Arrays

What is an Array?

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- An array is a special variable, which can hold more than one value at a time.

In Java:-

- To declare an array, define the variable type with **square brackets**.
- String[] cars = {"Volvo", "BMW", "Ford"};
- int[] myNum = {10, 20, 30, 40};

```
public class Array {
    Run | Debug
    public static void main(String[] args) {
        String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
        int[] myNum = {10, 20, 30, 40};
        System.out.println("Array is :- " + cars);
        System.out.println("Array is :- " + myNum);

}
}
```

Access the Elements of an Array

- Cars[0]
- myNum[0]

Python:-

- List, Tuple, Set and Dictionary…
- Initially we use list only…

```
cars = ["Volvo", "BMW", "Ford", "Mazda"]
myNum = [10, 20, 30, 40]

print("Array is :- ", cars)
print("Array is :- ", myNum)
```

Access the Elements of an Array

- Cars[0]
- myNum[0]

In C++:-

- 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.

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

int main() {
    string cars[] = {"Volvo", "BMW", "Ford", "Mazda"};
    int myNum[] = {10, 20, 30, 40};

    cout << "Array is :- " << cars << endl;
    cout << "Array is :- " << myNum << endl;

}
```

Access the Elements of an Array

- Cars[0]
- myNum[0]

Multidimensional Arrays

- A multidimensional array is an array of arrays.
- Multidimensional arrays are useful when you want to store data as a tabular form, like a table with rows and columns.

In Java:-

- To create a two-dimensional array, add each array within its own set of **curly braces**.

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
System.out.println("Two Dimension Array...... --> " + myNumbers[1][2]);
```

In Python:-

```
# Two Dimensional Array..

arr1 = [ [1, 2, 3, 4], [5, 6, 7]]

[2] arr1[1][2]

    7
```

In C++ :-

```
// Two Dimensional Array....
int mynumbers[][4] = { {1, 2, 3, 4}, {5, 6, 7} };
cout << "Two Dimensional Array --> " << mynumbers[1][2] << endl;
```

Array Methods

In Java (using ArrayList) import “java.util.ArrayList;”:-

- add(): Adds an element at the end of the list.
- clear(): Removes all the elements from the list.
- clone(): Returns a shallow copy of the list.
- contains(): Returns true if the list contains the specified element.
- addAll(): Adds all the elements of a list (or any iterable) to the end of the current list.
- indexOf(): Returns the index of the first occurrence of the specified element.
- add(index, element): Inserts the specified element at the specified position.
- remove(index): Removes the element at the specified position.
- remove(element): Removes the first occurrence of the specified element.
- Collections.reverse(): Reverses the order of the list.
- Collections.sort(): Sorts the list.

```java
public static void main(String[] args) {
    ArrayList<Integer> list1 = new ArrayList<Integer>();
    list1.add(5);
    System.out.println("First element: " + list1.getindex:(0));

}
```

In Python (using list):-

- append(): Adds an element at the end of the list.
- clear(): Removes all the elements from the list.
- copy(): Returns a copy of the list.
- count(): Returns the number of elements with the specified value.
- extend(): Adds the elements of a list (or any iterable) to the end of the current list.
- index(): Returns the index of the first occurrence of the specified element.
- insert(): Inserts the specified element at the specified position.
- pop(): Removes the element at the specified position.
- remove(): Removes the first occurrence of the specified element.
- reverse(): Reverses the order of the list.
- sort(): Sorts the list.

```
[3] # Declaring and initializing a list (Python's equivalent to arrays) of integers
    numbers = [1, 2, 3, 4, 5]
    numbers.append(6)
```

```
[4] numbers
```

```
    [1, 2, 3, 4, 5, 6]
```

In C++ (using vector):-
“#include <vector>”

- push_back(): Adds an element at the end of the vector.
- clear(): Removes all the elements from the vector.
- (Copy Constructor): Returns a copy of the vector.
- count(): Returns the number of elements with the specified value.
- insert(): Adds the elements of a vector (or any iterable) to the end of the current vector.
- find(): Returns an iterator to the first occurrence of the specified element.
- insert(): Inserts the specified element at the specified position.
- pop_back(): Removes the element at the end of the vector.
- erase(): Removes the element at the specified position or range.
- reverse(): Reverses the order of the elements in the vector.
- sort(): Sorts the elements in the vector.

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

int main()
{
    vector<int> vec;
    vec.push_back(95);

    cout << "First element: " << vec[0] << endl;

    return 0;
}
```

Task:--

1. Create Fruits Array with 10 number of fruits name, print all frits name without using loops.
2. Declare an array of 5 integers and print the first and last elements.
3. Create an array of 5 strings and modify the second element. Print the modified array.
4. Declare an array of 10 floating-point numbers and print the element at index 5.
5. Create an array of 4 strings and print the first and last elements.
6. Create an array of 6 floating-point/double numbers and modify the second element. Print the modified array.
7. Declare a list or vector of 5 integers and use a method to append two more integers at the end and also use a method to sort the elements in ascending order..
8. Create a list or vector of 5 strings and use a method to remove all the elements.
9. Create an array that stores integers and strings.
10. Declare an array that stores floating-point numbers and characters. Add two floating-point numbers and three characters to the array.
11. Create a mixed array that stores integers and boolean values. Add two integers and three boolean values to the array.
12. Declare an array that stores strings and boolean values. Add three strings and two boolean values to the array.
13. Create a mixed array that stores floating-point numbers and strings. Add two floating-point numbers and three strings to the array.
14. Declare a 2D array of integers with 3 rows and 4 columns. Initialize the array with values and print the element at the second row and third column.
15. Create a 3D array of floating-point numbers with dimensions 2x3x4. Initialize the array with values and print the element at the first layer, second row, and fourth column.
16. Declare a 2D array of strings with 4 rows and 5 columns. Initialize the array with values and print the element at the fourth row and first column.
17. Create a 3D array of strings with dimensions 2x2x3. Initialize the array with values and print the element at the second layer, first row, and third column.
18. Create a 2D array that stores integers and strings. Add three rows to the array, where each row contains two integers and one string.
19. Declare a 3D array that stores floating-point numbers and characters. Add two layers to the array, where each layer contains three rows and each row contains two floating-point numbers and one character.
20. Declare a 3D array that stores integers and strings. Add three layers to the array, where each layer contains two rows and each row contains one integer and two strings.
21. Create a 2D array that stores characters and floating-point numbers. Add five rows to the array, where each row contains two characters and one floating-point number.