Passing Array to a Function : (1-D)

→ An entire array can be passed as an argument to a function.

→ To pass an array as an argument to a function, the array-name must be specified, without brackets or subscripts, as an actual argument within the function call.

→ The corresponding formal argument is written in the same manner, though it must be declared as an array within the formal argument declarations. In the formal argument declaration, the array name is written with a pair of empty square brackets. The size of the array is not specified within the formal argument declaration.

→ In function prototypes an empty pair of square brackets must follow the name of each array argument, thus indicating that the argument is an array.

→ If argument declaration names are not included in a function declaration, then an empty pair of square bracket must follow the array argument data type.

→ The program segment illustrates the concept of passing array as an argument to a function.

```
int largest (int a[], int m); /* Function prototype */
void main()
{
   int x[100], i, n, max;
   ----------
   ----------
   max = largest (x, n);      /* Function call */
printf (" Largest = %d", max);
getch();
}
int largest (int a[], int m)  /* Function definition */
    {
    -----
    }
```

→ In the main function, the function largest has been called. This function call contains two actual arguments i.e. the one-dimensional integer array x and an integer variable n.

→ The function definition contains two formal arguments a and m. The formal argument declarations establish a as one-dimensional integer array and m as integer variable.

→ The function prototype can be written without argument name as;

```
int largest (int a[], int);  /* Function prototype */
```

→ The arguments are passed to a function by value when the arguments are ordinary variables. But to pass an array to a function, the value of the array elements are not passed to the function. Rather the array name is interpreted as the address of the first array element (i.e. the address of the memory location containing the first array element). The address is assigned to the corresponding formal argument when the function is called. This formal argument therefore becomes a pointer to the first array element. Arguments that are passed in this manner are said to be <u>passed by reference</u> rather than by value.

<u>Array passing Examples :</u>

```
/* program to input the values into an array and display them */
#include <stdio.h>
#include <conio.h>
void print_array (int a[], int n);
void main()
{
   int a[10], n, i;
   clrscr();
   printf("Enter the no. of elements :");
```

```
    scanf("%d", &n);
    for(i=0; i<n ; i++)
        { printf("Enter the values of an array:");
          scanf("%d", &a[i]);
        }
    print-array(a,n);
    getch();
}
void print-array(int a[], int n)
  {
     int i;
     printf("The array elements are:");
       for(i=0; i<n; i++)
            printf("%d", a[i]);
  }
/* write a program using function to find the addition of
   elements in an array */

#include <stdio.h>
#include <conio.h>
int sum (int a[], int n);
void main()
{
   int a[100], i, n, s;
   clrscr();
   printf("Enter the no. of elements:");
    scanf("%d", &n);
   for(i=0; i<n; i++)
        {
          printf("Enter the elements of an array: \n");
           scanf("%d", &a[i]);
        }
```

```
s = sum(a,n);
printf (" sum of elements of an array is : %d\n", s);
getch();
}
int sum ( int a[], int n)
  {
    int i, s=0;
    for (i=0; i<n; i++)
      {
        printf (" %d\t", a[i]);
        s = s + a[i];
      }
    return s;
}
```

```
/* write a program to reverse the elements of an array */
#include <stdio.h>
#include <conio.h>
void reverse ( int a[], int n);
void main()
{
  int a[80], i, n;
  clrscr();
  printf (" Enter the no. of elements in an array :");
   scanf (" %d", &n);
 printf (" Enter the elements of an array :");
  for (i=0; i<n; i++)
      { scanf (" %d", &a[i]);
      }
  reverse (a, n);
getch();
}
```

```
void reverse ( int a[], int n)
{
   int i, j, temp;
 for ( i=0, j=n-1 ; i<j ; i++, j--)
    {
      temp = a[i] ;
      a[i] = a[j] ;
      a[j] = temp ;
    }
 printf (" The reverse of an array is : ") ;
    for ( i=0 ; i<n ; i++ )
        printf(" %d", a[i]);
 }
```

/* write a program using function to search an element in an array by linear search technique */

```
#include <stdio.h>
#include <conio.h>
void Lsearch ( int a[], int n) ;
void main()
{
   int a[100], i, n;
     clrscr();
    printf (" Enter the no. of elements in an array : ") ;
     scanf (" %d", &n) ;
     printf ( Enter elements of an array.") ;
       for ( i=0; i<n ; i++ )
            scanf (" %d", &a[i]) ;
    Lsearch (a, n) ;
  getch() ;
}
```

```
void maxmin ( int a[], int n)
 {
   int  min , max ;
    min = max = a[0] ;
   for ( i=1; i<n ; i++)
      {
          if ( a[i] < min)
              min = a[i];
         if ( a[i] > max)
              max = a[i] ;
       }
   printf ( " Minimum = %d  Maximum = %d\n" , min, max ) ;
 }
```

/* write a program to sort the elements of an array using function */

```
#include <stdio.h>
# include <conio.h>
void  sort ( int a[], int n);
void main()
 {
   int a[100], n, i ;
   clrscr() ;
   printf (" Enter the no. of elements in an array: ");
    scanf (" %d", &n);
   printf (" Enter the elements of an array: ");
    for( i=0 ; i<n ; i++ )
        scanf ( " %d", &a[i]) ;
   sort ( a, n) ;
  getch() ;
}
```

```
void sort (int a[], int n)
{
    int i, j, temp;
    for (i = 0; i<n-1; i++)
    {
        for (j= i+1; j<n; j++)
        {
            if (a[i] > a[j])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
printf (" The sorted array is :");
    for (i=0; i<n; i++)
        printf (" %d", a[i]);
}
```

/* write a program to search an item in an array using binary search */

```
# include <stdio.h>
# include <conio.h>
```

Passing 2-D Array to a Function :

→ The argument of 2-D array can be passed to a function in the same way 1-D arrays are passed.

→ In 2-D Array, Function prototype is written as ;

function-type function-name (int x[][10], int r);

For example : int diagonal-sum (int x[][10], int r);

```
/* write a program to find the addition of two matrices using
   function */
#include <stdio.h>
#include <conio.h>
void addition ( int a[][10], int r, int c, int b[][10], int p, int q);
void main()
{
  int a[10][10], b[10],[10], r, c, p, q, i, j;
  clrscr();
  printf (" Enter the row & column of matrix a :");
   scanf (" %d %d", &r, &c);
  printf (" Enter the %d values to matrix a :", r*c);
     for ( i=0; i<r; i++ )
         for ( j=0; j<c; j++ )
             scanf (" %d", &a[i][j]);
  printf (" Enter the order of matrix b :");
    scanf (" %d %d", &p, &q);
   printf (" Enter %d values to matrix b :", p*q);
        for ( i=0; i<p; i++ )
            for ( j=0; j<q; j++ )
                scanf (" %d", &b[i][j]);
```

```
printf(" Elements of matrix a :\n");
    for(i=0; i<r; i++)
        {
          for(j=0; j<c; j++)
              {
                printf("%d\t", a[i][j]);
              }
          printf("\n");
 }
printf(" Elements of matrix b: \n");
    for(i=0; i<p; i++)
        {
          for(j=0; j<q; j++)
              {
                printf("%d\t", b[i][j]);
              }
          printf("\n");
    }
addition(a, r, c, b, p, q);
getch();
}
void addition(int a[][10], int r, int c, int b[][10], int p, int q)
{
   int i, j, k, d[10][10];
  if(r==p && c==q)
      {
        for(i=0; i<r; i++)
            {
              for(j=0; j<q; j++)
                  d[i][j] = a[i][j] + b[i][j];
            }
  printf(" The resultant matrix is :\n");
      for(i=0; i<r; i++)
        {
```

```
    for ( j=0 ; j<q ; j++ )
        {  printf ( "%d\t", d[i][j] );
        }
     printf ( "\n" ) ;
} }
 else
     printf ( "Addition is not possible\n") ;
 }
```

/* write a program to find the sum of the principal diagonal of a matrix */

```
#include <stdio.h>
# include <conio.h>
int diagonal-sum ( int a[][10], int p, int q) ;
void main()
{
  int a[10][10] , p, q, i, j, sum ;
  clrscr() ;
  printf (" Enter the row and column of a matrix a:") ;
   scanf ("%d %d", &p, &q);
  printf (" Enter the %d values to matrix a :", p*q);
     for ( i = 0 ; i<p , i++ )
        for ( j = 0 ;  j<q ; j++ )
             scanf ( "%d", & a[i][j] ) ;
    printf ( " Elements of matrix a is : \n") ;
       for ( i=0 ;  i<p ; i++ )
          { for ( j=0 ; j<q ; j++ )
              {
                 printf ("%d\t", a[i][j]) ;
              }
        } printf ("\n" ) ;
```

```
if ( p!=q)
   printf (" Sum of diagonal of the matrix is not possible");
else
    {
       sum = diagonal-sum ( a, p, q);
      printf (" diagonal sum is %d \n", sum);
    }
 getch();
}
int diagonal-sum ( int a[][10], int p, int q)
  {
     int i, j, s = 0;
    for ( i=0; i<p; i++)
      {
        for ( j=0; j<q; j++)
          {
            if ( i==j)
               s = s + a[i][j];
          }
       return s;
  }
```

/* write a program using function to find the multiplication of two matrices */

```
#include <stdio.h>
#include <conio.h>
void multiplication( int [][], int, int, int[][], int, int);
void main()
{
   int a[10][10], b[10][10], c[10][10];
   int i, j, p, q, m, n;
   clrscr();
   printf("Enter the order of matrix a :");
    scanf ("%d %d", &m, &n);
```

```
printf("Enter the order of matrix b:");
scanf("%d %d", &p, &q);
if (n != p)
    printf("Addit Multiplication is not possible");
else
    {
      printf("Enter the elements of matrix a:");
        for(i=0; i<m; i++)
          {
            for(j=0; j<n; j++)
              scanf("%d", &a[i][j]);
          }
printf("Enter the elements of matrix b:");
    for(i=0; i<p; i++)
      {
        for(j=0; j<q; j++)
          scanf("%d", &b[i][j]);
      }
 printf("Elements of matrix a is:");
   for(i=0; i<m; i++)
     {
       for(j=0; j<n; j++)
         {
           printf("%d\t", a[i][j]);
         }
       printf("\n");
     }
printf("Elements of matrix b is:");
     for(i=0; i<p; i++)
       {
         for(j=0; j<q; j++)
           {
             printf("%d\t", b[i][j]);
           }
       printf("\n");
     }
```

```
    multiplication (a, m, n, b, p, q);
    getch();
}
}
void multiplication ( int a[][10], int m, int n, int b[][10], int p, int q)
{
    int i, j, k, c[10][10];
    printf(" The resultant matrix is :");
    for ( i=0; i< m ; i++ )
        {
            for ( j=0; j<q ; j++)
                {
                    c[i][j] = 0 ;
                for (k=0 ; k<p; k++ )
                c[i][j] = ( a[i][k] * b[k][j]) + c[i][j] ;
        }
    }
    for (i=0; i< m; i++ )
        {
            for ( j=0; j<q ; j++)
                { printf (" %d", c[i][j]);
                }
            printf(" \n");
        }
}
```