Review on a Linear Data Structure-Array

Gausiya Yasmeen*
Department of Computer Applications, Integral University, Lucknow, India.

****Corresponding Author***
E-mail Id:*-gausiyay@iul.ac.in*

ABSTRACT
Arrays are the simplest data structure used in any programming language. It is used to store element which belongs to same data type. Arrays signify the generic collection or list of data elements. Array uses subscript variable to denote its elements. The name of the array is its base address with the help of which rest of the memory address can be figured out. They are used to build complex data structure. In the following paper I have pen down the basic characteristics of array, its memory representation and calculation of address.

Keywords:*-Arrays, abstract data type (ADT), pointers*

INTRODUCTION

The array is an abstract data type (ADT) that holds a collection of elements accessible by an index. The elements stored in an array can be anything from primitives' types such as integers to more complex types like instances of classes. While writing program code in any language we need to store data. For e.g. we need to store marks of A student. Hence taking different variables for each mark can cause too much handling of variables. In such case Arrays are the most suitable collection to store similar data under same name. Thus, arrays are a list of similar type of items stored at contiguous memory locations. Similar type refers to the data type to which particular item belongs. Data type can be are…. etc. If we create a collection of Cities then we need to store only cities in that collection otherwise the data will be invalid.

OPERATIONS ON ARRAY

Traverse- Accessing of data elements in an array. In traversing we process every element of array for once from either first to last or vice versa. It is also called visiting array members.

Insertion- Inserting an element in the array. Elements are inserted from the first index moving towards end of the array. In case of sorted array element if inserted according to its value. If it is ascending order then elements with small value will be placed in the beginning else at the end of the array.

Deletion- Remove the data stored at particular index.

Search- Searching of an element can be performed on array elements. Linear and binary search techniques are used to search an element in the array.

Sorting- Array elements can be sorted either in ascending or descending order.

CLASSIFICATION OF ARRAY

Arrays can be of following types:

One dimensional (1-D) arrays or Linear arrays:

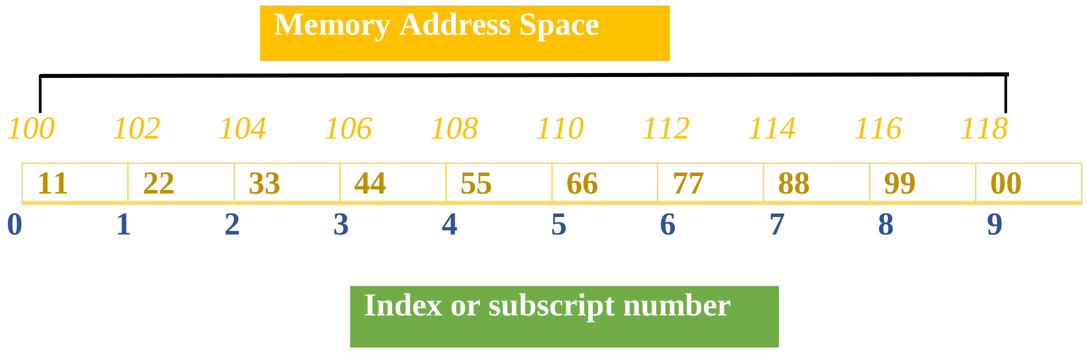
1-D array has only single row without any columns. It uses one subscript to represent the elements. Elements are arranged serially.

Multi-dimensional arrays:

Two dimensional (2-D) arrays or Matrix arrays: 2-D arrays use two subscripts to represent the elements. They have rows and columns-mxn. One row will have one column only. If first row has three columns then all the rest of the rows in 2-D array will have three columns only. E.g., Matrix, Sparse Matrix

Three dimensional arrays:
3-D arrays have row, columns and height. It uses three subscripts to represent the elements-mxnxh.

MEMORY REPRESENTATION
Array can be termed as a collection of homogeneous data elements which stores its elements in contiguous memory allocation manner. Each element is accessed and stored by using index number or subscript value.

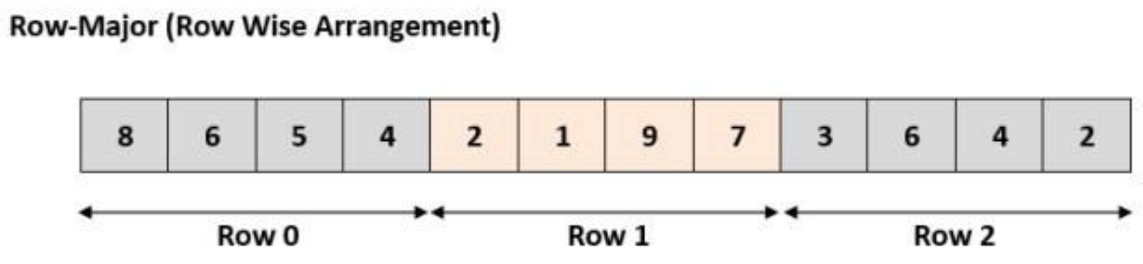


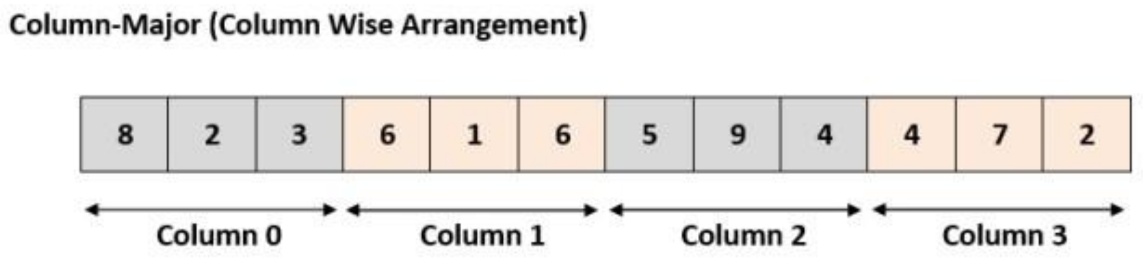
Here is a diagram representing the contiguous allocation of array in the memory.0,1…9 is the index or subscript value through which each element can be accessed separately. 100,102…118 are the memory blocks where data is stored. Here I have taken integer data type which occupies 2 bytes of memory in C language. Memory is allocated in blocks and continuous format. These blocks are data type dependent. Array elements share the same name; so, in order to access any element of array index numbers are used. Like element 55 can be addressed as ar [5] =55.Arrays subscripts starts from 0th index number and size-1 will be the last index number. So, array is declared in the program with its size. Arrays can never store element equal to its size. If we say 10 elements in an array then it means from 0th to 9(10-1)th position. Array will not store element at 10th index number.

Address Calculation in single (one) Dimension Array- By using the given formula we can calculate the address of any index number.
Address of A [I] = B + W * (I – LB)
B- base address of array
W- Storage Size of one element stored in the array (in byte according to data type)
I = Subscript of element whose address is to be calculated
LB = Lower limit / Lower Bound of subscript, if not specified assume 0 (zero)
Taking the example of array given above, suppose we need to find the address of ar[5].
B=100,W=2 bytes,I=5,LB=0
Address of A [5]= 100+2*(5-0)
=110 [Ans]
Address Calculation in Double (Two) Dimensional Array-

	0	*1*	*2*
0	2	3	4
1	5	6	7
2	8	9	1

2-D array is also called matrix. It’s an image of 3x3 matrix having 3 rows and 3 columns. But in memory 2-day is also stored in contiguous fashion which can be either Row-Major and Column-Major format.





Row Major System:

The address of a location in Row Major System is calculated using the following formula:

Address of A [I][J] = B + W * [N * (I – Lr) + (J – Lc)]

Column Major System:
The address of a location in Column Major System is calculated using the following formula:
Address of A [I][J] Column Major Wise = B + W * [(I – Lr) + M * (J – Lc)]

Where,
B = Base address
I = Row subscript of element whose address is to be found
J = Column subscript of element whose address is to be found

W = Storage Size of one element stored in the array (in byte)
Lr = Lower limit of row/start row index of matrix, if not given assume 0 (zero)
Lc = Lower limit of column/start column index of matrix, if not given assume 0 (zero)
M = Number of rows of the given matrix
N = Number of columns of the given matrix
Number of rows (M) will be calculated as = (Ur – Lr) + 1
Number of columns (N) will be calculated as = (Uc – Lc) + 1

Examples:

Q 1. An array is [-15………10, 15……………40] requires one byte of storage. If beginning location is 1500 determine the location of X [15][20].

Solution:
Number or rows say M = (Ur – Lr) + 1 = [10 – (- 15)] +1 = 26
Number or columns say N = (Uc – Lc) + 1 = [40 – 15)] +1 = 26
(i) Column Major Wise Calculation of above equation
The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, M = 26

Address of A [I][J] = B + W * [(I – Lr) + M * (J – Lc)]
= 1500 + 1 * [(15 – (-15)) + 26 * (20 – 15)] = 1500 + 1 * [30 + 26 * 5] = 1500 + 1 * [160] = 1660 [Ans]

(ii) Row Major Wise Calculation of above equation
The given values are: B = 1500, W = 1 byte, I = 15, J = 20, Lr = -15, Lc = 15, N = 26

Address of A [I][J] = B + W * [N * (I – Lr) + (J – Lc)]
= 1500 + 1* [26 * (15 – (-15))) + (20 – 15)] = 1500 + 1 * [26 * 30 + 5] = 1500 + 1 * [780 + 5] = 1500 + 785
= 2285 [Ans]

ADVANTAGES AND DISADVANTAGES OF ARRAYS

Advantages

1. Array is the simplest and efficient data structure. It has O (1) time complexity in both best and worst case.
2. Any array element can be accessed directly by using its index number.
3. With array complex data structure like Linked O (can be possible.
4. Insertion and deletion are comparatively easy as no overheads of pointers is involved.

Disadvantages

1. It uses compile time memory allocation. So, its size has to be declared in the program.
2. If we need to add more elements than array size for code has to be modified.
3. When a smaller number of elements are stored in the array then the declared size, unused memory is wasted.
4. No actual deletion is performed in case of array. It overwrites the data element and memory occupied by the deleted element remains with the array.

CONCLUSION

Hence, we observed that due to fixed size memory allocation its implementation is hindered. Array can be implemented using pointers. Jagged arrays are also used in some programming languages. The concept of jagged array gives user to create dynamic size columns in each row.

Jagged Arrays are 2-D arrays. Arrays can be sorted using different types of sorting like bubble, heap, shell, radix, insertion and quick sort techniques.

REFERENCES

1. Balagurusamy, E. (1992). programming in ANSI C. Tata McGraw-Hill Education.
2. Kanetkar, Y. (1999). Let us C, BPB Pub. New Delhi.
3. Kernighan, B. W., & Ritchie, D. M. (2006). The C programming language.
4. Schildt, H. C. (1987). ö The Complete Reference (AI-based problem solving). London: Osborne-McGraw Hill, 645-648.
5. S. Srivastava, C - In Depth - 2Nd Revised Edition, 30 June 2009.
6. K. Pimparkhede, Computer Programming with C++, Cambridge University Press,, 16-Jan-2017 .
7. L. Uhr, Algorithm-Structured Computer Arrays and Networks, Academic Press, 10-May-2014.