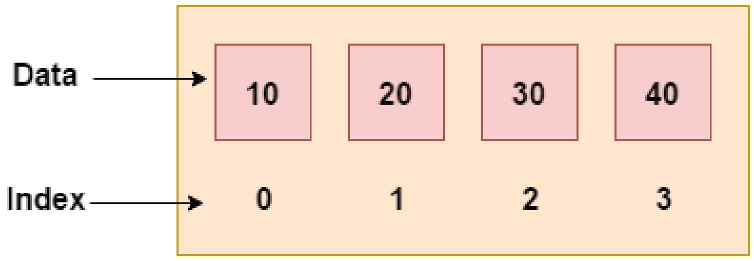
C++ Arrays

Like other programming languages, array in C++ is a group of similar types of elements that have contiguous memory location.

In C++ **std::array** is a container that encapsulates fixed size arrays. In C++, array index starts from 0. We can store only fixed set of elements in C++ array.

A collection of related data items stored in adjacent memory places is referred to as an array in the C/C++ programming language or any other programming language for that matter. Elements of an array can be accessed arbitrarily using its 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. The array representation in a picture is provided below.



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 Types

There are 2 types of arrays in C++ programming:

1. Single Dimensional Array
2. Multidimensional Array

C++ Single Dimensional Array

Let's see a 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";
        }
}
```

Output:

```
10
0
20
0
30
```

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";
        }
}
```

Output:

```
10
20
30
40
50
```

Why do we need arrays?

With a limited number of objects, we can use regular variables (v1, v2, v3,..), but when we need to hold many instances, managing them with normal variables becomes challenging. To represent numerous instances in one variable, we use an array.

What happens if we try to access out of bound array?

The array's elements will be numbered 0 through 9 if we define an array of size 10.

We will have Undefined Behaviour if we attempt to access an element at an index higher than 10, though.

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++

1. The array's indexes begin at 0. Meaning that the first item saved at index 0 is x[0].
2. The final element of an array with size n is kept at index (n-1). This example's final element is x[5].
3. 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.

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 it in 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 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
```

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
```

← Prev   Next →

Youtube For Videos Join Our Youtube Channel: Join Now

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

Splunk tutorial Splunk	SPSS tutorial SPSS	Swagger tutorial	T-SQL tutorial Transact-SQL

C++ Multidimensional Arrays

The multidimensional array is also known as rectangular arrays in C++. It can be two dimensional or three dimensional. The data is stored in tabular form (row ∗ column) which is also known as matrix.

C++ Multidimensional Array Example

Let's see a simple example of multidimensional array in C++ which declares, initializes and traverse two dimensional arrays.

```
#include <iostream>
using namespace std;
int main()
{
  int test[3][3];  //declaration of 2D array
    test[0][0]=5;  //initialization
    test[0][1]=10;
    test[1][1]=15;
    test[1][2]=20;
    test[2][0]=30;
    test[2][2]=10;
    //traversal
    for(int i = 0; i < 3; ++i)
    {
        for(int j = 0; j < 3; ++j)
        {
            cout<< test[i][j]<<" ";
        }
        cout<<"\n"; //new line at each row
    }
    return 0;
}
```

Output:

```
5 10 0
0 15 20
30 0 10
```

C++ Multidimensional Array Example: Declaration and initialization at same time

Let's see a simple example of multidimensional array which initializes array at the time of declaration.

```
#include <iostream>
using namespace std;
int main()
{
  int test[3][3] =
    {
      {2, 5, 5},
      {4, 0, 3},
      {9, 1, 8}  };  //declaration and initialization
    //traversal
    for(int i = 0; i < 3; ++i)
    {
      for(int j = 0; j < 3; ++j)
      {
        cout<< test[i][j]<<" ";
      }
      cout<<"\n"; //new line at each row
    }
    return 0;
}
```

Output:"

```
2 5 5
4 0 3
```

```
9 1 8
```

← Prev Next →

AD

Youtube For Videos Join Our Youtube Channel: Join Now

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

Splunk tutorial Splunk	SPSS tutorial SPSS	Swagger tutorial Swagger	T-SQL tutorial Transact-SQL
Tumblr tutorial Tumblr	React tutorial ReactJS	Regex tutorial Regex	Reinforcement learning tutorial Reinforcement Learning

C++ Passing Array to Function

In C++, to reuse the array logic, we can create function. To pass array to function in C++, we need to provide only array name.

```
functionname(arrayname); //passing array to function
```

C++ Passing Array to Function Example: print array elements

Let's see an example of C++ function which prints the array elements.

```
#include <iostream>
using namespace std;
void printArray(int arr[5]);
int main()
{
        int arr1[5] = { 10, 20, 30, 40, 50 };
        int arr2[5] = { 5, 15, 25, 35, 45 };
        printArray(arr1); //passing array to function
        printArray(arr2);
}
void printArray(int arr[5])
{
    cout << "Printing array elements:"<< endl;
    for (int i = 0; i < 5; i++)
    {
                cout<<arr[i]<<"\n";
    }
}
```

Output:

```
Printing array elements:
10
20
30
40
```

```
50
Printing array elements:
5
15
25
35
45
```

C++ Passing Array to Function Example: Print minimum number

Let's see an example of C++ array which prints minimum number in an array using function.

```
#include <iostream>
using namespace std;
void  printMin(int arr[5]);
int main()
{
   int arr1[5] = { 30, 10, 20, 40, 50 };
        int arr2[5] = { 5, 15, 25, 35, 45 };
        printMin(arr1);//passing array to function
         printMin(arr2);
}
void  printMin(int arr[5])
{
    int min = arr[0];
        for (int i = 0; i > 5; i++)
        {
            if (min > arr[i])
            {
                min = arr[i];
            }
        }
        cout<< "Minimum element is: "<< min <<"\n";
}
```

Output:

```
Minimum element is: 10
Minimum element is: 5
```

C++ Passing Array to Function Example: Print maximum number

Let's see an example of C++ array which prints maximum number in an array using function.

```
#include <iostream>
using namespace std;
void  printMax(int arr[5]);
int main()
{
        int arr1[5] = { 25, 10, 54, 15, 40 };
        int arr2[5] = { 12, 23, 44, 67, 54 };
        printMax(arr1); //Passing array to function
         printMax(arr2);
}
void  printMax(int arr[5])
{
    int max = arr[0];
        for (int i = 0; i < 5; i++)
        {
            if (max < arr[i])
            {
                max = arr[i];
            }
        }
        cout<< "Maximum element is: "<< max <<"\n";
}
```

Output:

```
Maximum element is: 54
Maximum element is: 67
```

← Prev Next →