

Array



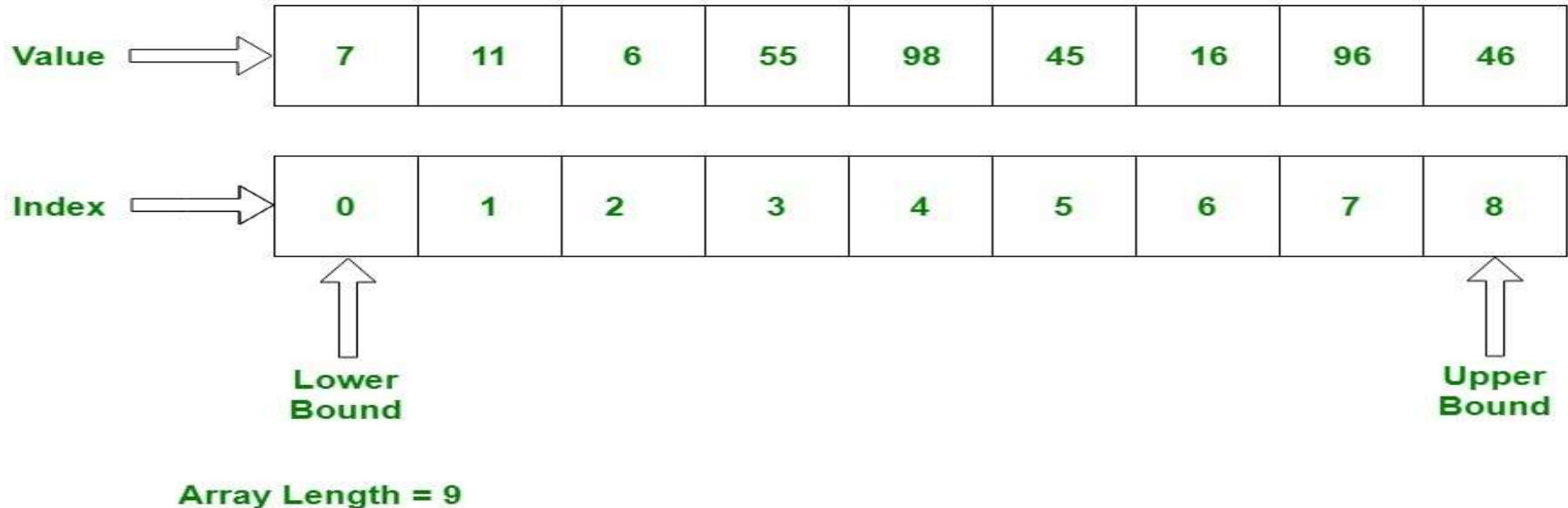
Today's Agenda

- Array.
- Array declaration and accessing array.
- Advantage and Disadvantage of Array.
- Multidimensional array
- Accessing Multidimensional Array
- Array Object
- String Class
- String Operation
- Coding Question

Let's Get Started-

Array

An array is a collection of similar items stored in contiguous memory locations. In programming, sometimes a simple variable is not enough to hold all the data. For example, let's say we want to store the marks of 500 students, having 500 different variables for this task is not feasible, we can define an array with size 500 that can hold the marks of all students



Declaring array

Method 1:-
`int arr[5];`
`arr[0] = 10;`
`arr[1] = 20;`
`arr[2] = 30;`
`arr[3] = 40;`
`arr[4] = 50;`

Method 2:-

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

Method 3:-

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

Accessing Array

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

```
int main(){
    int arr[] = {11, 22, 33, 44, 55};
    cout<<arr[0]<<endl;
    cout<<arr[1]<<endl;
    cout<<arr[2]<<endl;
    cout<<arr[3]<<endl;
    cout<<arr[4]<<endl;
    return 0;
}
```

Advantage of Array

- Random access of elements using array index.
- Use of less line 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 less line of code.

Disadvantage of Array

- 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 Rotation

Write a program of array rotation .

Input array:-

1 2 3 4 5 6

The rotation of array by one will make :-

2 3 4 5 6 1

Input array:-

1 2 3 4 5 6

The rotation of array by two will make :-

3 4 5 6 1 2

Move all zeroes to end of array

Input : arr[] = {1, 2, 0, 0, 0, 3, 6}

Output : 1 2 3 6 0 0 0

Input: arr[] = {0, 1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0, 9}

Output: 1 9 8 4 2 7 6 9 0 0 0 0 0

Multidimensional Array

General Syntax:

type name[size1][size2]...[sizeN];

Eg.

```
int arr[3][2];
```

Multidimensional Array

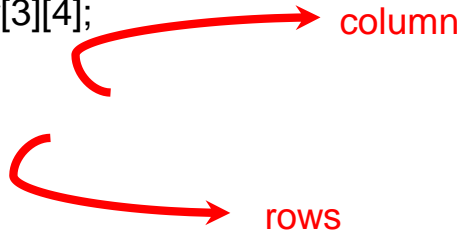
Two Dimensional Array:

General Syntax:

```
type arrayName [ x ][ y ];
```

Eg.

```
int arr[3][4];
```



	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]

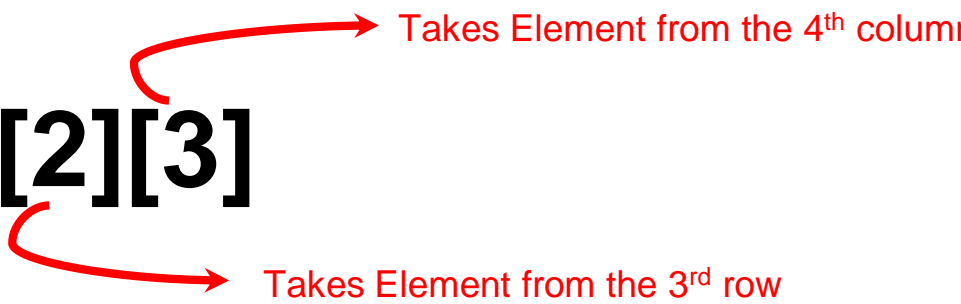
Multidimensional Array

Initializing 2D arrays:

```
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 */ };
```

Accessing Multidimensional Array

a[2][3]



Takes Element from the 4th column

Takes Element from the 3rd row

The diagram illustrates the indexing of a 2D array. The expression **a[2][3]** is shown. A red arrow originates from the first index, **2**, and points to the text "Takes Element from the 3rd row". Another red arrow originates from the second index, **3**, and points to the text "Takes Element from the 4th column".


Accessing Multidimensional Array

Suppose there is a class:

```
Class c1{  
    int a[5][7];  
};
```

It has an Object:
c1 obj1;

obj1.a[2][3]



Takes Element from the 4th column



Takes Element from the 3rd row

Program to add two matrix

```
#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];
        }
}
```


Program to add two matrix

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

Program to add two matrix

```
// 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;
}
```

Program to add two matrix

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

String

C++ has in its definition a way to represent **sequence of characters as an object of class**. This class is called `std::string`. String class stores the characters as a sequence of bytes with a functionality of allowing **access to single byte character**.

String vs Char Array

- A character array is simply an array of characters can terminated by a null character. A string is a class which defines objects that be represented as stream of characters.
- Size of the character array has to allocated statically, more memory cannot be allocated at run time if required. Unused allocated memory is wasted in case of character array. In case of strings, memory is allocated dynamically. More memory can be allocated at run time on demand. As no memory is preallocated, no memory is wasted

String vs Char Array

- Implementation of character array is faster than `std::string`. Strings are slower when compared to implementation than character array.
- Character array do not offer much inbuilt functions to manipulate strings. String class defines a number of functionalities which allow manifold operations on strings.

Operation on String

- **getline()** :- This function is used to store a stream of characters as entered by the user in the object memory.
- **push_back()** :- This function is used to input a character at the end of the string.
- **pop_back()** :- Introduced from C++11(for strings), this function is used to delete the last character from the string

Operation on String

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str;
    getline(cin,str);
    cout << "The initial string is : ";
    cout << str << endl;
    str.push_back('s');
    cout << "The string after push_back operation is : ";
    cout << str << endl;
    str.pop_back();
```


Operation on String

```
cout << "The string after pop_back operation is : ";  
cout << str << endl;
```

```
return 0;
```

```
}
```

Input:-

LPU

Output:-

The initial string is : LPU

The string after push_back operation is : LPUs

The string after pop_back operation is : LPU

Operation on String

4. capacity() :- This function returns the capacity allocated to the string, which can be equal to or more than the size of the string. Additional space is allocated so that when the new characters are added to the string, the operations can be done efficiently.

5. resize() :- This function changes the size of string, the size can be increased or decreased.

6. size()

7.shrink_to_fit() :- This function decreases the capacity of the string and makes it equal to the minimum capacity of the string. This operation is useful to save additional memory if we are sure that no further addition of characters have to be made.

Operation on String

```
#include<iostream>
#include<string> // for string class
using namespace std;
int main()
{
    string str = "learn and grow and learn fast";

    cout << "The initial string is : ";
    cout << str << endl;
    str.resize(13);
    cout << "The string after resize operation is : ";
    cout << str << endl;
    cout << "The capacity of string is : ";
    cout << str.capacity() << endl;
    cout<<"The length of the string is :"<<str.length()<<endl;
```

Operation on String

```
str.shrink_to_fit();  
cout << "The new capacity after shrinking is : ";  
cout << str.capacity() << endl;  
  
return 0;  
  
}
```

Operation on String

- 8. **begin()** :- This function returns an iterator to beginning of the string.
- 9. **end()** :- This function returns an iterator to end of the string.
- 10. **rbegin()** :- This function returns a reverse iterator pointing at the end of string.
- 11. **rend()** :- This function returns a reverse iterator pointing at beginning of string.

Operation on String

```
#include<iostream>
#include<string> using namespace std;
int main()
{
    string str = "learnandgrow";
    std::string::iterator it;
    std::string::reverse_iterator it1;
    cout << "The string using forward iterators is : ";
    for (it=str.begin(); it!=str.end(); it++)
        cout << *it;
    cout << endl;
    cout << "The reverse string using reverse iterators is : ";
    for (it1=str.rbegin(); it1!=str.rend(); it1++)
        cout << *it1;
    cout << endl;
    return 0; }
```

Operation on String

12. copy("char array", len, pos) :- This function copies the substring in target character array mentioned in its arguments. It takes 3 arguments, target char array, length to be copied and starting position in string to start copying.

13. swap() :- This function swaps one string with other.

Operation on String

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str1 = "learn grow and explore";
    string str2 = "learn grow";
    char ch[80];
    str1.copy(ch,13,0);
    cout << "The new copied character array is : ";
    cout << ch << endl << endl;
    cout << "The 1st string before swapping is : ";
    cout << str1 << endl;
    cout << "The 2nd string before swapping is : ";
```


Operation on String

```
cout << str2 << endl;  
str1.swap(str2);  
cout << "The 1st string after swapping is : ";  
cout << str1 << endl;  
cout << "The 2nd string after swapping is : ";  
cout << str2 << endl;  
return 0;  
  
}
```

String Operation

- **find("string")**: Searches the string for the first occurrence of the substring specified in arguments. It returns the position of the first occurrence of substring.
- **find_first_of("string")**: Searches the string for the first character that matches any of the characters specified in its arguments. It returns the position of the first character that matches.
- **find_last_of("string")**: Searches the string for the last character that matches any of the characters specified in its arguments. It returns the position of the last character that matches.
- **rfind("string")**: Searches the string for the last occurrence of the substring specified in arguments. It returns the position of the last occurrence of substring

String Operation

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string str("Learn and Learn very fast");

    cout << "First occurrence of \"Learn\" starts from : ";
    cout << str.find("Learn") << endl;

    cout << "First occurrence of character from \"arn\" is at : ";
    cout << str.find_first_of("arn") << endl;
```

String Operation

```
cout << "Last occurrence of character from \"arn\" is at : ";  
cout << str.find_last_of("arn") << endl;  
  
cout << "Last occurrence of \"Learn\" starts from : ";  
cout << str.rfind("Learn") << endl;  
  
return 0;  
  
}
```

Practice Question

1. Write a C++ program to reverse a given string.

Example:

Sample Input: upgrade

Sample Output: edargup

2. Write a C++ program to capitalize the first letter of each word of a given string. Words must be separated by only one space.

Example:

Sample Input: learn and grow

Sample Output: Learn And Grow

3. Write a C++ program to count all the vowels in a given string.

Example:

Sample Input: eagerer

Sample output: number of vowels -> 4

Any Questions??

Thank You!

See you guys in next class.