C++ Programming

Position neighbours

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

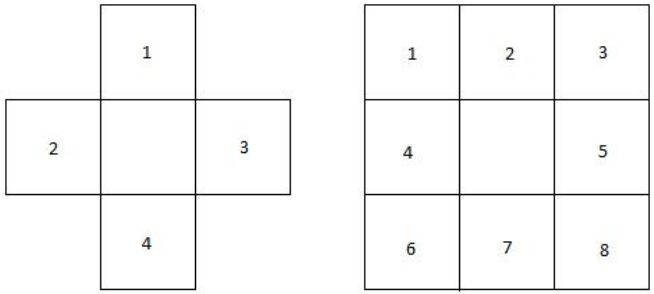
Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Matrix (grid) Structure

- In menu scenarios, we consider data in 2D structure where we have **same number of columns**
- We call it matrix, grid, 2D array
- Let’s see one of the code tricks that simplifies coding when necessary

Position neighbours

- For a position (i, j)
  - Sometimes we use 4 neighbours
    - **up, right, down, left**
  - Sometimes we use 8 neighbours
    - **up, right, down, left**, up right, up left, down right, down left
    - Given (i, j), can u use a loop of 8 steps and print theses 4 or 8 positions, elegantly?



Hint

- Think in position (0, 0)
  - What is its relationships between the 8 neighbours?
  - Create 2 1D lists
  - In each list record the differences such that from any (i, j) we get neighbours?

Let’s find the relation

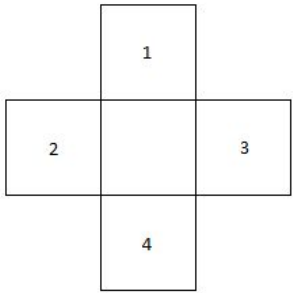
	Up = (r-1, c)	
Left = (r, c-1)	**(r, c)**	Right = (r, c+1)
	Down = (r+1, c)	Diagonal: (r+1, c+1)

- What is change from (r, c) to the down?
  - (r+1, c): row is changed by +1, col is not changed
- What is change from (r, c) to the Left?
  - (r+1, c): row is not changed, col is changed by -1
- We can create 2 arrays to encode these +1/-1/0 changes between locations!
  - Some guys call it **the direction array**

4 Neighbours

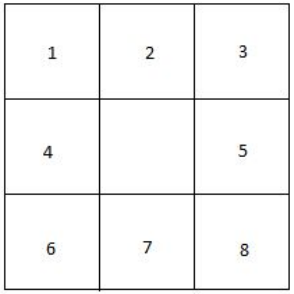
```
//{ d, r,  u,  l};
int di[4] = { 1, 0, -1, 0 };
int dj[4] = { 0, 1, 0, -1 };

int i = 4, j = 7;
for (int d = 0; d < 4; ++d) {
    int ni = i + di[d];
    int nj = i + dj[d];
    cout << ni << " " << nj << "\n";
}
/*
    5 4
    4 5
    3 4
    4 3
 */
```



8 Neighbours

- Your turn: extend the di and ji arrays to be 8 entries for these 8 cells



"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."