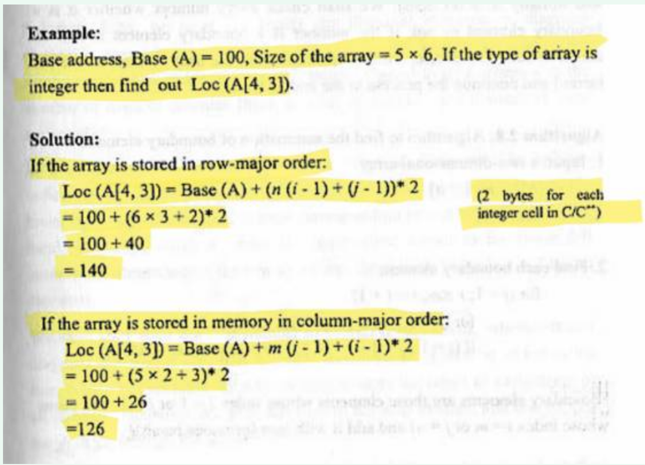
2D Array Element Location



Example:

Base address, Base (A) = 100, Size of the array = 5 × 6. If the type of array is integer then find out Loc (A[4, 3]).

Solution:

If the array is stored in row-major order:

$\text{Loc}(A[4,3]) = \text{Base}(A) + (n(i-1) + (j-1))^{*}\, 2$

$= 100 + (6 \times 3 + 2)^{*}\, 2$

$= 100 + 40$

$= 140$

(2 bytes for each integer cell in C/C++)

If the array is stored in memory in column-major order:

$\text{Loc}(A[4,3]) = \text{Base}(A) + m(j-1) + (i-1)^{*}\, 2$

$= 100 + (5 \times 2 + 3)^{*}\, 2$

$= 100 + 26$

$= 126$

explain why they used 2 byte for integers in row major order and column major order problem of finding element location in a 2D array

In the given problem, **each integer element** in the **2D array** is assumed to take **2 bytes of memory**. This assumption is based on the fact that in some older C/C++ compilers or systems, an **integer is stored in 2 bytes** rather than 4 bytes (which is common in modern systems).

Why Multiply by 2?

- The formula for finding the location of an element in a **2D array** (both row-major and column-major orders) gives the **offset (number of elements) from the base address**.
- However, since each integer element takes **2 bytes**, the offset must be multiplied by **2** to convert it into **memory address units (bytes).**

Row-Major Order Formula:

$$\text{Loc}(A[i,j]) = \text{Base Address} + [(n(i-1)) + (j-1)] \times \text{size of each element}$$

Since each element is **2 bytes**, we multiply by **2**.

Column-Major Order Formula:

$$\text{Loc}(A[i,j]) = \text{Base Address} + [(m(j-1)) + (i-1)] \times \text{size of each element}$$

Again, each element is **2 bytes**, so we multiply by **2**.

If integers were **4 bytes**, we would multiply by **4** instead.

Would you like help solving a similar problem with different array sizes or types?

ChatGPT can make mistakes. Check important info.