FAST National University of Computer and Emerging Sciences Peshawar

OOP Lab # 4.2

DEPARTMENT OF COMPUTER SCIENCE

C++ Programming (Arrays)

Computer Instructor: Muhammad Abdullah Orakzai

Programming

Prepared By: Muhammad Abdullah Orakzai (Lab Instructor CS)

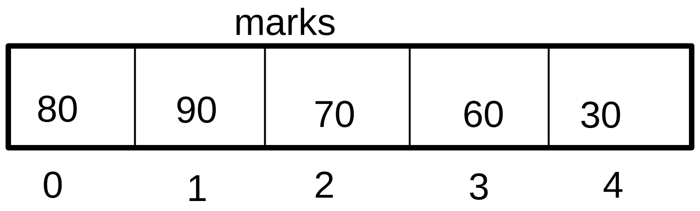
Contents

Array

- Same name which store multiple values.
- It is a collection of similar type of elements that have contiguous memory location.

Array is:

1. Linear data structure (consecutive location)
2. Static data structure (fixed size)
3. Homogeneous data will be stored.



Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

Array…

To declare an array, define the variable type, specify the name of the array followed by **square brackets** and specify the number of elements it should store:

To create an array of five integers, you could write:

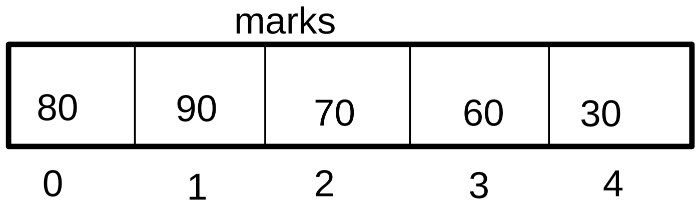
```
int myNum[5] =  { 10, 20, 30, 40, 50 };
```

One Dimensional Array

A **one-dimensional array** is a structured collection of components (often called array elements) that can be accessed individually by specifying the position of a component with a single index value.

To create an array of five integers, you could write:

```
int marks[5] = {80, 90, 70, 60, 30};
```



Array Declaration

```
dataType arrayName[arraySize];
```

For example,

```
int x[6];
```

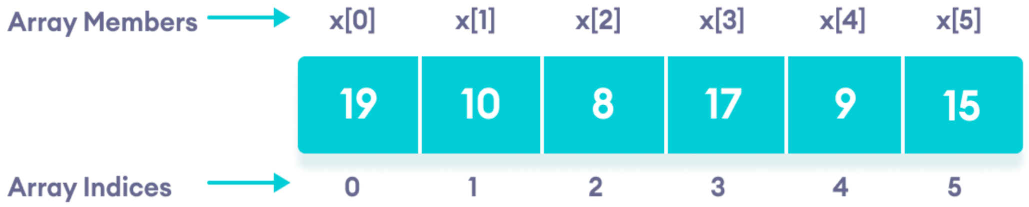
Here,

- int - type of element to be stored
- x - name of the array
- 6 - size of the array

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 };
```



Array Initialization…

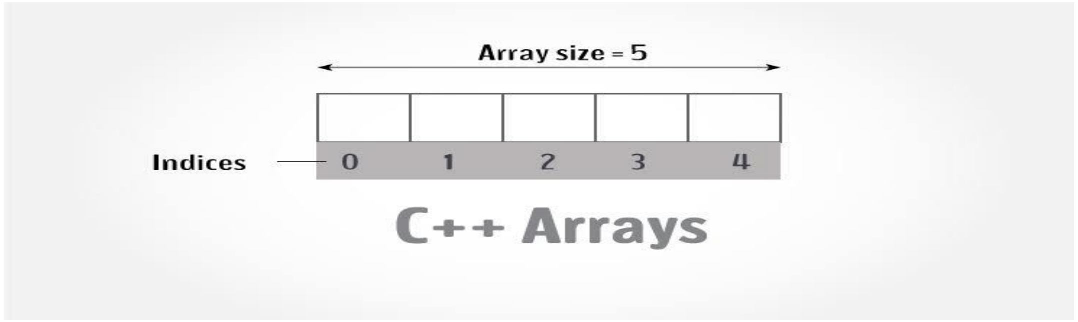
Another method to initialize array during declaration:

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

Here, we have not mentioned the size of the array. In such cases, the compiler automatically computes the size.

Access the Elements of an Array

- To access the element you have to provide the index number along with the name.



```
cout << myNum[0];
```

Access the Elements of an Array…

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

int main() {

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

cout<<myNum[0]<<endl;
cout<<myNum[1]<<endl;
cout<<myNum[2]<<endl;
cout<<myNum[3]<<endl;
cout<<myNum[4]<<endl;

  return 0;
}
```

Output:

```
1
2
3
4
5
```

Array with empty members

In C++, if an array has a size n, we can store upto n number of elements in the array. However, what will happen if we store less than n number of elements.

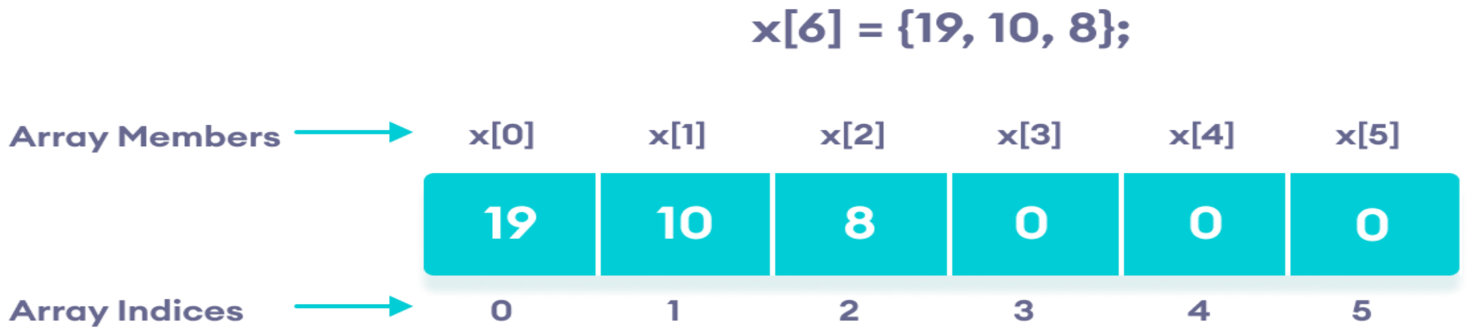
For example,

```
// store only 3 elements in the array
 int x[6] = {19, 10, 8};
```

Here, the array x has a size of 6. However, we have initialized it with only 3 elements.

In such cases, the compiler assigns random values to the remaining places. Oftentimes, this random value is simply 0.

Array with empty members…



Array with empty members

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

int main() {

int myNum[6] ={1,2,3};

for(int number: myNum)
{
    cout<<number<<endl;
}



  return 0;

}
```

Output:

```
1
2
3
0
0
0
```

Change an Array Element

To change the value of a specific element, refer to the index number:

Example

```
myNum [2] = 4000;
```

Change an Array Element...

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

int main() {

int myNum[] ={1,2,3,4,5,6,7};

cout<<"Value at myNum[2]: "<<myNum[2]<<endl;

myNum[2]=7777;

cout<<"Value at myNum[2]: "<<myNum[2];

  return 0;
}
```

Output:

```
Value at myNum[2]: 3

Value at myNum[2]: 7777
```

Loop through an Array (for loop)

You can loop through the array elements with the for loop.

The following example outputs all elements in the **myNum** array:

```
int myNum[5] = {10, 20, 30, 40, 50};
for(int i = 0;  i < 4; i++)
{
  cout <<myNum[i] <<endl;
}
```

Loop through an Array (for loop)...

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

int main() {
int myNum[] ={1,2,3,4,5,6,7};

cout<<"***Array Iteration Using for loop***"<<endl;

for(int i=0; i<sizeof(myNum)/sizeof(int); i++)
{

cout<<myNum[i]<<endl;

}
  return 0;
}
```

Output:

```
1
2
3
4
5
6
7
```

Loop through an Array (for loop)...

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

int main() {
int myNum[] ={1,2,3,4,5,6,7};

cout<<"***Array Iteration Using for loop***"<<endl;
for(int i=0; i<7; i++)
{

cout<<myNum[i]<<endl;

}
  return 0;
}
```

Output:

```
1
2
3
4
5
6
7
```

Loop through an Array (enhanced for loop)…

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

int main() {
int myNum[] ={1,2,3,4,5,6,7};

cout<<"***Array Iteration Using enhanced for loop***"<<endl;

for (int number : myNum)
{
    cout<<number<<endl;
}
  return 0;
}
```

Output:

```
***Array Iteration Using enhanced for loop***
1
2
3
4
5
6
7
```

Advantages of an Array in C/C++

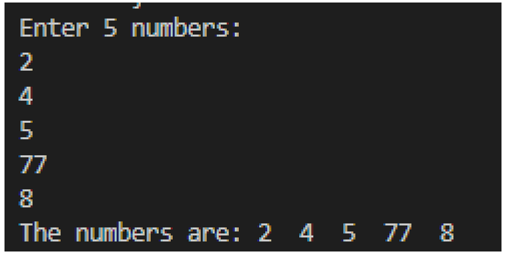
1. Random access of elements using array index.
2. Use of less line of code as it creates a single array of multiple elements.
3. Easy access to all the elements.
4. Traversal through the array becomes easy using a single loop.
5. Sorting becomes easy as it can be accomplished by writing less line of code.

Disadvantages of an Array in C/C++

1. 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.
2. Insertion and deletion of elements can be costly since the elements are needed to be managed in accordance with the new memory allocation.

Take Inputs from User and Store Them in an Array

```cpp
#include <iostream>
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;
}
```



```
Enter 5 numbers:
2
4
5
77
8
The numbers are: 2  4  5  77  8
```

Take Inputs from User and Store Them in an Array

Once again, we have used a for loop to iterate from i = 0 to i = 4. In each iteration, we took an input from the user and stored it in numbers[i].

Then, we used another for loop to print all the array elements.

String Arrays

```
#include<iostream>
using namespace std;
int main() {

string names[4] = {"Ali", "Asia", "Zain", "Zainab"};

for(int i = 0; i < 4; i++)
{
  cout << names[i] << "\n";
}

  return 0;
}
```

Output:

```
Ali
Asia
Zain
Zainab
```

String Arrays

The following example outputs the index of each element together with its value:

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

int main() {

string names[4] = {"Ali", "Asia", "Zain", "Zainab"}
;
for(int i = 0; i < 4; i++) {

cout << i << ": " << names[i] << "\n";

}
  return 0;
}
```

Output:

```
0: Ali
1: Asia
2: Zain
3: Zainab
```

How to Pass 1D array to function

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

void arrayIterationFunction(int test[])
{
    for (int i = 0; i <6 ; i++)
    {
        cout<<"myNum["<<i<<"]="<<test[i]<<endl;
    }
}

int main()
{
    int myNum[]= {1,3,4,5,6,7};
    arrayIterationFunction(myNum);
}
```

Output:

```
myNum[0]=1
myNum[1]=3
myNum[2]=4
myNum[3]=5
myNum[4]=6
myNum[5]=7
```

How to Pass 1D array to function…

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

int array_size;    // initialization at the top
void arrayIterationFunction(int test[])
{
    for (size_t i = 0; i <array_size; i++)
    {
        cout<<"Enter value at test["<<i<<"]=";
        cin>>test[i];
    }

    for (int i = 0; i <array_size; i++)
    {
        cout<<"myNum["<<i<<"]="<<test[i]<<endl;
    }
}
```

How to Pass 1D array to function…

```
int main()
{
    cout<<"Enter Array Size: ";
    cin>>array_size;
    int myNum[array_size];

    arrayIterationFunction(myNum);

}
```

Output:

```
Enter Array Size: 4
Enter value at test[0]=3
Enter value at test[1]=2
Enter value at test[2]=1
Enter value at test[3]=6

myNum[0]=3
myNum[1]=2
myNum[2]=1
myNum[3]=6
```

One Dimensional Array Tasks

1. Write a C++ program that will add two single dimensional array elements. Take values from user at runtime.
2. How to generate random number in C++, write a simple C++ program that will generate random number from 1 to 100 ?
3. Write a C++ program that will add two single dimensional arrays elements using random numbers.
4. Write a C++ program that will find maximum number in an array.
5. Write a C++ program that will find minimum number in a array.

Multidimensional Array

- C++ allows multidimensional arrays. Here is the general form of a multidimensional array declaration –
- type name[size1] [size2]...[sizeN];
- For example, the following declaration creates a three dimensional 5 . 10 . 4 integer array –

```
int   threedim[5][10][4];
```

But our focus will on 2- dimensional arrays

Two Dimensional Arrays (2D Array)

- The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays.
- An array that is represented with two indices/subscripts is called 2D array.
- It is similar to matrix in maths.
- Logically it consist of rows and columns.
- 2D array is called an array of an arrays.

Two Dimensional Arrays (2D Array)...

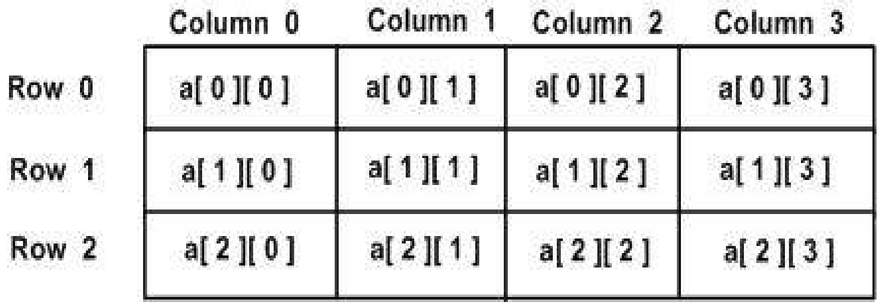
- To declare a two-dimensional integer array of size x,y, you would write something as follows:

  type arrayName [r][c];

- Where **type** can be any valid C++ data type and **arrayName** will be a valid C++ identifier.
- A two-dimensional array can be think as a table, which will have **r** number of rows and **c** number of columns.

Two Dimensional Arrays (2D Array)...

A 2-dimensional array **a**, which contains three rows and four columns can be shown as below:



	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Thus, every element in array a is identified by an element name of the form **a[r][c]**, where **a** is the name of the array, and **r** and **c** are the subscripts that uniquely identify each element in **a**.

2D Array logical Representation

	0	1	2
0	**(0,0)** 70	**(0,1)** 80	**(0,2)** 90
1	**(1,0)** 10	**(1,1)** 20	**(1,2)** 30
2	**(2,0)** 5	**(2,1)** 10	**(2,2)** 15
3	**(3,0)** 50	**(3,1)** 60	**(3,2)** 70

Initializing Two-Dimensional Arrays

Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row have 4 columns.

```
int a[3][4] = {    {0, 1, 2, 3} ,       /*  initializers for row indexed by 0 */
                   {4, 5, 6, 7} ,       /*  initializers for row indexed by 1 */
                   {8, 9, 10, 11}     /*  initializers for row indexed by 2 */
              } ;

int a[3][4] = {    {0, 1, 2, 3}    ,    {4, 5, 6, 7}    ,   {8, 9, 10, 11}  };
```

Initializing Two-Dimensional Arrays…

The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to previous example –

```
int a[3][4] = {0,1,2,3,   4,5,6,7,8,    9,10,11};
```

Accessing Two-Dimensional Array Elements

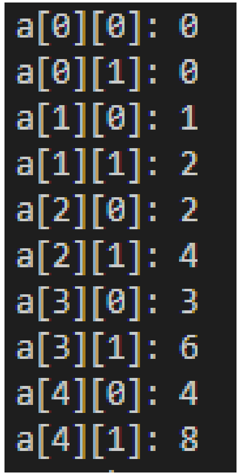
An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example –

```
int val = a[2][3];     //assigning value of a[2][3] that is 11 to variable val
cout<<a[2][3];
```

The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

Accessing Two-Dimensional Array Elements...

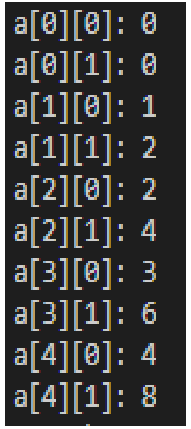
```
#include <iostream>
using namespace std;
int main () {
   // an array with 5 rows and 2 columns.
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};

   // output each array element's value
   for ( int r = 0; r < 5; r++ )
     {
           for ( int c = 0; c < 2; c++ )
           {

            cout << "a[" << r << "][" << c << "]: ";
            cout << a[r][c]<< endl;
           }
     }
   return 0;
}
```



```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

Accessing Two-Dimensional Array Elements

- When the above code is compiled and executed, it produces the following result –



```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

- As explained above, you can have arrays with any number of dimensions, although it is likely that most of the arrays you create will be of one or two dimensions.

Two Dimensional Array Tasks

1. Write a C++ program that will create 2D array using random numbers and then show these values.
2. Write a C++ program that will find maximum and minimum number in 2D array. Note array elements must be random values.
3. Write a C++ program that will add two 2D arrays elements. Take values from user at runtime. Note display values 1st, 2nd and their resultant array.

References

- https://beginnersbook.com/2017/08/cpp-data-types/
- https://www.geeksforgeeks.org/c-data-types/
- http://www.cplusplus.com/doc/tutorial/basic_io/
- https://www.geeksforgeeks.org/basic-input-output-c/
- https://www.w3schools.com/cpp/default.asp
- https://www.javatpoint.com/cpp-tutorial

THANK YOU