ARRAY IN C++

C++ provides a data structure, **the array**, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

Declaring Arrays

To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array as follows

```
type arrayName [ arraySize ];
```

Array declaration by specifying the size

- C++
- C

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

int main()
```

ARRAY IN C++

```
{
    // array declaration by specifying size
    int arr1[10];

    // With recent C/C++ versions, we can also
    // declare an array of user specified size
    int n = 10;
    int arr2[n];

    return 0;
}
```

Array declaration by initializing elements

C++

C

```
// Array declaration by initializing elements
#include <iostream>
using namespace std;
int main()
{
    int arr[] = { 10, 20, 30, 40};
    return 0;
// Compiler creates an array of size 4.
// above is same as  "int arr[4] = {10, 20, 30, 40}"
}
```

Array declaration by specifying the size and initializing elements

C++

ARRAY IN C++

C

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

int main()
{
    // Array declaration by specifying size and initializing
    // elements
    int arr[6] = { 10, 20, 30, 40 };

    // Compiler creates an array of size 6, initializes first
    // 4 elements as specified by user and rest two elements as
    // 0. above is same as  "int arr[] = {10, 20, 30, 40, 0, 0}"

    return 0;
}
```

Advantages of an Array in C/C++:

Random access of elements using the array index.

Use of fewer lines of code as it creates a single array of multiple elements.

Easy access to all the elements.

Traversal through the array becomes easy using a single loop.

Sorting becomes easy as it can be accomplished by writing fewer lines of code.

Disadvantages of an Array in C/C++:

Allows a fixed number of elements to be entered which is decided at the time of declaration. Unlike a linked list, an array in C is not dynamic.

Insertion and deletion of elements can be costly since the elements are needed to be managed in accordance with the new memory allocation.

ARRAY IN C++

Example

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

int main()
{
    int arr[5];
    arr[0] = 5;
    arr[2] = -10;

    // this is same as arr[1] = 2
    arr[3 / 2] = 2;
    arr[3] = arr[0];

    cout << arr[0] << " " << arr[1] << " " << arr[2] << " "
         << arr[3];

    return 0;
}
```

Output

```
5 2 -10 5
```

No Index Out of bound Checking:

There is no index out of bounds checking in C/C++, for example, the following program compiles fine but may produce unexpected output when run.

```
// This C++ program compiles fine
// as index out of bound
// is not checked in C.
```

ARRAY IN C++

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

int main()
{
    int arr[2];

    cout << arr[3] << " ";
    cout << arr[-2] << " ";

    return 0;
}
```

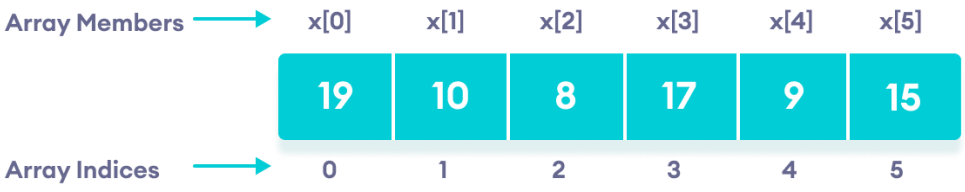
Output

```
211343841 4195777
```

C++ Array Initialization

In C++, it's possible to initialize an array during declaration. For example,

```
// declare and initialize and array
int x[6] = {19, 10, 8, 17, 9, 15};
```



Take Inputs from User and Store Them in an Array

```
#include <iostream>
```

ARRAY IN C++

```cpp
using namespace std;

int main() {

  int numbers[5];

  cout << "Enter 5 numbers: " << endl;

  //  store input from user to array
  for (int i = 0; i < 5; ++i) {
    cin >> numbers[i];
  }

  cout << "The numbers are: ";

  //  print array elements
  for (int n = 0; n < 5; ++n) {
    cout << numbers[n] << "  ";
  }

  return 0;
}
```

Output

```
Enter 5 numbers:
11
12
13
14
15
The numbers are: 11  12  13  14  15
```

ARRAY IN C++

Display Sum and Average of Array Elements Using for Loop

```
int main() {

  // initialize an array without specifying 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;
}
```

ARRAY IN C++

Output

```
The numbers are: 7  5  6  12  35  27
Their Sum = 92
Their Average = 15.3333
```

Write a C++ program to find the largest element of a given array of integers

```
#include<iostream>
using namespace std;
int find_largest(int nums[], int n) {
  return *max_element(nums, nums + n);
}

int main() {
  int nums[] = {
    5,
    4,
    9,
    12,
    8
  };
  int n = sizeof(nums) / sizeof(nums[0]);
    cout << "Original array:";
    for (int i=0; i < n; i++)
    cout << nums[i] <<" ";

  cout << "\nLargest element of the said array: "<<
find_largest(nums, n);
  return 0;
}
```

ARRAY IN C++

C++ Multidimensional Arrays

In this tutorial, we'll learn about multi-dimensional arrays in C++. More specifically, how to declare them, access them, and use them efficiently in our program.

In C++, we can create an array of an array, known as a multidimensional array. For example:

```
int x[3][4];
```

Here, `x` is a two-dimensional array. It can hold a maximum of 12 elements.

We can think of this array as a table with 3 rows and each row has 4 columns as shown below.

	Col 1	Col 2	Col 3	Col 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Three-dimensional arrays also work in a similar way. For example:

ARRAY IN C++

```
float x[2][4][3];
```

This array `x` can hold a maximum of 24 elements.
We can find out the total number of elements in the array simply by multiplying its dimensions:

```
2 x 4 x 3 = 24
```

Multidimensional Array Initialization

Like a normal array, we can initialize a multidimensional array in more than one way.

1. Initialization of two-dimensional array

```
int test[2][3] = {2, 4, 5, 9, 0, 19};
```

The above method is not preferred. A better way to initialize this array with the same array elements is given below:

```
int  test[2][3] = { {2, 4, 5}, {9, 0, 19}};
```

This array has 2 rows and 3 columns, which is why we have two rows of elements with 3 elements each.

ARRAY IN C++

	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

Example 1: Two Dimensional Array

```cpp
// C++ Program to display all elements
// of an initialised two dimensional array

#include <iostream>
using namespace std;

int main() {
    int test[3][2] = {{2, -5},
                      {4, 0},
                      {9, 1}};

    // use of nested for loop
    // access rows of the array
    for (int i = 0; i < 3; ++i) {

        // access columns of the array
        for (int j = 0; j < 2; ++j) {
            cout << "test[" << i << "][" << j << "] = " << test[i][j] << endl;
        }
    }

    return 0;
}
```

ARRAY IN C++

Example 2: Taking Input for Two Dimensional Array

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

int main() {
    int numbers[2][3];

    cout << "Enter 6 numbers: " << endl;

    // Storing user input in the array
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 3; ++j) {
            cin >> numbers[i][j];
        }
    }

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

    //  Printing array elements
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 3; ++j) {
            cout << "numbers[" << i << "][" << j << "]: " << numbers[i][j] <<
endl;
        }
    }

    return 0;
}
```

Output

```
Enter 6 numbers:
1
2
3
4
5
6
The numbers are:
numbers[0][0]: 1
```

```
numbers[0][1]: 2
numbers[0][2]: 3
numbers[1][0]: 4
numbers[1][1]: 5
numbers[1][2]: 6
```

Output

```
Enter 6 numbers:
1
2
3
4
5
6
The numbers are:
numbers[0][0]: 1
numbers[0][1]: 2
numbers[0][2]: 3
numbers[1][0]: 4
numbers[1][1]: 5
numbers[1][2]: 6
```

Output

```
test[0][0] = 2
test[0][1] = -5
test[1][0] = 4
test[1][1] = 0
test[2][0] = 9
test[2][1] = 1
```

ARRAY IN C++

C++ Program to Add Two Matrix Using Multi-dimensional Arrays

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

int main()
{
    int r, c, a[100][100], b[100][100], sum[100][100], i, j;

    cout << "Enter number of rows (between 1 and 100): ";
    cin >> r;

    cout << "Enter number of columns (between 1 and 100): ";
    cin >> c;

    cout << endl << "Enter elements of 1st matrix: " << endl;

    // Storing elements of first matrix entered by user.
    for(i = 0; i < r; ++i)
       for(j = 0; j < c; ++j)
       {
           cout << "Enter element a" << i + 1 << j + 1 << " : ";
           cin >> a[i][j];
       }

    // Storing elements of second matrix entered by user.
    cout << endl << "Enter elements of 2nd matrix: " << endl;
    for(i = 0; i < r; ++i)
       for(j = 0; j < c; ++j)
       {
           cout << "Enter element b" << i + 1 << j + 1 << " : ";
           cin >> b[i][j];
       }

    // Adding Two matrices
    for(i = 0; i < r; ++i)
        for(j = 0; j < c; ++j)
```

ARRAY IN C++

```
            sum[i][j] = a[i][j] + b[i][j];

    // Displaying the resultant sum matrix.
    cout << endl << "Sum of two matrix is: " << endl;
    for(i = 0; i < r; ++i)
        for(j = 0; j < c; ++j)
        {
            cout << sum[i][j] << "  ";
            if(j == c - 1)
                cout << endl;
        }

    return 0;
}
```

Output

```
Enter number of rows (between 1 and 100): 2
Enter number of columns (between 1 and 100): 2

Enter elements of 1st matrix:
Enter element a11: -4
Enter element a12: 5
Enter element a21: 6
Enter element a22: 8

Enter elements of 2nd matrix:
Enter element b11: 3
Enter element b12: -9
Enter element b21: 7
Enter element b22: 2

Sum of two matrix is:
-1   -4
13   10
```

ARRAY IN C++

Passing Array to a Function in C++ Programming

Syntax for Passing Arrays as Function Parameters

The syntax for passing an array to a function is:

```
returnType functionName(dataType arrayName[arraySize]) {
    // code
}
```

Let's see an example,

```
int total(int marks[5]) {
    // code
}
```

Example 1: Passing One-dimensional Array to a Function

```
// C++ Program to display marks of 5 students

#include <iostream>
using namespace std;

// declare function to display marks
// take a 1d array as parameter
void display(int m[5]) {
    cout << "Displaying marks: " << endl;

    // display array elements
    for (int i = 0; i < 5; ++i) {
        cout << "Student " << i + 1 << ": " << m[i] << endl;
    }
```

ARRAY IN C++

```
}

int main() {

    // declare and initialize an array
    int marks[5] = {88, 76, 90, 61, 69};

    // call display function
    // pass array as argument
    display(marks);

    return 0;
}
```

Output

```
Displaying marks:
Student 1: 88
Student 2: 76
Student 3: 90
Student 4: 61
Student 5: 69
```

Passing Multidimensional Array to a Function

Example 2: Passing Multidimensional Array to a Function

```
// C++ Program to display the elements of two
// dimensional array by passing it to a function

#include <iostream>
using namespace std;

// define a function
// pass a 2d array as a parameter
void display(int n[][2]) {
    cout << "Displaying Values: " << endl;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 2; ++j) {
```

ARRAY IN C++

```
            cout << "num[" << i << "][" << j << "]: " << n[i][j] << endl;
        }
    }
}

int main() {

    // initialize 2d array
    int num[3][2] = {
        {3, 4},
        {9, 5},
        {7, 1}
    };
// call the function
    // pass a 2d array as an argument
    display(num);

    return 0;
}
```