Int x[4]

Memory will be allocated for x according to 5 int’s

1 int (as of date) -> 4 bytes

So 4x4 = 16 bytes will be allocated for x

←——————————————— RAM ————————————————————>

←––––x[0] ––––>←––––x[1] ––––>←––––x[2] ––––>←––––x[3] ––––>

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

←—---------- Reserve for x (16 Bytes for x[4] —------------------------------->

700 701 702 703 704 …………………………………………………….715

Address of x[0] - 700

Address of x[1] - 704

Address of x[2] - 708

Address of x[3] - 712

Technically there is no one by the name(s) of x[0]x[1]x[2] or x[3]

When we use these

There isi actually address calculation theory will work here.

**Formula for 1D array = base address + (index x size of one element)**

X[0] = 700 + (0x4) = 700

X[1] = 700 + (1x4) = 704

X[2] = 700 + (2x4) = 708

X[3] = 700 + (3x4) = 712

2D array : int x[3][4] 3 rows, 4 columns

Row major (whatever written in 1st brackets will be major it can be row or column)

X[3][4] it wil not make any shape like matrix

Addressing allocation will be linear

Memory will be allocated foir x

How much? According to (row x column x size of one element)

(3x4) x 4 (int type) = 48 bytes

←———————————— Reserve for x (48 Bytes for x[3][4] ——————————>

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

700 704 708 712 716 720 724 728 732 736 740 744

←––––– 0 index row ––––––>←–––––––– 1st row ––––––>←–––––––– 2nd row ––––––>

**2d Address calculation Formula**

**Base address + (RowIndex x number of columns x size of one element) + columnIndex x size of one element)**

X[0][0] = 700 + (0 x 4 x 4) + (0 x4) = 700

X[0][1] = 700 + (0 x 4 x 4) + (1 x4) = 704

X[0][2] = 700 + (0 x 4 x 4) + (2 x4) = 708

X[0][3] = 700 + (0 x 4 x 4) + (3 x4) = 712

X[1][0] = 700 + (1 x 4 x 4) + (0 x4) = 716

X[1][1] = 700 + (1 x 4 x 4) + (1 x4) = 720

X[1][2] = 700 + (1 x 4 x 4) + (2 x4) = 724

X[1][3] = 700 + (1 x 4 x 4) + (3 x4) = 728

X[2][0] = 700 + (2 x 4 x 4) + (0 x4) = 732

X[2][1] = 700 + (2 x 4 x 4) + (1 x4) = 736

X[2][2] = 700 + (2 x 4 x 4) + (2 x4) = 740

X[2][3] = 700 + (2 x 4 x 4) + (3 x4) = 744