Passing Arrays as Function Arguments in C

If you want to pass an array to a function, you can use either call by value or call by reference method. In call by value method, the argument to the function should be an initialized array, or an array of fixed size equal to the size of the array to be passed. In call by reference method, the function argument is a pointer to the array.

Pass array with call by value method

In the following code, the main() function has an array of integers. A user−defined function average () is called by passing the array to it. The average() function receives the array, and adds its elements using a for loop. It returns a float value representing the average of numbers in the array.

Example

</> Open Compiler

```
#include <stdio.h>
float average(int arr[5]);
int main(){
   int arr[] = {10, 34, 21, 78, 5};
   float avg = average(arr);
   printf("average: %f", avg);
}
float average(int arr[5]){
   int sum=0;
   int i;
   for (i=0; i<5; i++){
      printf("arr[%d]: %d\n", i, arr[i]);
      sum+=arr[i];
   }
   return (float)sum/5;
}
```

Output

```
arr[0]: 10
arr[1]: 34
```

```
arr[2]: 21
arr[3]: 78
arr[4]: 5
average: 29.600000
```

In the following variation, the average() function is defined with two arguments, an uninitialized array without any size specified. The length of the array declared in main() function is obtained by divising the size of the array with the size of int data type.

Example

</> Open Compiler

```
#include <stdio.h>
float average(int arr[], int length);
int main(){
   int arr[] = {10, 34, 21, 78, 5};
   int length = sizeof(arr)/sizeof(int);
   float avg = average(arr, length);
   printf("average: %f", avg);
}
float average(int arr[], int length){
   int sum=0;
   int i;
   for (i=0; i<length; i++){
      printf("arr[%d]: %d\n", i, arr[i]);
      sum+=arr[i];
   }
   return (float)sum/length;
}
```

Output

```
arr[0]: 10
arr[1]: 34
arr[2]: 21
arr[3]: 78
arr[4]: 5
average: 29.600000
```

Pass array with call by reference

To use this approach, we should understand that elements in an array are of similar data type, stored in continuous memory locations, and the array size depends on the data type. Also, the address of the 0th element is the pointer to the array.

In the following example –

```
int a[5] = {1,2,3,4,5};
```

The size of the array is 20 bytes (4 bytes for each int)

```
Int *x = a;
```

Here x is the pointer to the array. It points to the 0th element. If the pointer is incremented by 1, it points to the next element.

Example

</> Open Compiler

```
#include <stdio.h>
int main() {
   int a[] = {1,2,3,4,5};
   int *x = a, i;
   for (i=0; i<5; i++){
      printf("%d\n", *x);
      x++;
   }
   return 0;
}
```

Output

```
1
2
3
```

```
4
5
```

Let us use this characteristics for passing the array by reference. In the main() function, we declare an array and pass its address to the max() function. The max() function traverses the array using the pointer and returns the largest number in the array, back to main() function.

Example

</> Open Compiler

```
#include <stdio.h>
int max(int *arr, int length);
int main(){
   int arr[] = {10, 34, 21, 78, 5};
   int length = sizeof(arr)/sizeof(int);
   int maxnum = max(arr, length);
   printf("max: %d", maxnum);
}
int max(int *arr, int length){
   int max=*arr;
   int i;
   for (i=0; i<length; i++){
      printf("arr[%d]: %d\n", i, (*arr));
      if ((*arr)>max)
         max = (*arr);
      arr++;
   }
   return max;
}
```

Output

```
arr[0]: 10
arr[1]: 34
arr[2]: 21
arr[3]: 78
```

```
arr[4]: 5
max: 78
```

The max() function receives the address of the array from main() in the pointer arr. Each time, when it is incremented, it points to the next element in the original array.

The max() function can also access the array elements as a normal subscripted array as in the following definition –

```
int max(int *arr, int length){
   int max=*arr;
   int i;
   for (i=0; i<length; i++){
      printf("arr[%d]: %d\n", i, arr[i]);
      if (arr[i]>max)
         max = arr[i];
   }
   return max;
}
```


Pass two−dimensional array to function

You can also pass the pointer of a two-dimensional array to a function. Inside the function, the two dimensional array is traversed with a nested for loop construct

Example

</> Open Compiler

```
#include <stdio.h>
int twoDarr(int *arr);
int main(){
   int arr[][3]= {10, 34, 21, 78, 5, 25};
   twoDarr(*arr);
}
int twoDarr(int *arr){
   int max=*arr;
   int i, j;
   for (i=0; i<2; i++){
```

```
      for (j=0; j<3; j++){
         printf("%d\t", arr[i]);
         arr++;
      }
      printf("\n");
   }
}
```

Output

```
10    34    21
5     25    16
```

Function to compare string lengths

In the following program, two strings are passed to compare() functions. In C, as string is an array of char data type. We use strlen() function to find the length of string which is the number of characters in it.

Example

</> Open Compiler

```
#include <stdio.h>
#include <string.h>
int compare( char *, char *);
int main() {
   char  a[] = "BAT";
   char  b[] = "BALL";
   int ret = compare(a, b);
   return 0;
}
int compare (char *x, char *y){
   int val;
   if (strlen(x)>strlen(y)){
      printf("length of string a is greater than or equal to length of string
b");
   }
   else{
      printf("length of string a is less than length of string b");
```

```
   }
}
```

Output

```
length of string a is less than length of string b
```