Passing arrays to user defined functions

In C, there are various general problems which requires passing more than one variable of the same type to a function. For example, consider a function which sorts the 10 elements in ascending order. Such a function requires 10 numbers to be passed as the actual parameters from the main function. Here, instead of declaring 10 different numbers and then passing into the function, we can declare and initialize an array and pass that into the function. This will resolve all the complexity since the function will now work for any number of values.

One-Dimensional Arrays

- Like the values of simple variables, it is also possible to pass the value of an array to a function.
- To pass a one-dimensional array to a called function, it is sufficient to list the name of the array, without any subscripts, and the size of the array as arguments.

Example

largest(a,n);

- Will pass the whole array a to the called function.
- The called function expecting this call must be appropriately defined.
- The largest function header might look like:

float largest(float array[],int size)

- The function largest is defined to take two arguments, the array name and the size of the array to specify the number of elements in the array.
- The declaration of the formal argument

array is

float array[];

- The pair of brackets informs the compiler that the argument ***array*** is an array of numbers.
- It is not necessary to specify the size of the ***array*** here.

Methods to declare a function that receives an array as an argument

There are 3 ways to declare the function which is intended to receive an array as an argument.

First way:

1. return_type function(type arrayname[])

Declaring blank subscript notation [] is the widely used technique.

Second way:

1. return_type function(type arrayname[SIZE])

Optionally, we can define size in subscript notation [].

Third way:

1. return_type function(type *arrayname)