Addressing Array Elements

int A[low..high];
A[i] ++;

```
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
```

- width — width of each element
- base — address of the first
- low/high — lower/upper bound of subscript

Addressing an array element:

```
address(A[i])
= base + (i-low) * width = i * width + (base-low*width)
= i * width + C_1
```
Layout n-dimension items in 1-dimension memory

```c
int A[N1][N2]; /* int A[low1..high1][low2..high2]; */
A[i1][i2] ++;
```

Diagram: Multi-dimensional Arrays

- `A[low1,low2]`
- `A[high1,high2]`
- `i1`
- `i2`
- `N1`
- `N2`
Row major — store row by row

Need to store “blue” items before \( A[i_1,i_2] \)

\[
\text{address}(A[i_1,i_2]) = \text{base} + ((i_1-\text{low}_1) \times N_2 + (i_2-\text{low}_2) \times \text{width}) = (i_1 \times N_2 + i_2) \times \text{width} + C_{2r}
\]
Column major — store column by column

Need to store “blue” items before \( A[i_1, i_2] \)

\[
\text{address}(A[i_1, i_2]) = \text{base} + ((i_2 - \text{low}_2) \times N_1 + (i_1 - \text{low}_1)) \times \text{width}
\]
\[
= (i_2 \times N_1 + i_1) \times \text{width} + C_{2c}
\]