K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

Course: Data Structure

(Course Code : ENCS205)

UNIT-1: Foundations of Data Structures

School of Engineering & Technology
K.R. Mangalam University, Gurugram (Haryana)

Session 12:

Memory Representation of Arrays

Learning Objectives

By the end of this session, will be able to:

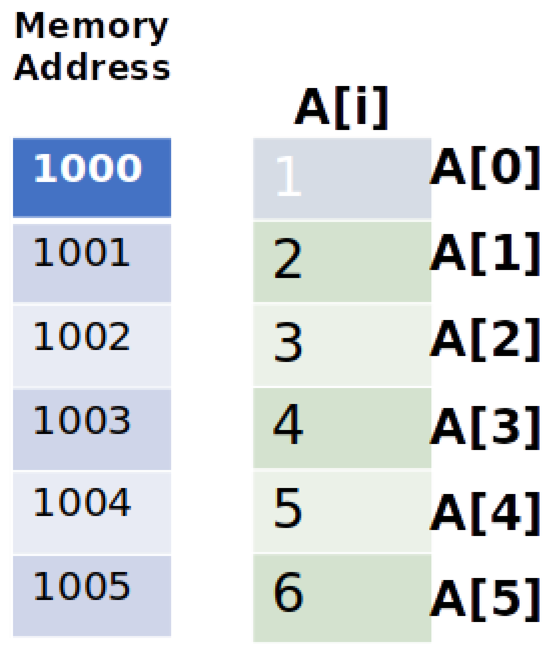
- Understand memory representation of linear and 2D arrays
- Calculate memory address using base address and index formulas
- Differentiate between row-major and column-major order
- Apply address calculation formulas for 2D arrays
- Solve numerical problems on array memory layout

Representation of Linear Array in Memory

- Let A be a linear array in the memory of the computer

- LOC(A[i]) = Memory address of the element A[i] of the array A

- The element of A are stored in the successive memory cells
- Computer does not need to keep track of the address of every element of A, but need to track only the address of the first element of the array denoted by Base(A) called the base address of LA



Computer Memory

Representation of Linear Array in Memory

LOC(A[i]) = Base(A) + w(i – lower bound)

- where w is the number of words per memory cell of the array A [w is aka size of the data type]

Example

Find the address for LA[6]
Each element of the array occupy 1 byte

200		LA[1]
201		LA[2]
202		LA[3]
203		LA[4]
204		LA[5]
205		LA[6]
206		LA[7]
207		LA[8]

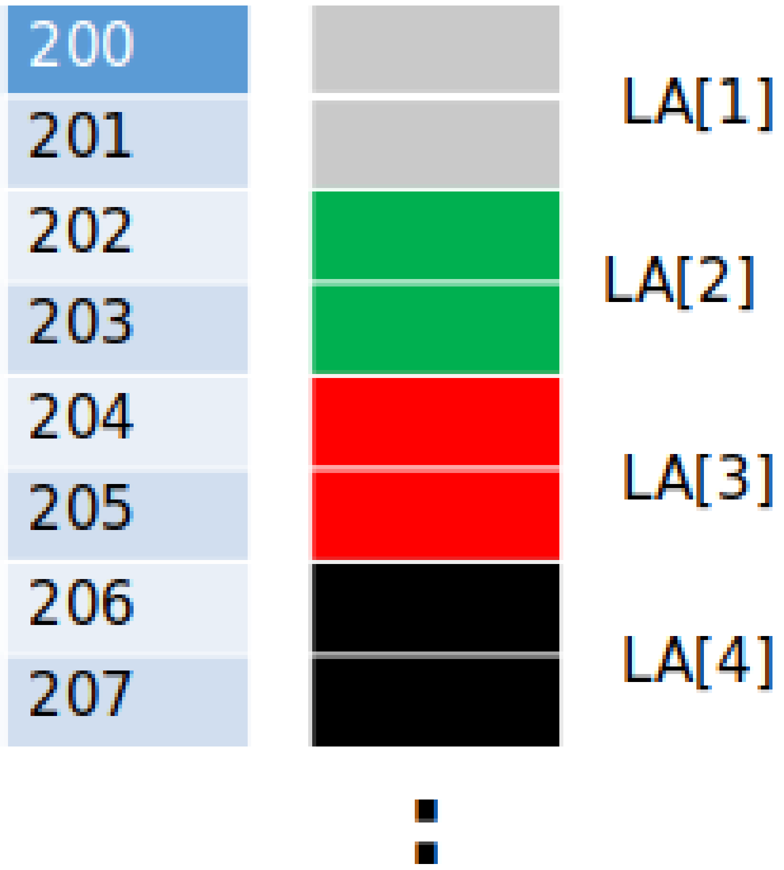
LOC(LA[i]) = Base(LA) + w(i – lower bound)

LOC(LA[6]) = 200 + 1(6 – 1) = 205

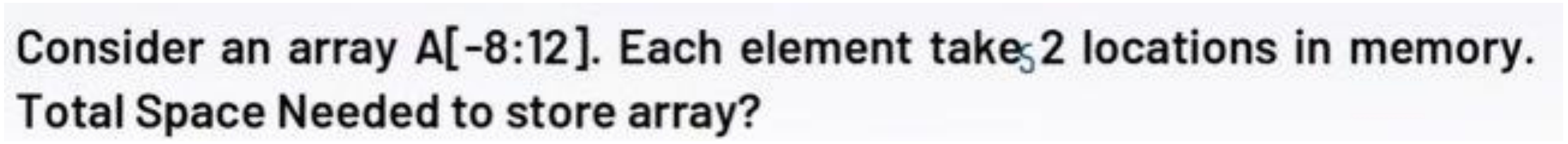
Find the address for LA[16]
Each element of the array occupy 2 byte

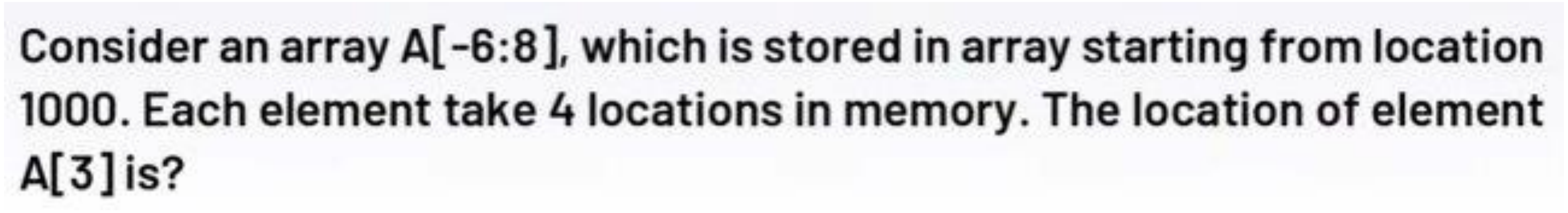
LOC(LA[i]) = Base(LA) + w(i – lower bound)

LOC(LA[16]) = 200 + 2(16 – 1) = 230



Address Calculation


Consider an array A[-8:12]. Each element takes 2 locations in memory. Total Space Needed to store array?


Consider an array A[-6:8], which is stored in array starting from location 1000. Each element take 4 locations in memory. The location of element A[3] is?

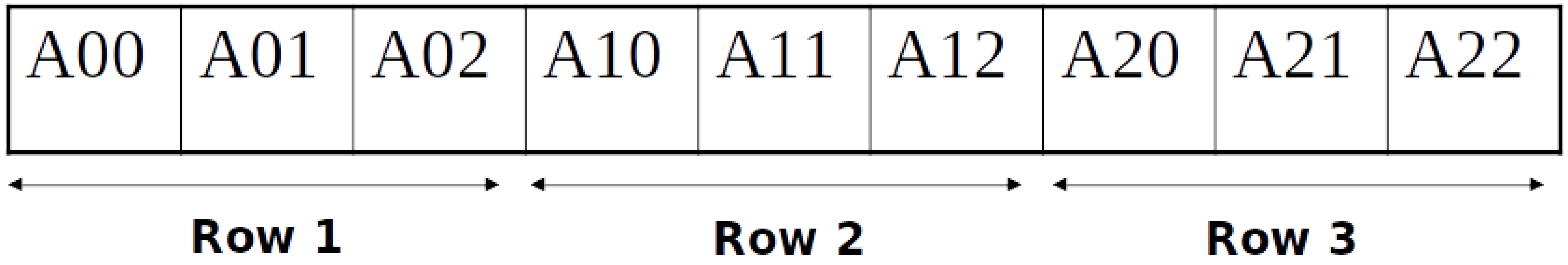
Implementation of Two-Dimensional Array in memory

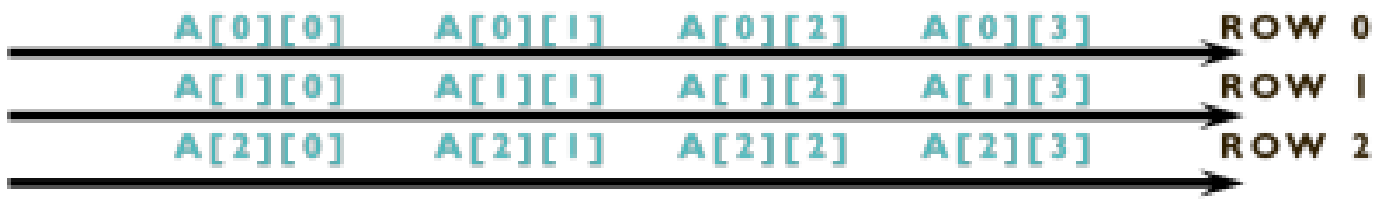
A two-dimensional array can be implemented in a programming language in two ways:

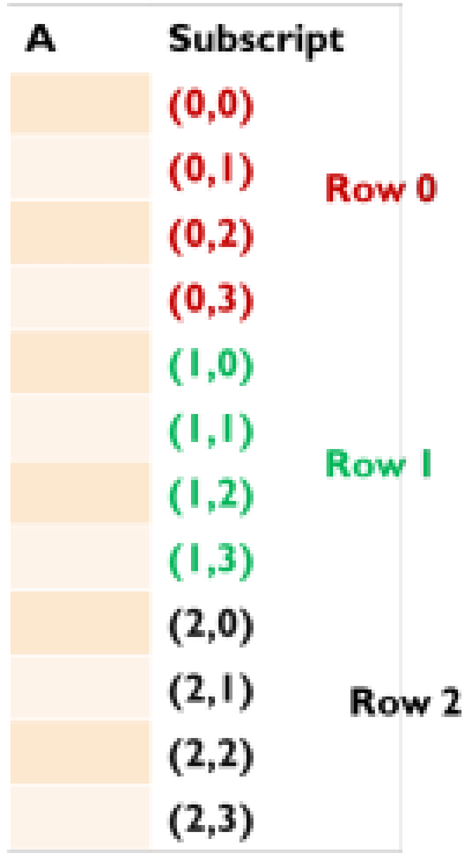
1. Row-major implementation
2. Column-major implementation

2D Arrays: Row-major Implementation

- Row-major implementation is a linearization technique in which elements of array are read from the keyboard row-wise that means the complete first row is stored then the complete second row is stored and so on.
- For example, an array [3][3] is stored in the memory as show bellow:




A[0][0] A[0][1] A[0][2] A[0][3]
ROW 0
A[1][0] A[1][1] A[1][2] A[1][3]
ROW 1
A[2][0] A[2][1] A[2][2] A[2][3]
ROW 2

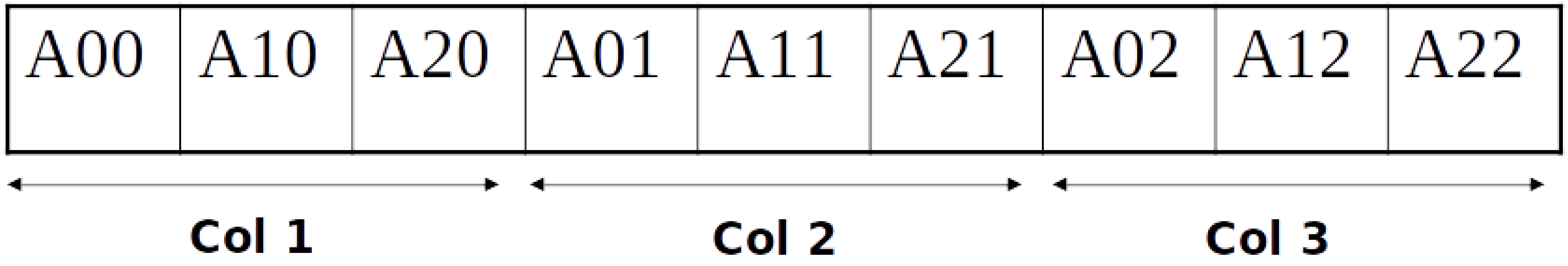


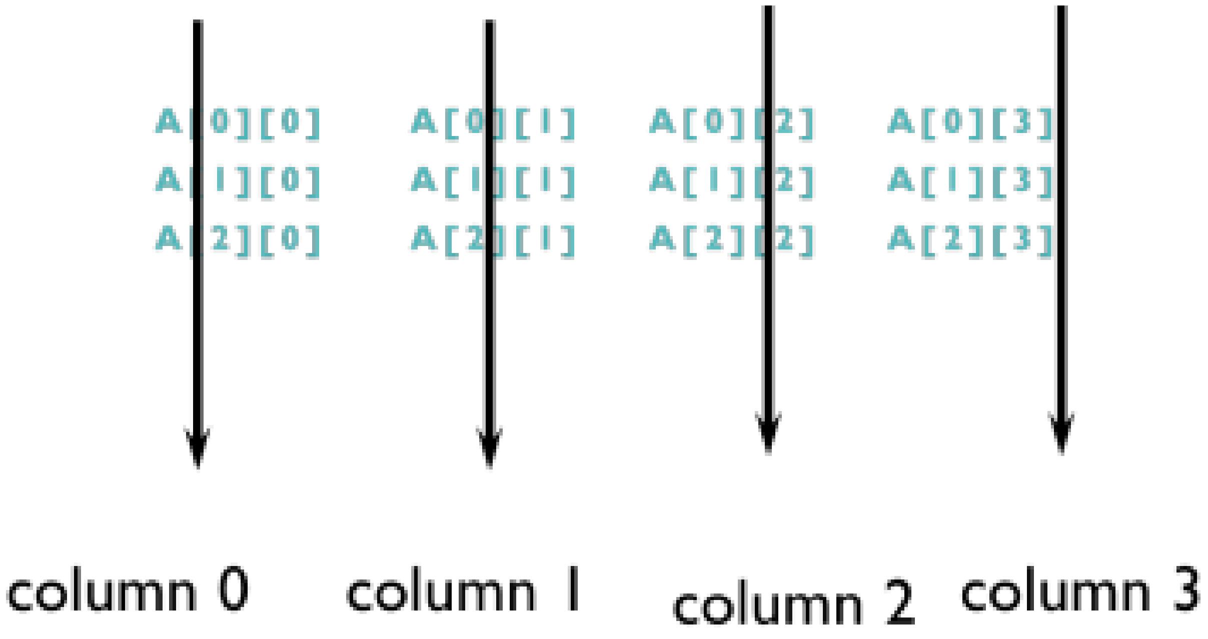
A	Subscript	
	(0,0)	
	(0,1)	Row 0
	(0,2)	
	(0,3)	
	(1,0)	
	(1,1)	Row 1
	(1,2)	
	(1,3)	
	(2,0)	
	(2,1)	Row 2
	(2,2)	
	(2,3)	

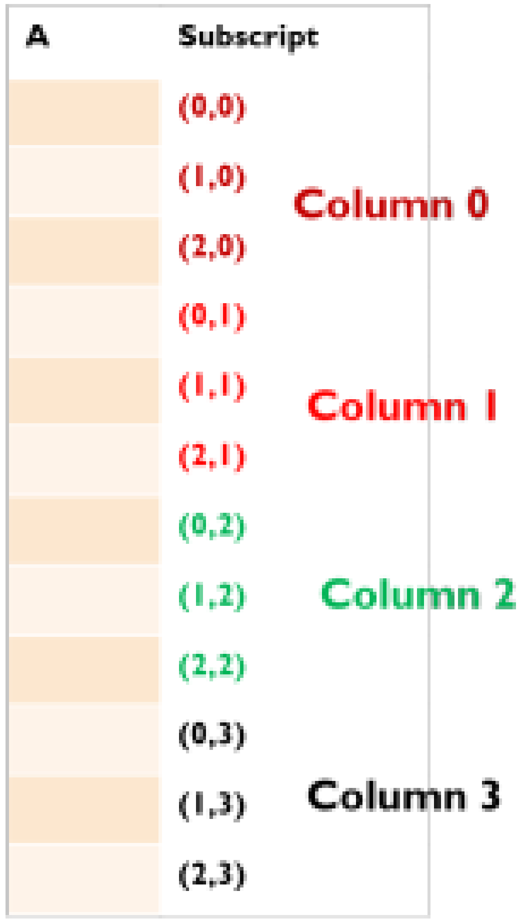
Row-Major Order

2D Arrays: Column-major Implementation

- In column major implementation memory allocation is done column by column that means first the elements of the complete first column is stored then elements of complete second column is stored and so on.
- For example, an array [3][3] is stored in the memory as shown below:







A	Subscript	
	(0,0)	Column 0
	(1,0)	
	(2,0)	
	(0,1)	Column 1
	(1,1)	
	(2,1)	
	(0,2)	Column 2
	(1,2)	
	(2,2)	
	(0,3)	Column 3
	(1,3)	
	(2,3)	

Column-Major Order

Address of elements in row major implementation

- The computer does not keep the track of all elements of the array, rather it keeps a base address and calculates the address of required element when needed.

- It calculates by the following relation:

address of element

LOC(A[i,j]) = Base(A) + w[n(i-LB1) + (j-LB2)]

m (number of rows) =UB1-LB1+1
n (number of columns) = UB2-LB2+1

- Consider a 25 x 4 array A. Suppose the Base(A) = 200 and w =4. Suppose the programming store 2D array using row-major. Compute LOC(A[12,3]), Considering the first index of A[1,1].

  - LOC(A[i,j]) = Base(A) + w[n(i-1) + (j-1)]

    LOC(A[12,3]) = 200 + 4[4(12-1) + (3 -1)]

    = 384

Address of elements in column major implementation

- It calculates by the following relation:

$$LOC(A[i,j]) = Base(A) + w[m(j-LB2) + (i-LB1)]$$

- m (number of rows) =UB1-LB1+1
- n (number of columns) = UB2-LB2+1

- A two-dimensional array defined as a[-20:20,10:35] requires one bytes of storage space for each element. If the array is stored in column-major form , then calculate the address of element at location a[0,30] given base address is 500.

- LOC(A[i,j]) = Base(A) + w[m(j-LB2) + (i-LB1)]

  = 500+1(41(30-10)+(0-(-20))

Summary of Session 12

- Linear arrays are stored in **contiguous memory** locations
- Only the **base address** is stored; other addresses are calculated as needed
- Address of 1D element: `LOC(A[i]) = Base(A) + w(i – LB)`
- 2D arrays can be stored in **row-major** or **column-major** order
- Row-major address formula:
  `LOC(A[i,j]) = Base(A) + w[n(i – LB1) + (j – LB2)]`
- Column-major address formula:
  `LOC(A[i,j]) = Base(A) + w[m(j – LB2) + (i – LB1)]`
- Address calculations depend on base address, element size (w), and array bounds
- Numerical problems help reinforce understanding of memory layout

Thank You

K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

Recognised under the section 2 (f) of the UGC Act 1956

Empowering the Youth; Empowering the Nation