Rearrange array in alternating positive & negative items with O(1) extra space

Given an array of positive and negative numbers, arrange them in an alternate fashion such that every positive number is followed by negative and vice-versa maintaining the order of appearance.
Number of positive and negative numbers need not be equal. If there are more positive numbers they appear at the end of the array. If there are more negative numbers, they too appear in the end of the array.

Example:

```
Input:  arr[] = {1, 2, 3, -4, -1, 4}
Output: arr[] = {-4, 1, -1, 2, 3, 4}

Input:  arr[] = {-5, -2, 5, 2, 4, 7, 1, 8, 0, -8}
output: arr[] = {-5, 5, -2, 2, -8, 4, 7, 1, 8, 0}
```

This question has been asked at many places (See this and this)

The above problem can be easily solved if O(n) extra space is allowed. It becomes interesting due to the limitations that O(1) extra space and order of appearances.
The idea is to process array from left to right. While processing, find the first out of place element in the remaining unprocessed array. An element is out of place if it is negative and at odd index, or it is positive and at even index. Once we find an out of place element, we find the first element after it with opposite sign. We right rotate the subarray between these two elements (including these two).

Following is C++ implementation of above idea.

```
/*  C++ program to rearrange positive and negative integers in alternate
    fashion while keeping the order of positive and negative numbers. */
#include <iostream>
#include <assert.h>
using namespace std;

// Utility function to right rotate all elements between [outofplace, cur]
void rightrotate(int arr[], int n, int outofplace, int cur)
{
    char tmp = arr[cur];
    for (int i = cur; i > outofplace; i--)
```

```
        arr[i] = arr[i-1];
    arr[outofplace] = tmp;
}

void rearrange(int arr[], int n)
{
    int outofplace = -1;

    for (int index = 0; index < n; index ++)
    {
        if (outofplace >= 0)
        {
            // find the item which must be moved into the out-of-place
            // entry if out-of-place entry is positive and current
            // entry is negative OR if out-of-place entry is negative
            // and current entry is negative then right rotate
            //
            // [...-3, -4, -5, 6...] -->   [...6, -3, -4, -5...]
            //      ^                          ^
            //      |                          |
            //     outofplace      -->      outofplace
            //
            if (((arr[index] >= 0) && (arr[outofplace] < 0))
                || ((arr[index] < 0) && (arr[outofplace] >= 0)))
            {
                rightrotate(arr, n, outofplace, index);

                // the new out-of-place entry is now 2 steps ahead
                if (index - outofplace > 2)
                    outofplace = outofplace + 2;
                else
                    outofplace = -1;
            }
        }


        // if no entry has been flagged out-of-place
        if (outofplace == -1)
        {
            // check if current entry is out-of-place
            if (((arr[index] >= 0) && (!(index & 0x01)))
                || ((arr[index] < 0) && (index & 0x01)))
            {
                outofplace = index;
            }
        }
    }
}

// A utility function to print an array 'arr[]' of size 'n'
```

```
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
      cout << arr[i] << " ";
    cout << endl;
}

// Driver program to test abive function
int main()
{
    //int arr[n] = {-5, 3, 4, 5, -6, -2, 8, 9, -1, -4};
    //int arr[] = {-5, -3, -4, -5, -6, 2 , 8, 9, 1 , 4};
    //int arr[] = {5, 3, 4, 2, 1, -2 , -8, -9, -1 , -4};
    //int arr[] = {-5, 3, -4, -7, -1, -2 , -8, -9, 1 , -4};
    int arr[] = {-5, -2, 5, 2, 4, 7, 1, 8, 0, -8};
    int n = sizeof(arr)/sizeof(arr[0]);

    cout << "Given array is \n";
    printArray(arr, n);

    rearrange(arr, n);

    cout << "Rearranged array is \n";
    printArray(arr, n);

    return 0;
}
```

Output:

```
Given array is
-5 -2 5 2 4 7 1 8 0 -8
Rearranged array is
-5 5 -2 2 -8 4 7 1 8 0
```

This article is contributed by **Sandeep Joshi**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

111 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

4 Average Difficulty : **4/5.0**
Based on **3** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.