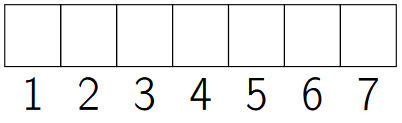
## Arrays

1. https://www.coursera.org/learn/data-structures/lecture/OsBSF/arrays

**Array**: contiguous area of memory consisting of equal-size elements indexed by contiguous integers.

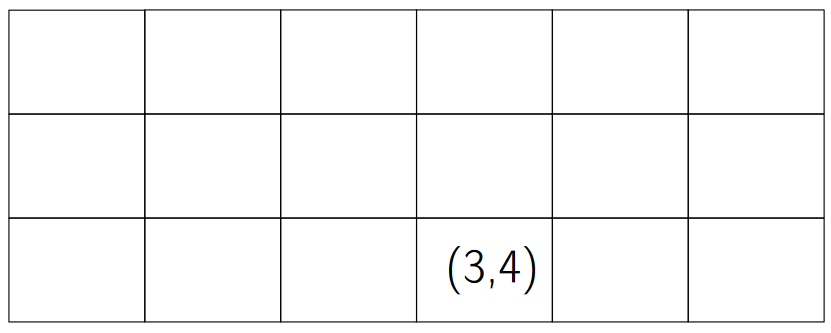


**Major advantage**: constant time access to read/write elements on any index.

*Constant-time access*

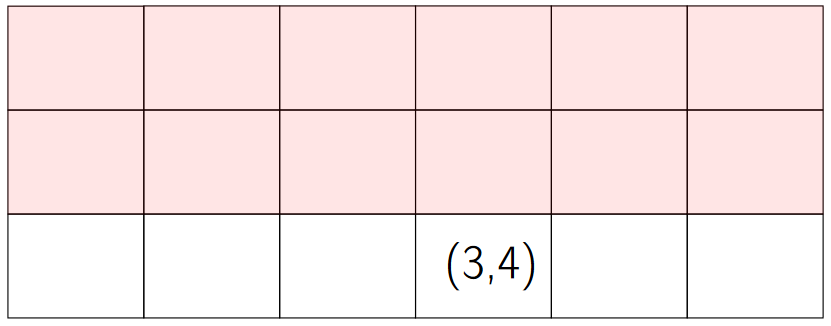
*array\_addr+elem\_size×(i−first\_index)*

### Multi-dimensional Arrays

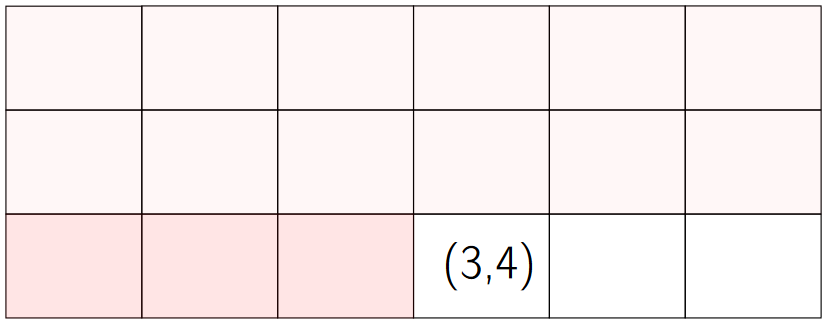


To find the address:

1. Skip the rows that aren’t being used.

(3−1)×6

1. Skip the elements before the index we’re searching for.

(3−1)×6+ (4−1)

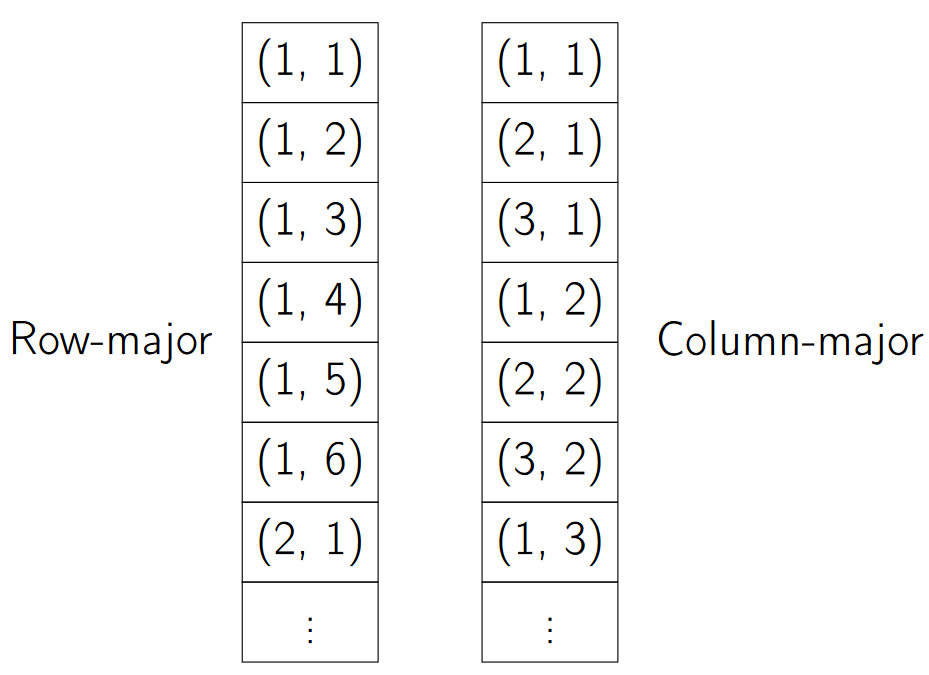
*elem\_size×((3−1)×6+ (4−1))*

1. Final .

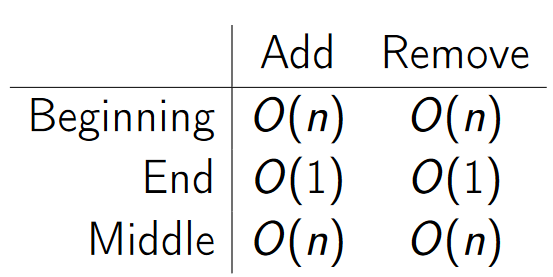
*array\_addr+elem\_size×((3−1)×6+ (4−1))*

### Ordering

1. **Row-major ordering**: the column number changes move rapidly.
2. **Column-major ordering**: the row number changes more rapidly.



### Times for common operations



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Add** | **Remove** | **Note** |
| **Beginning** | O (n) | O (n) | Constant time: to remove the first element, we remove it (the index becomes empty) and then we have to move the rest of the elements one index down. |
| **End** | O (1) | O (1) | Linear time: we only have to add and remove the last index. |
| **Middle** | O (n) | O (n) | Constant time. |

Note: arrays are great if we have to add/remove to the end, but it gets expensive if it’s in the beginning or in the middle.