
    Scanf ("%d %d", &L, &B);
    A = L * B;
    printf ("\n Area = %d", A);
}

Void CIRCLE // function definition
{
}
```

Calling functions (Library functions)

User defined functions

Home Work

Q- Write a note on functions and its advantages.

A- A function is a set of statements that take inputs, do some specific computation and produces output.

The idea is to put some commonly or repeatedly done task together and make a function so that instead of writting the same code again and again for different outputs, we can call the function.

Q- Write a program for volume of cylinder without arguements.

A-

```
Void ARC ()
  {
    int r, h, A;
    printf ("Enter the value of r & h : \n");
    Scanf ("%d %d", &r, &h);
    A = 3.14 * r * r * h;
    printf ("Area of Cylinder : %d \n", A);
  }

int main ()
{
    Void ARC ();
        ARC ();
}
```

Q- Write a program for greatest of 2 No.'s without arguments.

A-

```
Void Max 2 ()
{
    int A, B;
    printf ("Enter the value of A and B : \n");
    scanf ("%d %d", &A, &B);
    if (A > B)
      printf ("A is greatest");
    else
      printf ("B is greatest");
}

int main ()
{
    Void Max 2 ();
        Max 2();
}
```

Q- Program for Palindrome of a number without arguments.

A-

```
Void Palindrome ()
{
    int N, R, REV = 0, Prince;
    printf ("Enter the value of N : \n");
    scanf ("%d", &N);
Prince = N;
    while (N != 0)
    {
        R = N % 10;
        REV = REV * 10 + R;
    }   N = N/10;
```

```
    printf (" reverse = %d ", rev);
    if (Prince == rev)
      printf (" Number is Palindrome ");
    else
      printf (" Number is not a Palindrome ");
  }
int main ()
{
    Void Palindrome ();
        Palindrome ();
}
```

Q- Write a program for greatest element of an array without arguements.

A-

```
Void Gm ()
 {
   int N, i ; int max ;
   printf (" Enter the Value of N : \n ");
   Scanf (" %d ", & N);
  int A [N];
  for (i = 0 ; i < N ; i ++)
  {
    printf (" Enter Value at A [%d] : \n ", i);
    Scanf (" %d ", A [i]);
  }
```

```
    Max = A[0];
    for(i=0; i<N; i++)
    {
        if (max < A[i])
            Max = A[i]
    }
    printf (" Greatest element = %d ", max);
}

int main ( )
{
    Void Gm ( );
         Gm ( );
}
```

★ **Functions With Arguements** :- Data sharing among different functions.

With Return

```
int main()
{
    float RECT(int L, float B);  // function declaration
    int len;
    float Brt;
    printf("Enter length and breadth :");
    scanf("%d %f", &len, &Brt);
    float Ar = RECT(len, Brt);  // Calling
    printf("Ar = %d", Ar);
}
```

Actual Arguements

```
float RECT(int L, float B)  // function definition
{
    Ar = L * B;
    return(Ar);
}
```

Formal Arguements (Receiver variables)

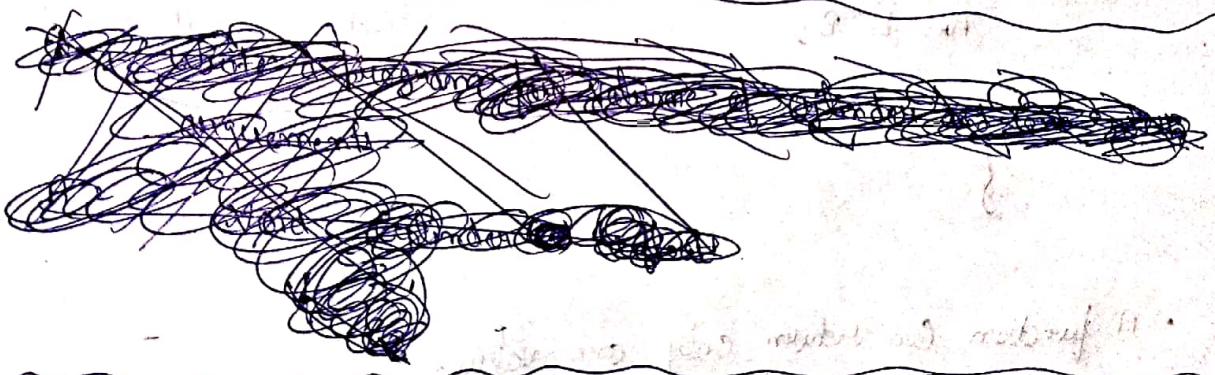
- A function can return only one value.
- Void means → No return.
- If we write void main(), then don't write return 0; at last.

Home Work

Q- What is the use of arguments / Parameters.

A- An Argument is referred to the values that are passed with in a function when the function is called. These values are generally the source of the function that require the Arguments during the process of execution. These values are assigned to the variables in the definition of the function that is called. The Type of values passed in the function is same as that of the variables defined in function definition. These are also called Actual Arguments or Actual Parameters.

Example :- Suppose a Sum ( ) function is needed to be called with two numbers to add. These two numbers are referred to as the Arguments and are passed to Sum ( ) when it Called from some where else.



(i) int fun ( )
{
return (1); → It will Return (1) only
return (2);
}
↓
No

(ii) Void fun ( )
{ printf ("A");
return ;
printf ("B");
}
⇒ No value return, Just Calling function

Q- Write a program for volume of cylinder and cone with arguements.

A-

```
int main ()
{
    float ra, he;
    printf ("Enter value of radius : \n");
    scanf (" %f %f", &ra, &he);
    float Cylinder ( float r, float h);
    float V = Cylinder ( ra, he);
    float Cone ( float r, float h);
    float m = Cone ( ra, he);
    printf (" Volume of cylinder : %f", V);
    printf (" Volume of Cone : %f", m);
}
float Cylinder ( float r, float h)
{
    int V;
    V = 3.14 * r * r * h;
    Return (V);
}

float Cone ( float r, float h)
{
    int m;
    m = (3.14 * r * r * h) / 3;
    return (m);
}
```

Q- Write a program for greatest of 2 No's with Arguements

A-
```
int main ( )
{
    Void N( int A, int B);
    int Aa, Bb;
    printf (" Enter Value of Aa & Bb : \n");
    Scanf (" %d %d", & Aa, & Bb);
        N( Aa, Bb);
}

Void N( int A, int B).
{
    if (A > B)
        printf (" A is greatest");
    else
        printf ("B is greatest");
}
```

Q- Write a program for Reverse of a number using arguements (Pass 'N' as argument)

A -
```
int main ( )
{
    int n;
    Printf (" Enter Value of n : \n");
    Scanf (" %d", & n);
    int Reverse (int N);
    int rev = Reverse (n);
}
    printf (" Reverse = %d", rev);
```

```
int Reverse (int N)
{
    int Rev = 0, r
    { while (N! = 0)
    { r = N % 10;
      Rev = Rev * r + 10;
    } N = N/10;
    } return (Rev);
}
```

Q- Write a program for greatest of 3 No.'s with argument with return.

A-

```
int main ()
{   int a, b, c;
    printf (" Enter value of a,b,c : \n");
    Scanf ("%d %d %d", &a, &b, &c);
    int Greatest (int A, int B, int C)
}int ABC = Greatest (a, b, c);
    Pf ("%d", x);
int Greatest (int A, int B, int C)
{
    if (A > B && A > C)
        return (A);
    else if (B > A && B > C)
        return (B);
    else
    Return (C);
}
```

This is optional but not efficient method.

```
if (A = Greatest (a,b,c))
printf ("%d is greatest", A);
else if (B = Greatest (a,b,c))
printf ("%d is greatest", B);
else if (C = Greatest (a,b,c));
printf ("%d is greatest", C);
```

Q- Write a function: do total() with 3 subject marks as arguments and return total to main(). The pass total from main() to function: do Per(), to calculate percentage and return 'Per' to main() and show on screen.

A-

```
int main()
{
    int M, P, B;
    printf("Enter value of M, P, B: \n");
    Scanf("%d %d %d", &M, &P, &B);
    int do total (int Maths, int Physics, int Biology);
int Total = do total (M, P, B);
    printf("Total marks : %d", Total);
    float doPer (float Maths, float Physics, float Biology);
    float Per = do Per (M, P, B);
    printf("Per = %f", Per);
}
int do total (int Maths, int Physics, int Biology)
{   int Total;
    Total = Maths + Physics + Biology;
    Return (Total);
}
float do Per ( float Maths, float Physics, float Biology)
{
    int main ();
    main ();
float Per = (Total/300) * 100;
Return (Per);
```

Q- Write a function: do total ( ) with 3 subjects marks as Arguements and return total to main ( ). Then pass total from main ( ) to function : do Per ( ), to calculate Percentage and return 'Per' to main ( ) and show on screen.

A -

```
int main ( )
{
    int M, P, B;
    printf ("Enter the value of M, P, B : \n");
    scanf ("%d %d %d", &M, &P, &B);
    int doTotal (int maths, int Physics, int Biology);
    int Tot = doTotal ( M, P, B);
    printf (" Total marks : %d \n", Tot);
    float doPer ( float Tot);
    float Per = doPer (tot);
    printf (" Percentage = %f \n", Per);
}
int do total ( int maths, int Physics, int Biology)
{
    int Total;
    Total = Maths + Physics + Biology;
    return (Total);
}
float do Per ( float tot )
{
    float Per = tot *100/300;
    return (Per);
}
```

★ **Call By value**

In call by value, a copy of values stored in actual arguement variables is passed to formal arguement variables within called function. If any change is made in the values using formal arguement variables, it will not affect the origional values of actual Arguement variables. Eg:-

```
int main()
{
  Void swap (int A, int B);
   int A, B;                         A [2]   B [4]
  printf ("Enter A&B: \n");          1012    2104
  Scanf ("%d %d", &A, &B);
   Swap (A, B);  // Calling
  printf (" Now A=%d B=%d", A, B); // Now
                                     A = 2
                                     B = 4
}
Void Swap (int A, int B)
{                30 [2]     [4] 40
  int temp = A;
     A = B;
     B = temp;
}
```

Call By Referance

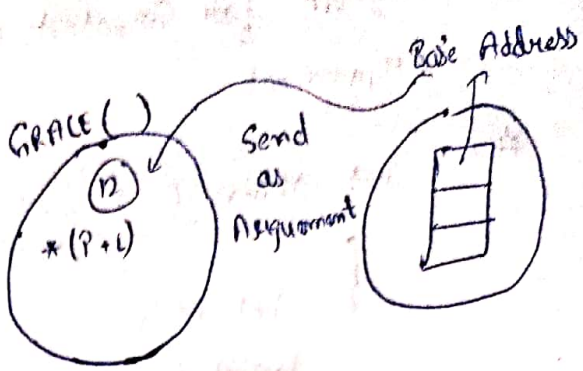
In call By reference, the addresses of actual arguement variables are passed to pointer variables taken as formal Arguements. If any change is made using these pointer, it change the origional values of actual Arguement variables.

```
int main()
{ Void Swap (int *PA, int *PB);
     int A, B;                      A [2]   B [4]
printf (" Enter A&B. \n");
Scanf ("%d %d", &A, &B);
   Swap (&A, &B);
printf (" Now A=%d, B=%d", A, B);
                                   // A = 4
                                      B = 2
}

Void swap (int * PA, int *PB)
{
   int temp = *PA;
       * PA = *PB;
       *PB = temp;
}
```

Passing Array as Argument (An Application of Pointer to array)

```
int main ( )
  {
Void GRACE (int * P, int N)
   int N, i ;
   printf ("Enter N: \n");
Scanf ("%d", &N); // 3
int Marks[N];
for (i=0; i<N; i++)
  {
  printf("Enter marks: \n");
   Scanf ("%d", & Marks[i]);
  }
  GRACE (&Marks[0], N); // Calling
     Marks
    Printf ("\n After grace : \n");
    for (i=0; i<N; i++)
      {
        printf ("%d \n", Marks[i]);
      }
  }
```



Become pointer to Array

```
Void GRACE (int * P, int N)
  { int i;
  for (i=0; i<N; i++)
    {
      if (* (P+i) < 50) // Accessing
          * (P+i) = 50; // Modification
    }
  }
```

Output :-

50
51
50

Input :- Marks

0	40	12 → Base
1	51	16 Address
2	30	20

Home Work

Q- Program for Greatest of Element by passing array as Argument.

A-

```
int main ( )
{
    int N, i
    printf ("Enter value of N: \n");
    Scanf ("%d", & N);
    int A [N];
    for (i=0; i<N; i++)
    {
        printf ("Enter value at A [%d] : \n", i);
        Scanf ("%d", & A [i]);
    }
    int Greatest (int *P, int N);
    int max = Greatest ( & A [0], N);
    printf ("\n output \n");
    printf ("Greatest Element = %d", max);
}
int Greatest (int *P, int N)
{
    int i, max = *P;
    for (i=0; i<N; i++)
    {
        if (max < *(P+i))
        {
            max = *(P+i);
        }
    }
    return (max);
```

Q- Program for Searching an element by passing array as Argumant

A :-

```
int main ( )
{
    int N, i
    printf ("Enter Value of N: \n");
    scanf ("%d", &N);
    int A [N];
    for (i = 0; i < N; i++)
    {
        printf ("Enter the Value at A [%d]: \n", i);
        scanf ("%d", &A[i]);
    }
    Void Searched (int * P, N);
        Searched (&A[0], N);
}
Void Searched (int * P, int N)
{ int i, Wanted;
  printf ("Enter the element want to Search : \n");
  scanf ("%d", &Wanted);
  for (i = 0; i < N; i++)
  { if (Wanted == *(P + i))
      printf ("Element found at %d location\n", i);
  }
}
```

Q-③ :- Write a program for Sorting by passing array as Arguement.

A -

```
int main ()
{
    int i, N
    printf ("Enter the value of N: \n");
    Scanf ("%d", &N);
    int a[N];
    for (i=0; i<N; i++)
    { printf ("Enter Value at a[%d] : \n", i);
      Scanf ("%d", &a[i]);
    }
    Void Sorted (int* P, int N);
        Sorted (&a[0], N);
    for (i=0; i<N; i++)
    { printf ("%d\n", a[i]);
    }
}
Void Sorted (int * P, int N)
{
    int i, j, temp;
    for (i=0 ; i<n-1 ; i++)
    {
        for (j=0 ; j<n-i-1 ; j++)
        {
            if (*(P+j) > *(P+j+1))
            {
                temp = *(P+j);
                *(P+j) = *(P+j+1);
                *(P+j+1) = temp;
            }
        }
    }
}
```

Q- Write a program for simple interest using call by Refarance

A-

```
int main ()
{
    int P, R, T
    printf ("Enter the value of P, R, T : \n");
    scanf (" %d %d %d ", &P, &R, &T);
float Simple interest (int* M, int *N, int *L);
float Prince = Simple interest (&P, &R, &T);
    printf (" Simple Interest = %f ", Prince);
}
float Simple interest (int * M , int *N , int * L)
{
    float S;
    S = (*M) * (*N) * (*L) / 100 ;
    return (S);
```

Q- Write a program for sum of 2 arrays using passing both arrays as arguments.

A-

```
int main()
{
    int n, i;
    printf("Enter the value of n: \n");
    scanf("%d", &n);
    int A[n], B[n];
    for(i=0; i<n; i++)
    {
        printf("Enter the value at A[%d]: \n", i);
        scanf("%d", &a[i]);
    }
    for(i=0; i<n; i++)
    {
        printf("Enter the value at B[%d]: \n", i);
        scanf("%d", &b[i]);
    }
    void add(int *P, int *Q, int n);
        add(&A[0], &B[0], n);
}

void add(int *P, int *Q, int n)
{
    int c, i;
    for(i=0; i<n; i++)
    {
        printf("c[%d] = %d \n", i, *(P+i) + *(Q+i));
    }
}
```

★ STORAGE CLASSES :- Scope and life time of a variable.

1) Local variables
2) Global / External variables
3) Register variables
4) Static variables.

---

(1) Local Variables :-

- Variables declared with in function body are known as local variables.
- It takes garbage value by default.
- It is not accessible in other functions.
- Its scope is limited within function body.

```
int main()
{
    int y; // local variable
    printf("y = %d", y); // garbage
    y = 10;
    void show();
        show(); // calling
}
void show()
{
    printf("%d", y); // Error → Not Accessible
}
```

---

(2) Global / External variables :-

- Variables declared outside function body are known as global variables.
- It takes 0 (zero) value by default.
- It is accessible within all functions in the program.
- It scope is wider than local.

```
int x;
int main()
{
    printf ("\n x = %d", x); // 0
        void show();
          x = 10;
          show();
        printf ("\n x = %d", x); // 5
}
Void show()
    { printf ("%d", x); // 10
          x = x - 5;
    }
```

(3) Register Variables :- Register variables tell the Compiler to store the variable in CPU Register instead of memory. Frequently used variable are kept in Registers and they have faster accessibility. We can never gets the addressof of these variables. "Register" keyword is used to declare the Register variables.

(4) Static Variables :-

```
int main()
{ int i, n;
    printf ("%d", fun());
    printf ("%d", fun());
}
```

```
int fun()
{
    Static int Count = 0;
    Count ++;
    Return (Count);
}
```

output :- 1
2

The value of static variable is not destroyed when the function terminates.

★ Recursion :- When a function calls itself than Recursion takes place.

Your recursive process must be Conditional

// Factorial of N using Recursion.

```
int main()
{
  int FACTORIAL (int N);
  int N, fact;
  printf ("Enter the value of N:\n");
  Scanf ("%d", &N); //3
  fact = FACTORIAL (N); // Calling
  printf ("Factorial : %d", fact); //6
}
int FACTORIAL (int N)
{
    int fact
    if (N == 1)
      return (1);
    else
    fact = N * FACTORIAL (N-1);
      return fact;
}
```

★ Important Question

```
int main()
{
Static int x = 1;
printf ("BCE\n");
x++;
if (x <= 3)
  main();
}
```

Recursion

it will give upto 3 BCE

BCE
BCE
BCE

FACTORIAAL (N 3)

```
if (2 == 1)
else
fact = 3 * FACTORIAL (2);
return (fact);
```

FACTORIAL (N 2)

```
if (2 == 1)
else
fact = 2 * FACTORIAL (1);
return (fact);
```

```
if (1 == 1)
return (1);
```