Move Zeros to End

Given an array of random numbers, Push all the zero's of a given array to the end of the array. For example, if the given arrays is {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0}, it should be changed to {1, 9, 8, 4, 2, 7, 6, 0, 0, 0, 0}. The order of all other elements should be same. Expected time complexity is O(n) and extra space is O(1).

Example:

```
Input :  arr[] = {1, 2, 0, 4, 3, 0, 5, 0};
Output : arr[] = {1, 2, 4, 3, 5, 0, 0, 0};

Input : arr[]  = {1, 2, 0, 0, 0, 3, 6};
Output : arr[] = {1, 2, 3, 6, 0, 0, 0};
```

There can be many ways to solve this problem. Following is a simple and interesting way to solve this problem.

Traverse the given array 'arr' from left to right. While traversing, maintain count of non-zero elements in array. Let the count be 'count'. For every non-zero element arr[i], put the element at 'arr[count]' and increment 'count'. After complete traversal, all non-zero elements have already been shifted to front end and 'count' is set as index of first 0. Now all we need to do is that run a loop which makes all elements zero from 'count' till end of the array.

Below is the implementation of the above approach.

```
// A C program to move all zeroes at the end of array
#include <stdio.h>
// Function which pushes all zeros to end of an array.
void pushZerosToEnd(int arr[], int n)
{
    int count = {0};  // Count of non-zero elements
    // Traverse the array. If element encountered is non-
    // zero, then replace the element at index 'count'
    // with this element
    for (int i = 0; i < n; i++)
        if (arr[i] != 0)
            arr[count++] = arr[i]; // here count is
        arr[count++] = 0;
```

```
}

// Driver program to test above function
int main()
{
    int arr[] = {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9};
    int n = sizeof(arr) / sizeof(arr[0]);
    pushZerosToEnd(arr, n);
    printf("%s\n", "Array after pushing all zeros to end of a
    for (int i = 0; i < n; i++)
      printf("%d ", arr[i]);
    return 0;
}
```

Output

```
Array after pushing all zeros to end of array :
1 9 8 4 2 7 6 9 0 0 0 0
```

Time Complexity: O(n) where n is number of elements in input array.

Auxiliary Space: O(1)