Array Traversal in C++

Given an integer array of size **N**, the task is to traverse and print the elements in the array.

Examples:

> Input: arr[] = {2, -1, 5, 6, 0, -3}
>
> ***Output:*** *2 -1 5 6 0 -3*
>
> ***Input:*** *arr[] = {4, 0, -2, -9, -7, 1}*
>
> ***Output:*** *4 0 -2 -9 -7 1*

There are three ways to traverse the elements of an array in C++:

1. Using for loop.
2. Using for_each loop.
3. using range-based for loop.

Let’s start discussing each of these methods in detail.

1. Using for Loop

Below is the approach for traversing an array using the for loop.

Approach:

A. Start a loop from **0** to **N-1**, where **N** is the size of the array.

```
for(i = 0; i < N; i++)
```

B. Access every element of the array with the help of

```
arr[index]
```

C. Print the elements.

```
cout << arr[i] << endl;
```

Below is the implementation of the above approach:

```
// C++ program to traverse
// the array
#include <bits/stdc++.h>
using namespace std;

// Function to traverse and
// print the array
void printArray(int* arr, int n)
{
    int i;

    cout << "Array: ";
    for (i = 0; i < n; i++)
    {
        cout << arr[i] << " ";
    }
}

// Driver code
int main()
{
    int arr[] = {2, -1, 5, 6, 0, -3};
    int n = sizeof(arr) / sizeof(arr[0]);

    printArray(arr, n);
    return 0;
}
```

Output

```
Array: 2 -1 5 6 0 -3
```

2.Using a for-each loop

for_each is a powerful STL algorithm to operate on range elements and apply custom-defined functions. It takes range starting and the last iterator objects as the first two parameters and the function object as the third one.

Below is the C++ program to implement the above approach:

```
// C++ program to traverse the
// array using for_each loop
#include <bits/stdc++.h>
#include <iostream>
using namespace std;

// Driver code
int main()
{
    int arr[] = {2, -1, 5, 6, 0, -3};

    // Traverse array with for_each
    // using array's data type
    cout << "Traverse using array's data type";
    for(int x : arr)
    cout << x << " ";
    cout << endl;

    // Traverse array with for_each
    // using auto keyword
    cout << "Traverse using auto keyword";
    for(auto x : arr)
    cout << x << " ";
    return 0;
}
```

Output

```
Traverse using array's data type2 -1 5 6 0 -3
Traverse using auto keyword2 -1 5 6 0 -3
```

3. Using range-based Loop

The range-based loop is the readable version of the for loop. The following code shows how to implement the above code using a range-based loop.

```
// C++ program to traverse the
// array using range-based loop
#include <bits/stdc++.h>
```

```
#include <iostream>
using namespace std;

// Driver code
int main()
{
    int arr[] = {2, -1, 5, 6, 0, -3};

    for (const auto &var : arr)
    {
    cout << var << " " ;
    }
    return 0;
}
```

Output

```
2 -1 5 6 0 -3
```