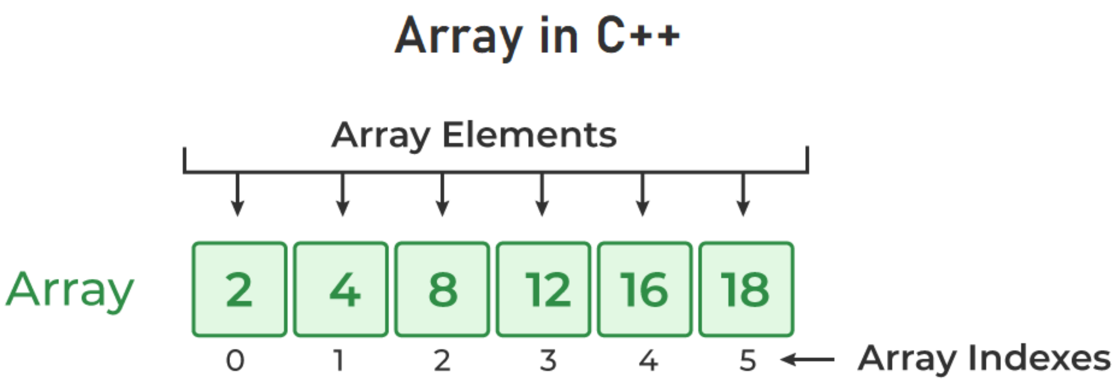
C++ Arrays

In C++, an array is a data structure that is used to store multiple values of similar data types in a contiguous memory location.

For example, if we have to store the marks of 4 or 5 students then we can easily store them by creating 5 different variables but what if we want to store marks of 100 students or say 500 students then it becomes very challenging to create that numbers of variable and manage them. Now, arrays come into the picture that can do it easily by just creating an array of the required size.



Properties of Arrays in C++

- An Array is a collection of data of the same data type, stored at a contiguous memory location.
- Indexing of an array starts from **0.** It means the first element is stored at the 0th index, the second at 1st, and so on.
- Elements of an array can be accessed using their indices.
- Once an array is declared its size remains constant throughout the program.
- An array can have multiple dimensions.
- The size of the array in bytes can be determined by the sizeof operator using which we can also find the number of elements in the array.
- We can find the size of the type of elements stored in an array by subtracting adjacent addresses.

Array Declaration in C++

In C++, we can declare an array by simply specifying the data type first and then the name of an array with its size.

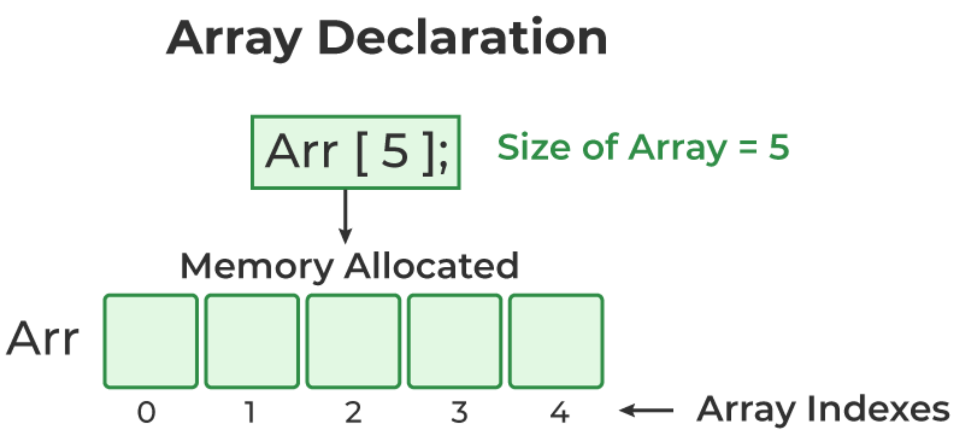
data_type array_name[Size_of_array];

Example

int arr[5];

Here,

- **int:** It is the type of data to be stored in the array. We can also use other data types such as char, float, and double.
- **arr:** It is the name of the array.
- **5:** It is the size of the array which means only 5 elements can be stored in the array.



Initialization of Array in C++

In C++, we can initialize an array in many ways but we will discuss some most common ways to initialize an array. We can initialize an array at the time of declaration or after declaration.

1. Initialize Array with Values in C++

We have initialized the array with values. The values enclosed in curly braces '{}' are assigned to the array. Here, 1 is stored in arr[0], 2 in arr[1], and so on. Here the size of the array is 5.

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

2. Initialize Array with Values and without Size in C++

We have initialized the array with values but we have not declared the length of the array, therefore, the length of an array is equal to the number of elements inside curly braces.

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

3. Initialize Array after Declaration (Using Loops)

We have initialized the array using a loop after declaring the array. This method is generally used when we want to take input from the user or we cant to assign elements one by one to each index of the array. We can modify the loop conditions or change the initialization values according to requirements.

```
for (int i = 0; i < N; i++) {
    arr[i] = value;
}
```

4. Initialize an array partially in C++

Here, we have declared an array 'partialArray' with size '5' and with values '1' and '2' only. So, these values are stored at the first two indices, and at the rest of the indices '0' is stored.

```
int partialArray[5] = {1, 2};
```

5. Initialize the array with zero in C++

We can initialize the array with all elements as '0' by specifying '0' inside the curly braces. This will happen in case of zero only if we try to initialize the array with a different value say '2' using this method then '2' is stored at the 0th index only.

```
int zero_array[5] = {0};
```

Accessing an Element of an Array in C++

Elements of an array can be accessed by specifying the name of the array, then the index of the element enclosed in the array subscript operator []. For example, arr[i].

Example 1: The C++ Program to Illustrate How to Access Array Elements

```
//  C++ Program to Illustrate How to Access Array Elements

#include <iostream>

using namespace std;


int main()

{


    int arr[3];


    // Inserting elements in an array

    arr[0] = 10;

    arr[1] = 20;

    arr[2] = 30;


    // Accessing and printing elements of the array

    cout << "arr[0]: " << arr[0] << endl;

    cout << "arr[1]: " << arr[1] << endl;
    cout << "arr[2]: " << arr[2] << endl;

    return 0;
}
```

Output

```
arr[0]: 10
arr[1]: 20
arr[2]: 30
```

Update Array Element

To update an element in an array, we can use the index which we want to update enclosed within the array subscript operator and assign the new value.

```
arr[i] = new_value;
```

Traverse an Array in C++

We can traverse over the array with the help of a loop using indexing in C++. First, we have initialized an array ‘table_of_two’ with a multiple of 2. After that, we run a for loop from 0 to 9 because in an array indexing starts from zero. Therefore, using the indices we print all values stored in an array.

Example 2: The C++ Program to Illustrate How to Traverse an Array

```
// C++ Program to Illustrate How to Traverse an Array
#include <iostream>
using namespace std;

int main()
{

    // Initialize the array
    int table_of_two[10]
        = { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 };

    // Traverse the array using for loop
    for (int i = 0; i < 10; i++) {
        // Print the array elements using indexing
        cout << table_of_two[i] << " ";
    }

    return 0;
}
```

Output

```
2 4 6 8 10 12 14 16 18 20
```

Size of an Array in C++

In C++, we do not have the length function as in Java to find array size but **we can calculate the size of an array using sizeof() operator** trick. First, we find the size occupied by the whole array in the memory and then divide it by the size of the type of element stored in the array. This will give us the number of elements stored in the array.

```
data_type size = sizeof(Array_name) / sizeof(Array_name[index]);
```

Example 3: The C++ Program to Illustrate How to Find the Size of an Array

```
// C++ Program to Illustrate How to Find the Size of an
// Array
#include <iostream>
using namespace std;

int main()
{
    int arr[] = { 1, 2, 3, 4, 5 };

    // Size of one element of an array
    cout << "Size of arr[0]: " << sizeof(arr[0]) << endl;

    // Size of array 'arr'
    cout << "Size of arr: " << sizeof(arr) << endl;
    // Length of an array
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Length of an array: " << n << endl;

    return 0;
}
```

Output

```
Size of arr[0]: 4
Size of arr: 20
Length of an array: 5
```