04/01/25

Arrays

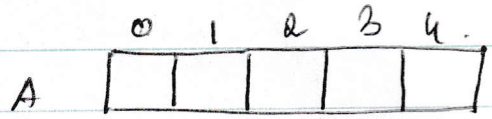
① What is an array.
② Declaring an array. and initialising
③ Accessing the elements of an array.

→ Array is a collection of similar data elements groups under same name.

* Arrays are vector variable.
* variables are scalar variable.

int n = 10 → scalar, 10 is the magnitude of n

int Array[5] = {. . .} vector.



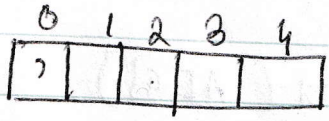
So all the elements in this array has the same ~~element~~ variable name A. how can we differentiate them. In this case under came into picture. we can call them

Array of index 0. A[0] etc and so on.

So using the name and index we can access the elements of an array.

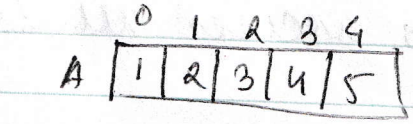
declaring and initializing an array.

int A[5];

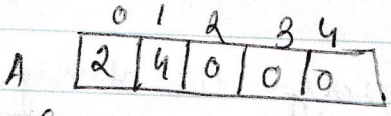


garbage value as we are not initializing with elements.

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

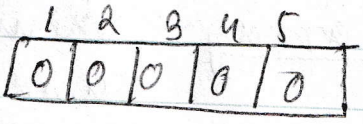


int A[5] = {2, 4};



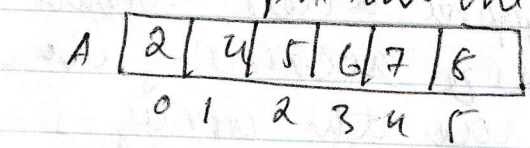
when we declared the array with a size of 5 and the initializing with only two elements so the rest of the elements will be initialized with zero. (0

int A[5] = {0}



same as the previous one.

int A[6] = {2, 4, 5, 6, 7, 8}



↓

if we declare array like this in the program.

then this is how it will be represented in main memory.

Accessing the elements in an array.

A [5] = {1, 2, 3, 4, 5}

print (A[0])

Traversing through an array means accessing all the elements in an array. We can use a for loop to access all the elements in an array

Static array and dynamic array.

* Static array - memory is allocated at compile time having a fixed size. We cannot alter or update the size of the array.

int A[5] = {. . . }

* Dynamic array: Memory is allocated at run time but not having fixed size.

Suppose a user wants to allocate memory size according to his/her need, in this case the array is dynamic.

int A[] = { }. .. dynamic.

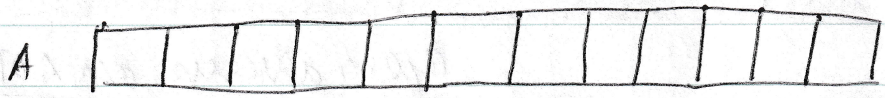
2D Array

int A[3][4] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}}

A	0	1	2	3
	1	2	3	4
	5	6	7	8
	9	10	11	12

it will have 12 elements in memory like a one D arra

- memory representation.



When we need to store data in a tabular form then we might need to use multidimensional array. (2D array)

multi demsional array basically means array of arrays.

for accessing elements in two D array we need to use nested loop. (for loop)

10/02/24

Array ADT.

→ Abstract data type - Representation of data and operations on that data.

Array is a basic data structure provided by almost every ~~data structure~~ programming language. And its representation is done by the compiler. But operations are user defined.

Representations

① Array Space
② Size
③ Length (no of elements)

Operations on ADT

① Display
② Add (n) / append (n)
③ Insert (index x)
④ Delete (index)
⑤ Search (x)
⑥ Get (index)
⑦ Set (index, x)
⑧ Max () / Min()
⑨ Reverse ()
⑩ Shift () / Rotate ()

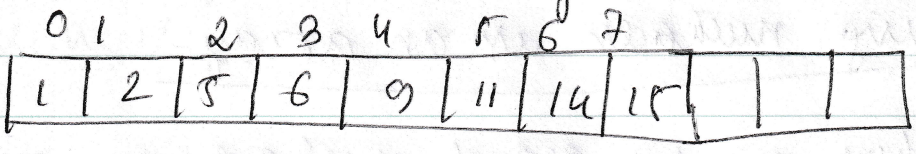
① Display - Traversing through the array using a for loop.

② Add (n) / Append (n) → It means adding an array ~~with~~ in the last free stage of an array.

Time complexity O(1)
Constant

Size of an array is the total size,
length of an array is how many elements are present

Insert → Insert operation takes, index and elements. This means inserting an element in a given index.



Suppose we want to insert elements at index 3, where 6 is present, then we need to shift the elements to ~~next~~ right side.

```
for (i = length ; i > ~~length~~ index ; i--)
    A[i] = A[i-1]
}
```

Time complexity - Two operations -

→ Shifting — O(n)
→ Inserting — O(1)

min O(1)
max (O(n))

— o —

Delete element

operation to performed when deleting from array with a specific index.

— deleting the element - arr[2] — 1
→ Shifting the elements of ~~to~~ - include element — 0-n
→ decreasing the length of the array — 1

Best case time of deleting an element - O(1)
Worst case time (O(n))

Searching methods in an array -

→ There are two search methods in array

① Linear search
② Binary search

A	8	9	4	7	6	3	10	5	14	2
	0	1	2	3	4	5	6	7	8	9

For searching the element should be unique there should not be any duplicate elements.

Usually the result of a search is the location where the element is found, we return the index.

when we are writing we need to take care of two things.

Successful search - when the element is found

unsuccessful - when the element is not found.

→ Linear Search →

Traversing through the array and find if the key element is present.

Time complexity -

for successful search → { min O(1) - best
max O(n) - worst

unsuccessful search O(n) (maximum)

Improvement in Linear Search -

① Transposition
② Move to front / head.