



Arrays

+ Introduction to Arrays

- Arrays are defined as a finite ordered set of homogeneous elements.
- Finite means there is a specific number of elements in the array.
- Ordered means the elements of the array are arranged so that so that there is zeroth, first, second and so on elements.
- Homogeneous means all the elements in the array must be of the same type.
- E.g. An array may contain all integers or all characters but not integers and characters.
- Declaration in C++ `int a[100];`

+ Operations on Array

- The two basic operations that access an array are Extraction and Storing.
- The Extraction operation is a function that accepts an array a , and an index I , and returns an element of the array.

In C++ this operation is denoted by the expression $a[i];$

- The storing operation accepts an array a , an index I and an element x .

In C++ this operation is denoted by the expression

$$a[I] = x;$$

+ Arrays Continued...

- The smallest element of an array's index is called its lower bound. It is 0 in C++.
- The highest element is called its upper bound.
- If Lower bound is “lower” and Upper bound is “upper” then number of elements in the array, called its “range” is given by

$$\text{upper} - \text{lower} + 1.$$

E.g. In array a, the lower bound is 0 and the upper bound is 99, so the range is 100.

+ Arrays Continued..

- An important feature of array in C++ is that the upper bound nor the lower bound may be changed during the program execution.
- Arrays will always be stored in contiguous memory locations.
- You can have static as well as dynamic arrays, where the range of the array will be decided at run time.

+ Single Dimension Array Implementation

```
Int arr[10], i;
```

```
for( i = 0; i < 10; i++ )
```

```
    arr[i] = i;
```

```
for ( i = 0; i < 10; i++ )
```

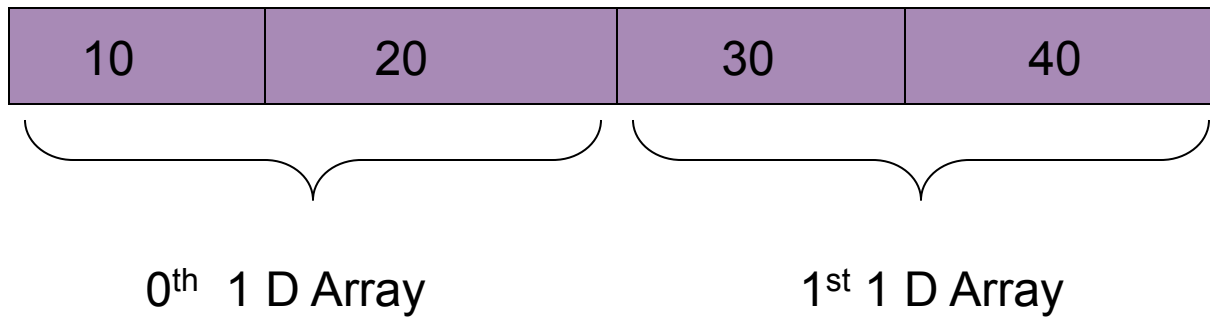
```
    cout<< i ;
```

+ Two Dimensional Array

- Two dimensional array can be considered as an array of 1-D array.
- Declaration `int arr[2][2];`
- This defines a new array containing two elements. Each of these elements is itself an array containing 2 integers.
- Each element of this array is accessed using two indices: row number and the column number.

+ 2-D Array Representation

■ Physical:



□ Logical:

	C 0	C 1
R0	10	20
R1	30	40

+ Using 2D Array

```
Int arr[2][2], i , j;
```

```
For( i = 0; i < 2; i ++ )
```

```
    for( j = 0; j < 2; j++ )
```

```
        arr[ i ][ j ] = i + j;
```

```
For( i = 0; i < 2; i ++ )
```

```
    for( j = 0; j < 2; j++ )
```

```
        cout<< arr[ i ][ j ];
```

+ Advantages of Arrays

- Simple and easy to understand.
- Contiguous allocation.
- Fast retrieval because of indexed nature.
- No need for the user to be worried about allocation and de allocation of arrays.

+ Disadvantages of Array

- If you need m elements out of n locations defined. $n-m$ locations are unnecessarily wasted if $n > m$
- You can not have more than n elements. (I.e static allocation of memory.) -
- large number of data movements in case of insertion & deletion, which leads to more overheads.

+ Disadvantages of Array

- If you need m elements out of n locations defined. $n-m$ locations are unnecessarily wasted if $n > m$
- You can not have more than n elements.
(I.e static allocation of memory.) -
- large number of data movements in case of insertion & deletion, which leads to more overheads.