Chapter - 6 : Functions

Pass arrays to a function in C

If you want to pass a single-dimension array as an argument in a function, you would have to declare a formal parameter in one of following three ways and all three declaration methods produce similar results because each tells the compiler that an integer pointer is going to be received.

Method-1 Formal parameters as a pointer −

```
void myFunction (int *param ) {
  .
  .
}
```

Method-2 **Formal parameters as a sized array −**

```
void myFunction( int param[10] ) {
  .
  .
}
```

Method-3 **Formal parameters as an unsized array −**

```
void myFunction(int param[ ]) {
  .
  .
}
```

Example - A function, which takes an array as an argument along with another argument and based on the passed arguments, it returns the average of the numbers passed through the array.

```
#include <stdio.h>

/* function declaration */
double getAverage(int arr[], int size);
```

```
void main () {

   /* an int array with 5 elements */
   int balance[5] = {1000, 2, 3, 17, 50};
   double avg;

   /* pass pointer to the array as an argument */
   avg = getAverage( balance, 5 ) ;

   /* output the returned value */
   printf( "Average value is: %f ", avg );

}

double getAverage(int arr[ ], int size) {

   int i;
   double avg;
   double sum = 0;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }
   avg = sum / size;

   return avg;
}
```

Example 2: Passing arrays to functions

```
// Program to calculate the sum of array elements by passing to a function

#include <stdio.h>
float calculateSum(float age[]);

void  main() {
    float result, age[] = {23.4, 55, 22.6, 3, 40.5, 18};

    // age array is passed to calculateSum()
    result = calculateSum(age);
    printf("Result = %.2f", result);
}

float calculateSum(float age[]) {

  float sum = 0.0;

  for (int i = 0; i < 6; ++i) {
                    sum += age[i];
  }

  return sum;
}
```

Passing Multidimensional Arrays to a Function

To pass multidimensional arrays to a function, only the name of the array is passed to the function(similar to one-dimensional arrays).

Example : Passing two-dimensional arrays

```
#include <stdio.h>
void displayNumbers(int num[2][2]);
int main()
{
   int num[2][2];
   printf("Enter 4 numbers:\n");
   for (int i = 0; i < 2; ++i)
      for (int j = 0; j < 2; ++j)
         scanf("%d", &num[i][j]);

   // passing multi-dimensional array to a function
   displayNumbers(num);
   return 0;
}

void displayNumbers(int num[2][2])
{
   printf("Displaying:\n");
   for (int i = 0; i < 2; ++i) {
      for (int j = 0; j < 2; ++j) {
         printf("%d\n", num[i][j]);
      }
   }
}
```

Difference between passing an array and passing single value data to a function

- There are two ways to pass a single value variable to a function, by value and by reference.
- If **passed by value**, a copy is of the variable is made and passed to the function. If such function modifies the value, it only modifies a copy; the caller remains with the unchanged value.
- If **passed by reference**, a copy of the address is passed. The program may write something to that address. The caller will have the value changed.
- Arrays in C are always passed by reference. The compiler thinks of an array as a starting address and a length, and by how much it should increment the pointer to get to the next array element.

```
// Program to pass the array elements to a function

#include <stdio.h>
void modify(float age[]);

void  main()
{
   float age[] = {23, 55, 22, 3, 40, 18};

   modify(age);

   for (int i = 0; i < 6; ++i) {
      printf(" %f", age[i]);
}

void modify(float age[ ])
{
  for (int i = 0; i < 6; ++i)
          age[i]= age[i] + 5;

}
```

Output

```
28 60 27 8 45 23
```