Passing Array to Function in C

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.

As we know that the array_name contains the address of the first element. Here, we must notice that we need to pass only the name of the array in the function which is intended to accept an array. The array defined as the formal parameter will automatically refer to the array specified by the array name defined as an actual parameter.

Consider the following syntax to pass an array to the function.

1. functionname(arrayname);//passing array

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)

You can also use the concept of a pointer. In pointer chapter, we will learn about it.

C language passing an array to function example

```
#include<stdio.h>
int minarray(int arr[],int size){
int min=arr[0];
int i=0;
for(i=1;i<size;i++){
if(min>arr[i]){
min=arr[i];
}
}//end of for
return min;
}//end of function

int main(){
int i=0,min=0;
int numbers[]={4,5,7,3,8,9};//declaration of array

min=minarray(numbers,6);//passing array with size
printf("minimum number is %d \n",min);
return 0;
}
```

Output

```
minimum number is 3
```

C function to sort the array

```
#include<stdio.h>
void Bubble_Sort(int[]);
void main ()
```

```
4.  {
5.      int arr[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
6.      Bubble_Sort(arr);
7.  }
8.  void Bubble_Sort(int a[]) //array a[] points to arr.
9.  {
10. int i, j,temp;
11.     for(i = 0; i<10; i++)
12.     {
13.         for(j = i+1; j<10; j++)
14.         {
15.             if(a[j] < a[i])
16.             {
17.                 temp = a[i];
18.                 a[i] = a[j];
19.                 a[j] = temp;
20.             }
21.         }
22.     }
23.     printf("Printing Sorted Element List ...\n");
24.     for(i = 0; i<10; i++)
25.     {
26.         printf("%d\n",a[i]);
27.     }
28. }
```

Output

```
Printing Sorted Element List ...
7
9
10
12
23
23
34
44
78
101
```

Returning array from the function

As we know that, a function can not return more than one value. However, if we try to write the return statement as return a, b, c; to return three values (a,b,c), the function will return the last mentioned value which is c in our case. In some problems, we may need to return multiple values from a function. In such cases, an array is returned from the function.

Returning an array is similar to passing the array into the function. The name of the array is returned from the function. To make a function returning an array, the following syntax is used.

```
int * Function_name() {
//some statements;
return array_type;
}
```

To store the array returned from the function, we can define a pointer which points to that array. We can traverse the array by increasing that pointer since pointer initially points to the base address of the array. Consider the following example that contains a function returning the sorted array.

```
#include<stdio.h>
int* Bubble_Sort(int[]);
void main ()
{
    int arr[10] = { 10, 9, 7, 101, 23, 44, 12, 78, 34, 23};
    int *p = Bubble_Sort(arr), i;
    printf("printing sorted elements ...\n");
    for(i=0;i<10;i++)
    {
        printf("%d\n",*(p+i));
    }
}
int* Bubble_Sort(int a[]) //array a[] points to arr.
{
int i, j,temp;
```

```
16.    for(i = 0; i<10; i++)
17.    {
18.        for(j = i+1; j<10; j++)
19.        {
20.            if(a[j] < a[i])
21.            {
22.                temp = a[i];
23.                a[i] = a[j];
24.                a[j] = temp;
25.            }
26.        }
27.    }
28.    return a;
29. }
```

Output

```
Printing Sorted Element List ...
7
9
10
12
23
23
34
44
78
101
```