**Evaluation of Prefix an Expression**

**Step:**

1.Right to left.

2.At a time one element.

3.check the element whether operand or operator

i.If operand push into the stack

ii.operator pop top two element.

Top most element=B

2nd Top most element=A

iii.BoA

4.push result back to the stack.

5.cont… until the end of an expression

**-,+,\*,2,3,\*,5,4,9.**

|  |  |  |
| --- | --- | --- |
| **Read Element** | **Stack operation** | **Stack status** |
| 9 | Push(9) | 9 |
| 4 | Push(4) | 9,4 |
| 5 | Push(5) | 9,4,5 |
| \* | Pop (5)\*pop(4)=20 |  |
|  | Push (20) | 9,20 |
| 3 | Push(3) | 9,20,3 |
| 2 | Push(2) | 9,20,3,2 |
| \* | Pop(2)\*pop(3)=6 |  |
|  | Push(6) | 9,20,6 |
| + | Pop(6)+pop(20)=26 |  |
|  | Push(26) | 9,26 |
| - | Pop(26)-pop(9)=17 |  |
|  | Push(17) | 17 |
| \ | Empty | 17(output) |

**Types of an Array**

1.single Dimension Array

2.Two Dimension Array

3.Multi Dimension Array

**Memory Representation of an Array**

**Single Dimension**

|  |  |  |  |
| --- | --- | --- | --- |
| A[0] | A[1] | A[2] | A[3] |

**Two Dimensional Array**

**Int a[3][3]**

**1 2 3**

**4 5 6**

**7 8 9**

**Memory Represenattion of matrix Element**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A[0][0]** | **A[0][1]** | **A[0][2]** | **A[1][0]** | **A[1][1]** | **A[1][2]** | **A[2][0]** | **A[2][1]** | **A[2][2]** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A[0][0]** | **A[0][2]** | **A[1][1]** | **A[1][2]** | **A[2][1]** | **A[1][1]** | **A[2][0]** |  | **A[2][1]** | **A[2][2]** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A[0][0]** | **A[2][1]** | **A[0][1]** | **A[2][2]** | **A[2][0]** | **A[1][0]** | **A[1][1]** | **A[1][2]** |  |

1.RMO(Row Major Ordering)

2.CMO(column Major ordering)

**1.RMO(Row Major Ordering)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A[0][0]** | **A[0][1]** | **A[0][2]** | **A[1][0]** | **A[1][1]** | **A[1][2]** | **A[2][0]** | **A[2][1]** | **A[2][2]** |

**2.CMO(column Major ordering)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A[0][0]** | **A[1][0]** | **A[2][0]** | **A[0][1]** | **A[1][1]** | **A[2][1]** | **A[0][2]** | **A[1][2]** | **A[2][2]** |

**2000 2002 2004 2006 2008 2010 2012 2014 2016 2018**

**Formula For RMO**

**Find the memory address of a[1][2]**

**Loc(a[j][k])=Base(A)+W[N\*(j-RLB)+(k-CLB]**

J&k –the location of elemen jth row kth column.

A-Base Address or starting address memory location

W-width Number of bytes require to store the individual element

N-No.of Column.

RLB-Lower Bound of Row.

CLB-Lower Bound of Column.

**Que-**A[3,2] where base address is 1000 and width is 2 .then calculate the memory address of location L[2,0] using RMO & CMO.

A-

1 2

4 5

6 7

**RMO**

Loc(A[2,0])=1000+2[2\*(2-0)+(0-0)]

=1000+2(4)

=1008

**Assignment**

Loc(A[1,1])

Loc(A[2,0])

**Que2.A=[-2…2,3…7] of elements the starting location is 2000.Each element occupies two memory cell.Calculate the location of A[0,5] using RMO & CMO.**

**Array**

**Row🡺-2 -1 0 1 2**

**Column🡺3 4 5 6 7**

No.rows=5

No.Cols.=5

Loc[0,5]

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 21 23 24 25

Loc(A[0,5])=2000+2[5\*(0-(-2))+(5-3)]

=2000+2[5\*(2)+(2)]

=2000+2(10+2)

=2000+2(12)

=2000+24

=2024(Memory Address)

**Assignment**

**Loc(A[-1,4])**

**Loc(A[2,6])**

**Loc(A[0,6])**