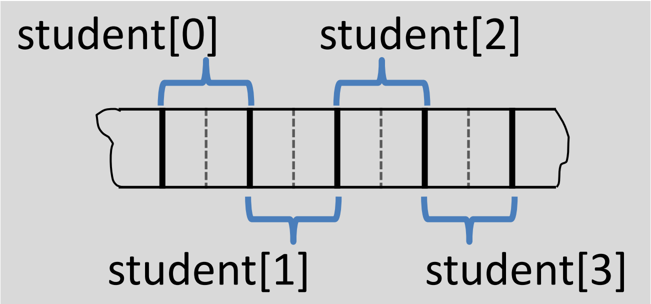
Lecture 5

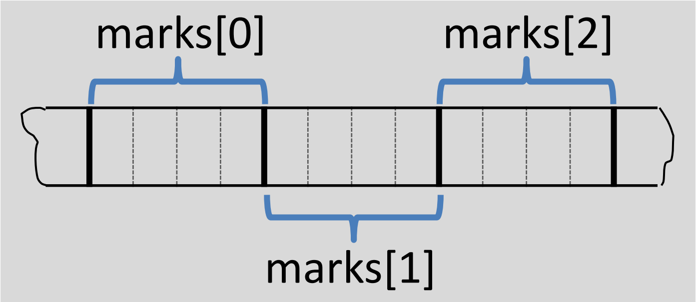
Array Memory Storage

Memory Storage – One Dimensional Array

```
int student[4];
```




```
float marks[3];
```



Memory Storage – Two Dimensional Array

```
int marks[3][5];
```

- Can be visualized in the form of a matrix as

	Col 0	Col 1	Col 2	Col 3	Col 4
Row 0	marks[0][0]	marks[0][1]	marks[0][2]	marks[0][3]	marks[0][4]
Row 1	marks[1][0]	marks[1][1]	marks[1][2]	marks[1][3]	marks[1][4]
Row 2	marks[2][0]	marks[2][1]	marks[2][2]	marks[2][3]	marks[2][4]

Contd…

- Row-major order

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
Row0					Row1					Row2				

- Column-major order

(0,0)	(1,0)	(2,0)	(0,1)	(1,1)	(2,1)	(0,2)	(1,2)	(2,2)	(0,3)	(1,3)	(2,3)	(0,4)	(1,4)	(2,4)
Col0			Col1			Col2			Col3			Col4		

Array Address Computation

1D array – address calculation

- Let A be a one dimensional array.
- Formula to compute the address of the I^{th} element of an array (A[I]) is:

Address of A[I] = B + W * (I – LB)

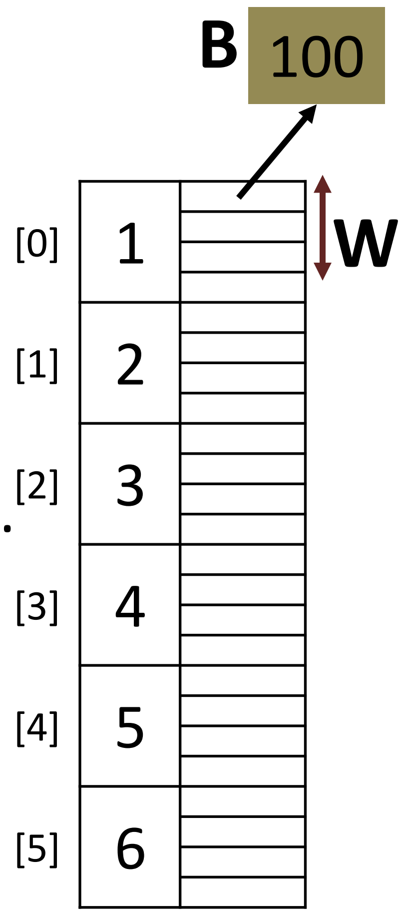
where,

B = Base address/address of first element, i.e. A[LB].

W = Number of bytes used to store a single array element.

I = Subscript of element whose address is to be found.

LB = Lower limit / Lower Bound of subscript, if not specified assume 0 (zero).



1D array – address calculation

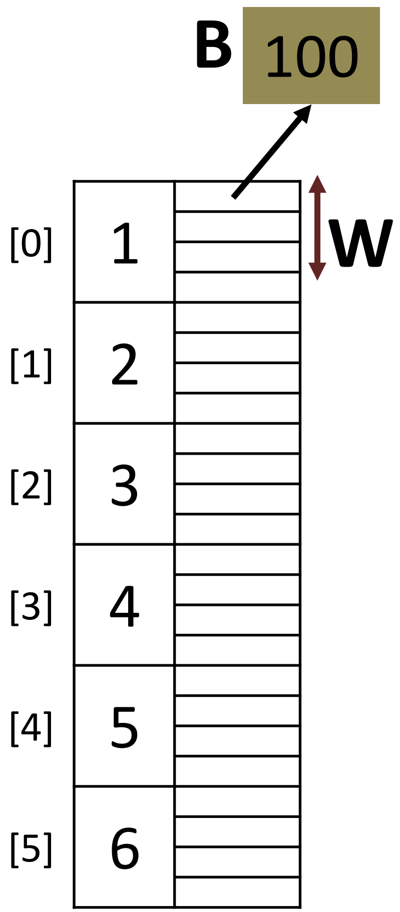
- Let A be a one dimensional array.
- Formula to compute the address of the I^{th} element of an array (A[I]) is:

Address of A[I] = B + W * (I – LB)

Given:

B = 100, W = 4, and LB = 0

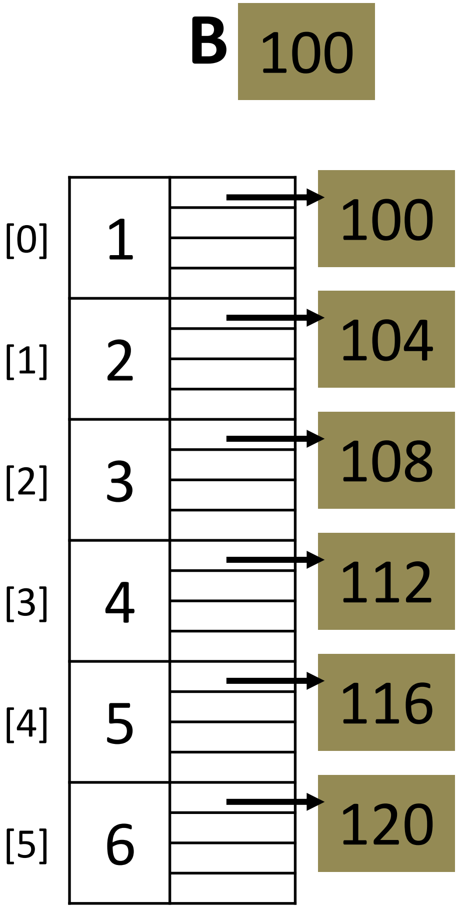
A[0] = 100 + 4 * (0 – 0) = 100



1D array – address calculation

- Let A be a one dimensional array.
- Formula to compute the address of the I^{th} element of an array (A[I]) is:

Address of A[I] = B + W * (I – LB)



Given:

B = 100, W = 4, and LB = 0

A[1] = 100 + 4 * (1 – 0) = 104

A[2] = 100 + 4 * (2 – 0) = 108

A[3] = 100 + 4 * (3 – 0) = 112

A[4] = 100 + 4 * (4 – 0) = 116

A[5] = 100 + 4 * (5 – 0) = 120

Example – 1

- Similarly, for a character array where a single character uses 1 byte of storage.
- If the base address is 1200 then,

Address of A[I] = B + W * (I – LB)

Address of A[0] = 1200 + 1 * (0 – 0) = 1200

Address of A[1] = 1200 + 1 * (1 – 0) = 1201

…

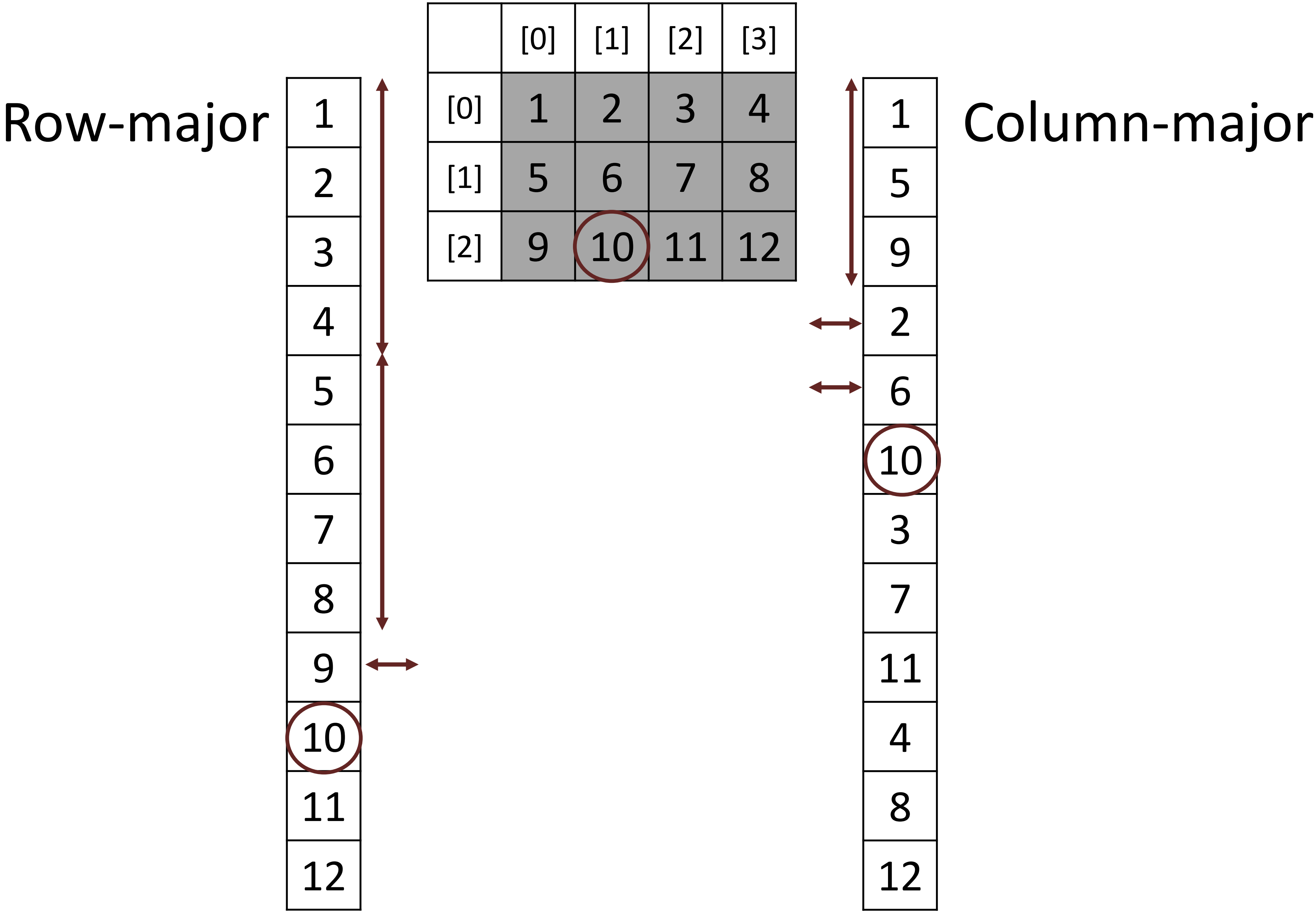
Address of A[10] = 1200 + 1 * (10 – 0) = 1210

Example – 2

- If **LB** = 5, **Loc(A[LB])** = 1200, and **W** = 4.
- Find **Loc(A[8])**.

Address of A[I] = B + W * (I – LB)

$$
\begin{aligned}
Loc(A[8]) &= Loc(A[5]) + 4 * (8 - 5) \\
&= 1200 + 4 * 3 \\
&= 1200 + 12 \\
&= 1212
\end{aligned}
$$


Row-major
Column-major
[0] [1] [2] [3]
[0] 1 2 3 4
[1] 5 6 7 8
[2] 9 10 11 12
1 2 3 4 5 6 7 8 9 10 11 12
1 5 9 2 6 10 3 7 11 4 8 12

2D Array – Address Calculation

- If **A** be a two dimensional array with **M** rows and **N** columns. We can compute the address of an element at **I**th row and **J**th column of an array (**A[I][J]**).

  B = Base address/address of first element, i.e. A[LBR][LBC]
  I = Row subscript of element whose address is to be found
  J = Column subscript of element whose address is to be found
  W = Number of bytes used to store a single array element
  LBR = Lower limit of row/start row index of matrix, if not given assume 0
  LBC = Lower limit of column/start column index of matrix, if not given assume 0
  N = Number of column of the given matrix
  M = Number of row of the given matrix

Row-major

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

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

M = 3

N = 4

Address of A[2][1] =

B + W * (4 * (2 – 0) + (1 – 0))

Address of A[I][J] =

B + W * (N * (I – LBR) + (J – LBC))

M = 3
N = 4

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

Column-major

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

Address of A [2][1] =

B + W * ((2 – 0) + 3 * (1 – 0))

Address of A [I][J] =

B + W * ((I – LBR) + M * (J – LBC))

Contd…

- Row Major

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

- Column Major

Address of A [I][J] = B + W * ((I – LBR) + M * (J – LBC))

- Note: **A[LBR…UBR, LBC…UBC]**

M = (UBR – LBR) + 1

N = (UBC – LBC) + 1

Example – 3

- Suppose elements of array **A[5][5]** occupies **4** bytes, and the address of the first element is **49**. Find the address of the element **A[4][3]** when the storage is row major.

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

- Given: **B** = 49, **W** = 4, **M** = 5, **N** = 5, **I** = 4, **J** = 3, **LBR** = 0, **LBC** = 0.

Address of A[4][3] = 49 + 4 * (5 * (4 – 0) + (3 - 0))

= 49 + 4 * (23)

= 49 + 92

= 141

Thank You