# Reference to Array in C++

Difficulty Level : **Medium** • Last Updated : 29 Jun, 2021

Reference to an array means aliasing an array while retaining its identity. Reference to an array will not be an int* but an int[]. Let us discuss this in detail by discussing the difference between these two. This is quite weird that **int[]** is the same as **int*** but still compiler perspective on both is entirely different. The major two differences are as follows:

1. **int[]** for the compiler is an array, so it provides an iterator for this (that's why we have For-Each Loop) but **int*** is just a pointer to an integer. This could just be a pointer to integer or pointer to the beginning of integer array which totally depends upon our perspective. Hence, no For-Each loop in this case.
2. For compiler, a and b are the same data type i.e int* here for compiler they are just int* pointing to address. But for compiler type of as an array is int[2] and type of b as an array is int[3] which are completely different from each other. Again just the compiler's perspective. For better understanding let the two arrays be int a[2], int b[3].

**Implementation:**

- Passing an array to function in a classical way.
- Later on, trying to call for-each loop on it, a clear difference can be obtained.

**Example:**

### C++

```cpp
// C++ Program to demonstrate Above Approach

// Importing input output stream to take input
// and to display anything on the console
#include <iostream>

using namespace std;

// Method 1
// To print array elements
void print(int arr[], size_t n)
{

    // Iterating over elements on an array
    // using the foreach loop
    for (int element : arr) {
        // Print the elements of the array
        cout << element << " ";
    }

    // New line as all the desired elements are printed
    cout << endl;
}

// Method 2
// Main driver method
int main()
{
    // Declaring and initializing Integer array with
    // custom input entries
    int a[]{ 1, 2, 3, 4 };

    size_t size = sizeof(a) / sizeof(a[0]);

    // Calling the Method1 as created above
    // in the main) method to
    // print array elements
    print(a, size);
}
```

**Output:**

```
test.cpp: In function 'void print(int*, size_t)':
test.cpp:5:21: error: 'begin' was not declared in this scope
    for(int element: arr){
```

**Output Explanation:**

Here it is clear that int* doesn't have any information about the underlying array but if you pass an array by reference using the template the above code will work. As reference array retains information about underlying array and its type would be int[4], not int*.

Now let us discuss a reference to an array.

**Methods:**

1. **Naive method**
2. **Reference to an Array**

**Method 1:** Naive method

First most the common way that comes into our mind is described below syntactically. This is clearly a Naive approach as it loses its array identity.

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

***Note:*** *int a[] = b; will not work as array can only be initialized using aggregate object*

**Method 2**: Reference to Array

- The ***size*** needs to be mentioned as int[x] and int[y] are different data types from the compiler's perspective.
- Reference to array needs to be initialized at the time of declaration.
- ***(&name)*** is not redundant. It has its own meaning.

**Syntax:**

```
data_type (&name)[size] = array;
```

int (*a)[10]:

a的类型为int (*)[10],即去掉名字a, a是一个指向 10元素数组, int [10], 的指针

(*a)[10] is an int

***Note: data_type &name[size]*** *is incorrect because it means an array of reference to some datatype which is clearly meaningless. By doing so we have "name" of data type* ***int (&) []*** *which is, of course, different from* ***int[]***

**Example:**

### C++

```cpp
// C++ Program to demonstrate Reference to an Array

// Importing input output classes
#include <iostream>
using namespace std;

// Main driver method
int main()
{
    // Creating and initializing an integer array
    // Custom input entries
    int a[]{ 1, 2, 3, 4 };

    // int (&b)[] = a;
    // Declaring this way wont work as an error will be
    // thrown invalid initialization of reference of type
    // 'int (&)[]' from expression of type 'int [4]' Here
    // you see compiler referred to "a" as int [4] not int*

    int(&b)[4] = a;

    // Iterating over elements using foreach loop
    for (int e : b) {
        // Print the elements of the array
        cout << e << " ";
    }
}
```

**Output**

```
1 2 3 4
```

applications
10, Sep 18

02 Difference between Call by Value and Call by Reference
26, Dec 18

03 Can C++ reference member be declared without being initialized with declaration?
16, Jan 19

04 Different ways to use Const with Reference to a Pointer in C++
03, Apr 20

06 Why do we need reference variables if we have pointers
30, Jun 20

07 Passing Reference to a Pointer in C++
19, Sep 18

08 When do we pass arguments by reference or pointer?
25, Jul 11

**Article Contributed By :**

mohdatishanx
@mohdatishanx

**Vote for difficulty**
Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : Code_Mech, adnanirshad158, simranarora5sos

Article Tags : cpp-array, C++

Practice Tags : CPP

Improve Article Report Issue

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

Load Comments