# PowerPoint presentation on …

# Array

This PPT will helpful for your project and self study also.

# What is Arrays ?

- An array is a group of consective memory locations with same name and data type.
- Simple variable is a single memory location with unique name and a type. But an Array is collection of different adjacent memory locations. All these memory locations have one collective name and type.
- The memory locations in the array are known as **elements** of array. The total number of elements in the array is called **length.**
- The elements of array is accessed with reference to its position in array, that is call **index** or **subscript.**

# Advantages / Uses of Arrays

- Arrays can store a large number of value with single name.
- Arrays are used to process many value easily and quickly.
- The values stored in an array can be sorted easily.
- The search process can be applied on arrays easily.

## Types of Arrays:

- One-Dimensional Array
- Two-Dimensional Array
- Multi-Dimensional Array

# One-D Array

A type of array in which all elements are arranged in the form of a list is known as **1-D array** or **single dimensional array** or **linear list.**

**Declaring 1-D Array:**

data_type identifier[length]; e.g: int marks[5];

- **Data _type:** Data type of values to be stored in the array.
- **Identifier:** Name of the array.
- **Length:** Number of elements.

0 1 2 3 4

int marks

# One-D array Intialization

The process of assigning values to array elements at the time of array declaration is called **array initialization.**

**Syntax:**

data_type identifier[length]={ List of values };

**e.g:** int marks[5]={70,54,82,96,49};

| 70 | 54 | 82 | 96 | 49 |
|---|---|---|---|---|

- **Data _type:** Data type of values to be stored in the array.
- **Identifier:** Name of the array.
- **Length:** Number of elements
- **List of values:** Values to initialize the array. Initializing values must be constant

# Accessing element of array

- **Individual Element:**

```
Array_name[index];
```

- **Using Loop:**

```
int marks[5];
for(int i=0;i<=4;i++)
marks[i]=i;
```

# Searching In Array

Searching is a process of finding the required data in the array. Searching becomes more important when the length of the array is very large.

There are two techniques to searching elements in array as follows:

- Sequential search
- Binary search

# Sequential Search

Sequential search is also known as **linear** or **serial search**. It follows the following step to search a value in array.

- Visit the first element of array and compare its value with required value.
- If the value of array matches with the desired value, the search is complete.
- If the value of array does not match, move to next element an repeat same process.

# Binary Search

Binary search is a quicker method of searching for value in the array. Binary search is very quick but it can only search an sorted array. It cannot be applied on an unsorted array.

- It locates the middle element of array and compare with desired number.
- If they are equal, search is successful and the index of middle element is returned.
- If they are not equal, it reduces the search to half of the array.
- If the search number is less than the middle element, it searches the first half of array.

Otherwise it searches the second half of the array. The process continues until the required number is found or loop completes without successful search.

# Sorting Arrays

Sorting is a process of arranging the value of array in  a particular order.
An array can be sorted in two order.

- Ascending Order
- Descending Order

| 12 | 25 | 33 | 37 | 48 |
|---|---|---|---|---|

| 48 | 37 | 33 | 25 | 12 |
|---|---|---|---|---|

# Techniques Of Sorting Array

There are two techniques of sorting array:

- Selection Sort
- Bubble Sort

# Selection Sort

Selection sort is a technique that sort an array. It selects an element in the array and moves it to its proper position. Selection sort works as follows:

1. Find the minimum value in the list.
2. Swap it with value in the first position.
3. Sort the remainder of the list excluding the first value.

# Bubble Sort

Bubble Sort is also known as **exchange sort.** It repeatedly visits the array and compares two items at a time. It works as follows:

- Compare adjacent element. If the first is greater than the second, swap them.
- Repeat this for each pair of adjacent element, starting with the first two and ending with the last two. (at this point last element should be greatest).
- Repeat the step for all elements except the last one.
- Keep repeating for one fewer element each time until there are no pairs to compare.

# Two-D Arrays

Two-D array can be considered as table that consists of rows and columns. Each element in 2-D array is refered with the help of two indexes. One index indicates row and second indicates the column.

**Declaring 2-D Array:**

Data_type Identifier[row][column];

e.g: int arr[4][3];

- **Data _type:** Data type of values to be stored in the array.
- **Identifier:** Name of the array.
- **Rows :** # of Rows in the table of array.
- **Column :** # of Columns in the table of array.

| | | |
|---|---|---|
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |
| 3,0 | 3,1 | 3,2 |

# Two-D array Intialization

The two-D array can also be initialized at the time of declaration. Initialization is performed by assigning the initial values in braces seperated by commas.

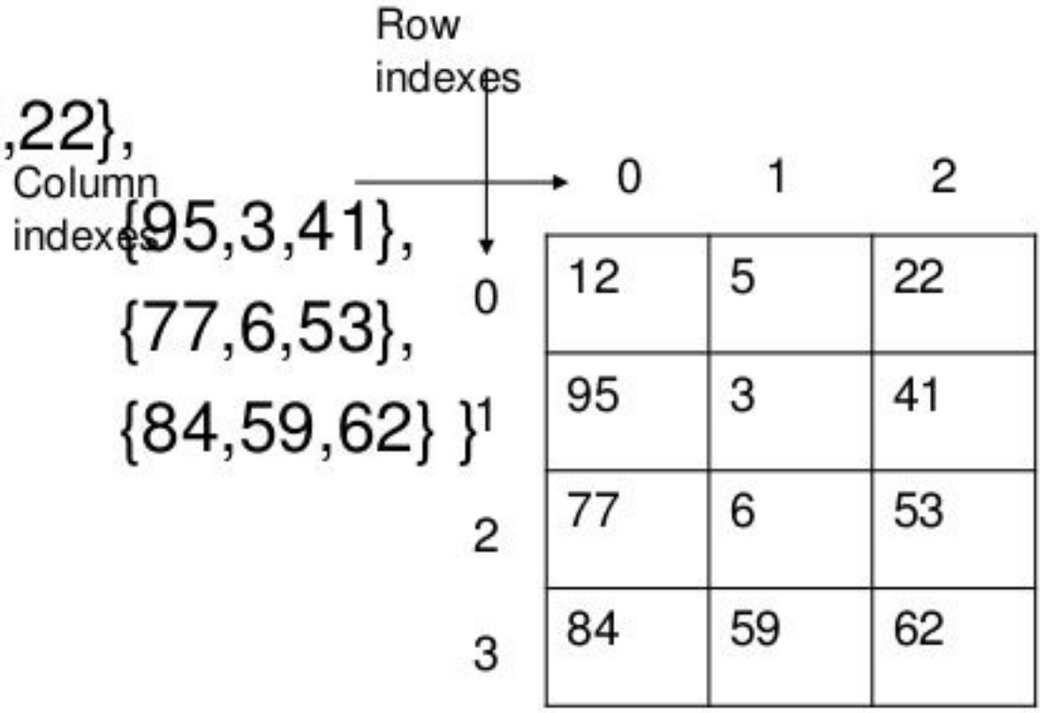
Some important points :

- The elements of each row are enclosed within braces and seperated by comma.
- All rows are enclosed within braces.
- For number arrays, if all elements are not specified , the un specified elements are initialized by zero.

# Two-D array Intialization

**Syntax:**

```
int arr[4][3]={ {12,5,22},
                {95,3,41},
                {77,6,53},
                {84,59,62} };
```



# Thank You...

* Prepared by .. .

1. Kaushal Mehta … .. twitter@kaushalmehta96
2. Jigar Patel … .. twitter@jigarpatel_jp

#KaushalJigar