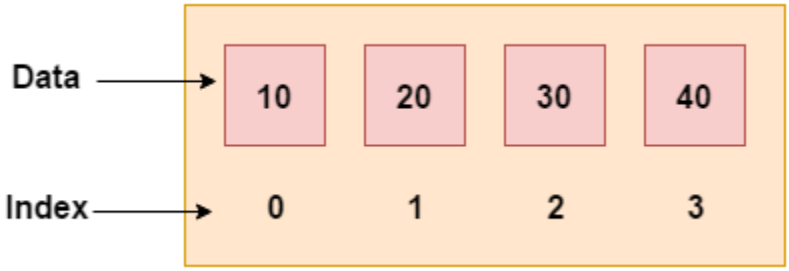
Introduction

A collection of related data items stored in adjacent memory places is referred to as an array in any programming language. Elements of an array can be accessed arbitrarily using it's indices. They can be used to store a collection of any type of primitive data type, including int, float, double, char, etc. An array in C/C++ can also store derived data types like structures, pointers, and other data types, which is an addition. An array can be represented as,



Advantages of C++ Array

Code Optimization (less code)

Random Access

Easy to traverse data

Easy to manipulate data

Easy to sort data etc.

Disadvantages of C++ Array

Fixed size

C++ array with empty members

The maximum number of elements that can be stored in an array in C++ is n. What will happen, though, if we store fewer than n elements?

For example,

// store only 3 elements in the array

int x[6] = {19, 10, 8};

The array x in this case is 6 elements wide. But we've only given it a three-element initialization.

When that happens, the compiler fills in the empty spaces with random values. This random value frequently appears as 0.

Important things to remember while using arrays in C++

The array's indexes begin at 0. Meaning that the first item saved at index 0 is x[0].

The final element of an array with size n is kept at index (n-1). This example's final element is x[5].

An array's elements have sequential addresses. Consider the scenario where x[0beginning]'s address is 2120.

The address of the subsequent element, x[1], will then be 2124, followed by x[2], 2128, and so forth.

Each element in this case has a four-fold increase in size. This is due to the fact that int has a 4 byte capacity.

Types of arrays in C++

1. Single Dimensional Array

2. Multidimensional Array

C++ Single Dimensional Array

Simple example of C++ array, where we are going to create, initialize and traverse array.

```
#include <iostream>

using namespace std;

int main()

{

 int arr[5]={10, 0, 20, 0, 30};  //creating and initializing array

      //traversing array

      for (int i = 0; i < 5; i++)

      {

         cout<<arr[i]<<"\n";

      }

}
```

C++ Array Example: Traversal using foreach loop

We can also traverse the array elements using foreach loop. It returns array element one by one.

```
#include <iostream>

using namespace std;

int main()

{

 int arr[5]={10, 0, 20, 0, 30}; //creating and initializing array

       //traversing array

      for (int i: arr)

       {

          cout<<i<<"\n";

       }

}
```

What is two-dimensional array?

Each element in this kind of array is described by two indexes, the first of which denotes a row and the second of which denotes a column.

As you can see, the components are arranged in a two-dimensional array using rows and columns; there are I number of rows and j number of columns.

What is a multi-dimensional array?

A two-dimensional array is the most basic type of multidimensional array; it also qualifies as a multidimensional array. There are no restrictions on the array's dimensions.

How to insert element in an array?

```
int mark[5] = {19, 10, 8, 17, 9}

// change 4th element to 9

mark[3] = 9;

// take input from the user

// store the value at third position

cin >> mark[2];

// take input from the user

// insert at ith position

cin >> mark[i-1];


// print first element of the array

cout << mark[0];

// print ith element of the array

cout >> mark[i-1];
```

How to display array elements?

```
#include <iostream>

using namespace std;

int main() {

int numbers[5] = {7, 5, 6, 12, 35};

cout << "The numbers are: ";
```

```
//  Printing array elements
// using range-based for loop
for (const int &n : numbers) {
cout << n << "  ";
}
cout << "\nThe numbers are: ";
//  Printing array elements
// using traditional for loop
for (int i = 0; i < 5; ++i) {
cout << numbers[i] << "  ";
}
return 0;
}
```

Output:

```
The numbers are: 7  5  6  12  35
The numbers are: 7  5  6  12  35
```

Program1: To display the sum and average of array elements.

```
#include <iostream>
using namespace std;
int main() {
// initialize an array without specifying the size
```

```
double numbers[] = {7, 5, 6, 12, 35, 27};
double sum = 0;
double count = 0;
double average;
cout << "The numbers are: ";
 //  print array elements
 // use of range-based for loop
 for (const double &n : numbers) {
  cout << n << "  ";
//  calculate the sum
sum += n;
// count the no. of array elements
++count;
  }
// print the sum
cout << "\nTheir Sum = " << sum << endl;
// find the average
average = sum / count;
cout << "Their Average = " << average << endl;
  return 0;
}
```

Output:

The numbers are: 7 5 6 12 35 27

Their Sum = 92

Their Average = 15.3333