Data Structure
Chapter 1:Arrays

GATE Computer Science Lectures

By

Monalisa Pradhan

- **Section 4: Programming and Data Structures**
  Programming in C. Recursion.Arrays, stacks, queues, linked lists, trees, binary search trees, binary heaps, graphs.

- Chapter 1:Arrays(Array using pointer ,1D and 2D,Array Location in Memory)

- Chapter 2: stacks, queues

- Chapter 3: linked lists

- Chapter 4: trees, binary search trees, binary heaps

- Chapter 5: graphs

- **Data Structure** :A logical or Mathematical model.

- **Array** :Collection of similar type of data

- 1D Array can be represented by a[i]=*(a+i)=*(i+a)=i[a]

- L_0=Starting location , b=base ,u=Upper Bound , c=Count or size

- Loc a[i]=L_0 +(i-b)×c if b=0 then Loc a[i]= L_0 +i×c

- Ex 1: Calculate the location of a[0] element in the array of a[-5…..5] of element. where the starting location is 1000 and each element occupy 2 memory cell.

- Sol: a[0]=1000+(0- (-5)) ×2 =1000+10=1010

Loc	1000	1002	1004	1006	1008	1010	1012	1014	1016	1018	1020
Add	a[-5]	a[-4]	a[-3]	a[-2]	a[-1]	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]

- a[i]=a+i*c   [a always point to starting location]

- Let find a[1] if a loc=10

- Char   10+1*1=11,   Int 10+1*2=12 , Float   10+1*4=14,

- Double 10+1*8=18,   Long double   10+1*10=20

Array	Pointer
1.User friendly operator . a[i] 2.Array is a constant. a+1 (correct) , a=a+1(Wrong) [Read only] a=&a 3.It hold only base address 4.Static 5.Early Binding 6.Compile Time Binding	1.Machine friendly operator. *(a+i) 2.Pointer is a variable p+1 (Correct) ,p =p+1(Correct) p≠ &p 3.It can hold any address 4.Dynamic 5.Late Binding 6.Run time Binding

- **2D Array**
- A[i][j] 1st subscript is row and 2nd is column
- a[i][j] =T[j] =*(T+j) =*(a[i] +j) =*(*(a+i)+j)   [where T=a[i]]
- **a   =   a[0][0] = 10
- **a+1 = a[0][0]+1 = 10+1=11
- **(a+1) = *a[1]=*(a[1]+0)=a[1][0] = 16
- *(*a+1) = a[0][1] = 12
- a[1][2] =*(*(a+1)+2)=20

a[2][3]					
0 rows	a_{00}	10	a_{01}	12	a_{02}	14
1 row	a_{10}	16	a_{11}	18	a_{12}	20

- 2D array []\[\]=*[]=[]*=**
- a $[b_1.......u_1, b_2.......u_2]$, b=base ,u=upper
- row=u_1-b_1+1 ,column=u_2-b_2+1
- Find location in row major order
- Loc a[i][j]=L_0 +[(i-b_1) ×(u_2-b_2+1) +(j- b_2)] ×c
- If base is 0 then Loc a[i][j]=L_0 +[i ×column +j] ×c
- Find location in column major order
- Loc a[i][j]=L_0 +[(j-b_2) ×(u_1-b_1+1) +(i- b_1)] ×c
- If base is 0 then Loc a[i][j]=L_0 +[j ×row +i] ×c
- **Note****
- Array index always start from 0.
- All value of array store in continuous location .
- Array name always contain base address .
- Array name itself a pointer that point to base address .
- 2D array is array of array or matrix.
- 3D array and more are called multidimensional array