fifteen

fifteen.c

- accepts/parses command-line argument
- creates board
- checks if game is won; exits accordingly
- gets input and calls `move`

TODO

- [ ] `init`
- [ ] `draw`
- [ ] `move`
- [ ] `won`

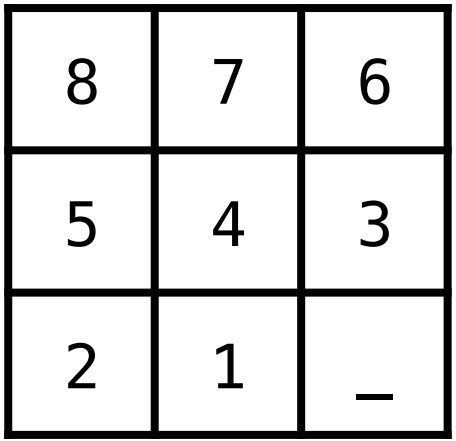
init

`int board[MAX][MAX]`

- board represented by a 2D array

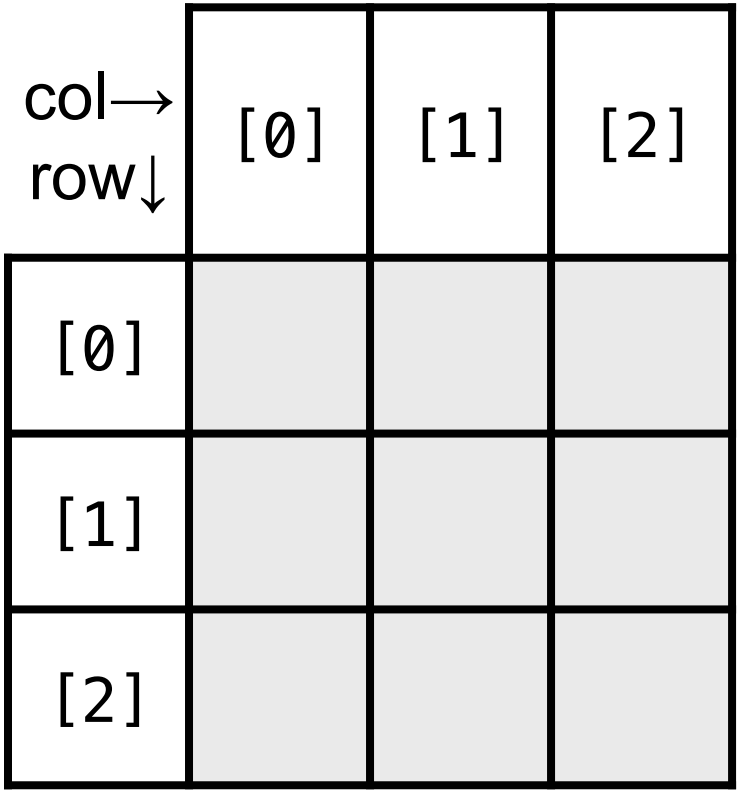
`int d`

- size of board
- `d <= MAX`

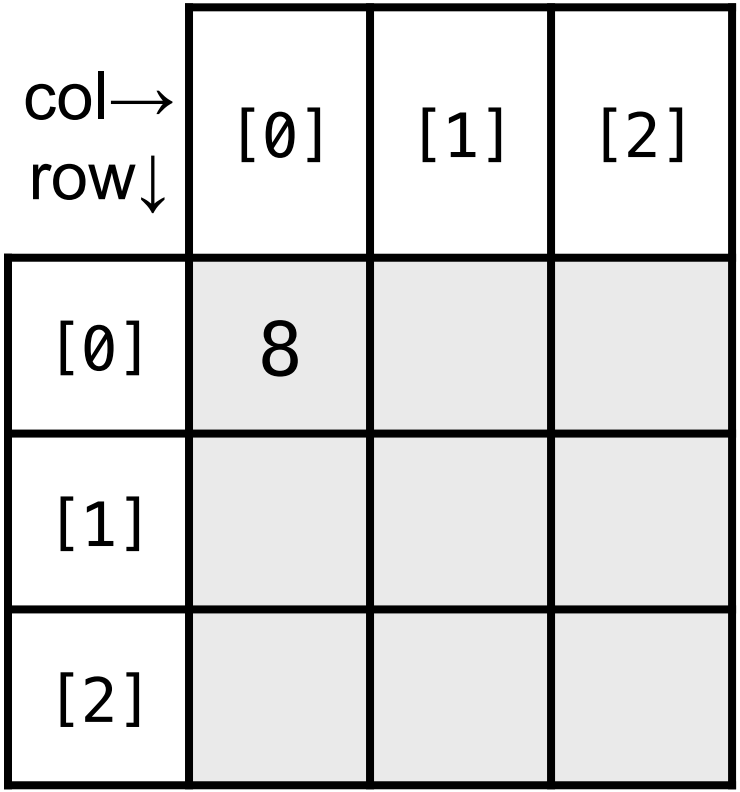


2D Arrays

```
int grid[3][3];
```

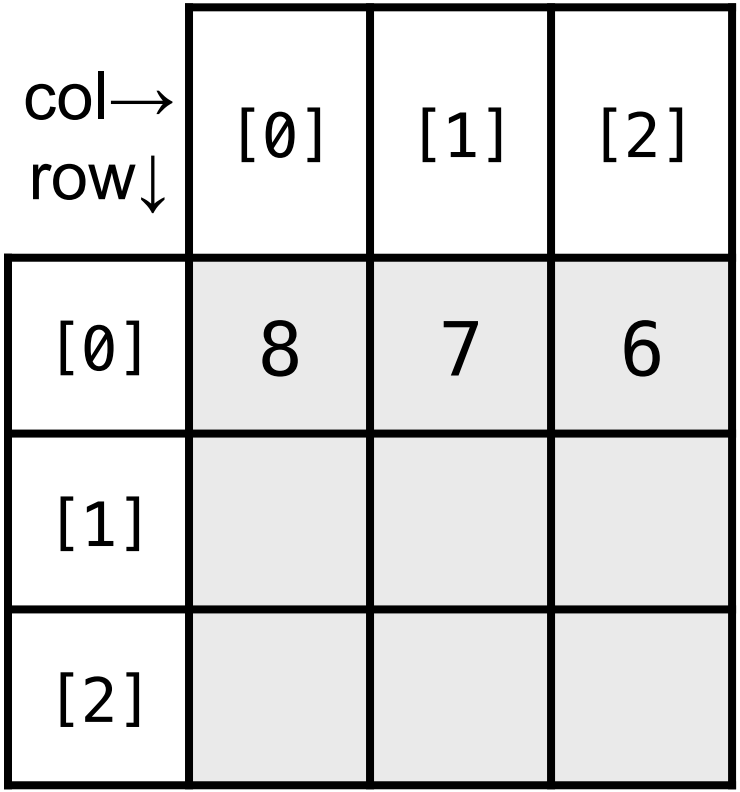


2D Arrays




```
int grid[3][3];
grid[0][0] = 8;
```

2D Arrays




```
int grid[3][3];
grid[0][0] = 8;
grid[0][1] = 7;
grid[0][2] = 6;
```

2D Arrays

col→ row↓	[0]	[1]	[2]
[0]	8	7	6
[1]	5	4	3
[2]			

```
int grid[3][3];
grid[0][0] = 8;
grid[0][1] = 7;
grid[0][2] = 6;
grid[1][0] = 5;
grid[1][1] = 4;
grid[1][2] = 3;
```

2D Arrays

col→ row↓	[0]	[1]	[2]
[0]	8	7	6
[1]	5	4	3
[2]	2	1	

```
int grid[3][3];
grid[0][0] = 8;
grid[0][1] = 7;
grid[0][2] = 6;
grid[1][0] = 5;
grid[1][1] = 4;
grid[1][2] = 3;
grid[2][0] = 2;
grid[2][1] = 1;
```

2D Arrays

col→ row↓	[0]	[1]	[2]
[0]	8	7	6
[1]	5	4	3
[2]	2	1	

```
int grid[3][3];
grid[0][0] = 8;
grid[0][1] = 7;
grid[0][2] = 6;
grid[1][0] = 5;
grid[1][1] = 4;
grid[1][2] = 3;
grid[2][0] = 2;
grid[2][1] = 1;
grid[2][2] = ???
```

`init`

- `board` should contain the starting state
  - `board[i][j]` represents the tile at row `i` and column `j`
- starts in descending order
  - left to right
  - top to bottom

iterate over the grid

```
for each row
    for each column
        set value for tile
```

iterate over the grid

```
for (int i = 0; i < d; i++)
{
    for (int j = 0; j < d; j++)
    {
        // set tile's value
    }
}
```

`init`

- `board` should contain the starting state
  - `board[i][j]` represents the element at row `i` and col `j`
- starts in descending order
  - if `d` is even, swap 2 and 1

odd number of tiles

when **d** is 3

	[0]	[1]	[2]
[0]	8	7	6
[1]	5	4	3
[2]	2	1	

when **d** is 4

	[0]	[1]	[2]	[3]
[0]	15	14	13	12
[1]	11	10	9	8
[2]	7	6	5	4
[3]	3	1	2	

TODO

- ☑ init
- ☐ draw
- ☐ move
- ☐ won

draw

```
for each row
    for each column
        print tile's value
    print new line
```

draw

- print the current state of the board
- print a blank space before single-digit #s
  - `printf("%2i", board[i][j]);`

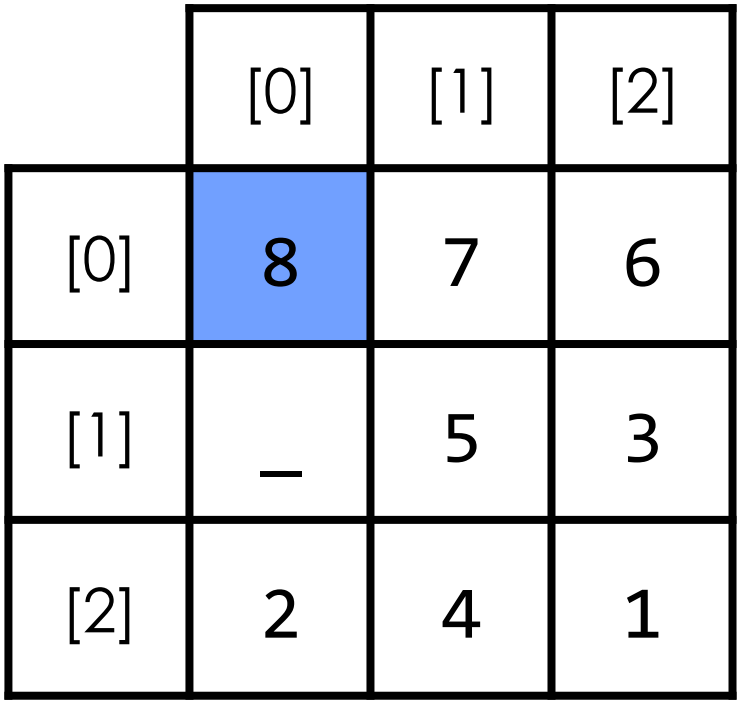
8	7	6
5	4	3
2	1	_

TODO

- ☑ init
- ☑ draw
- ☐ move
- ☐ won

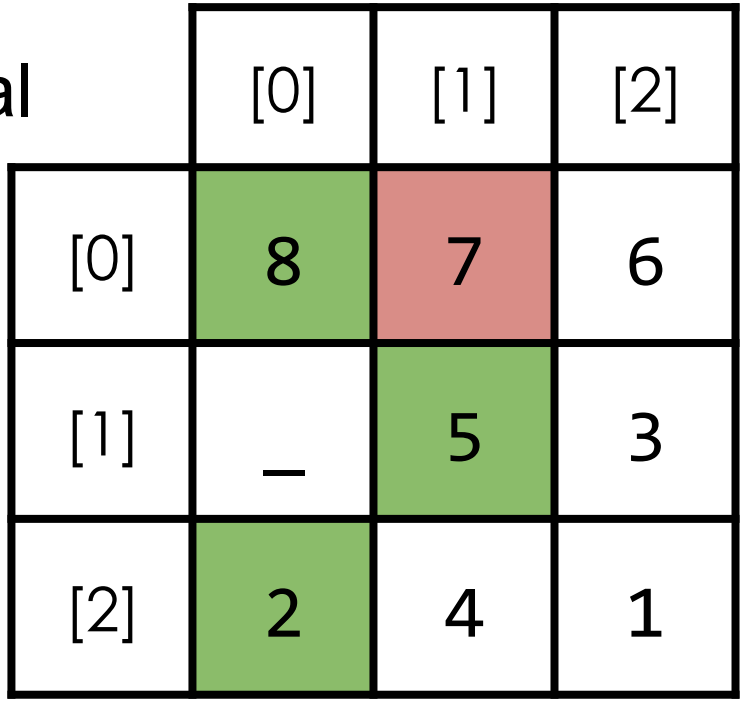
move

- given: tile number
  - not the tile location!
- find the tile location



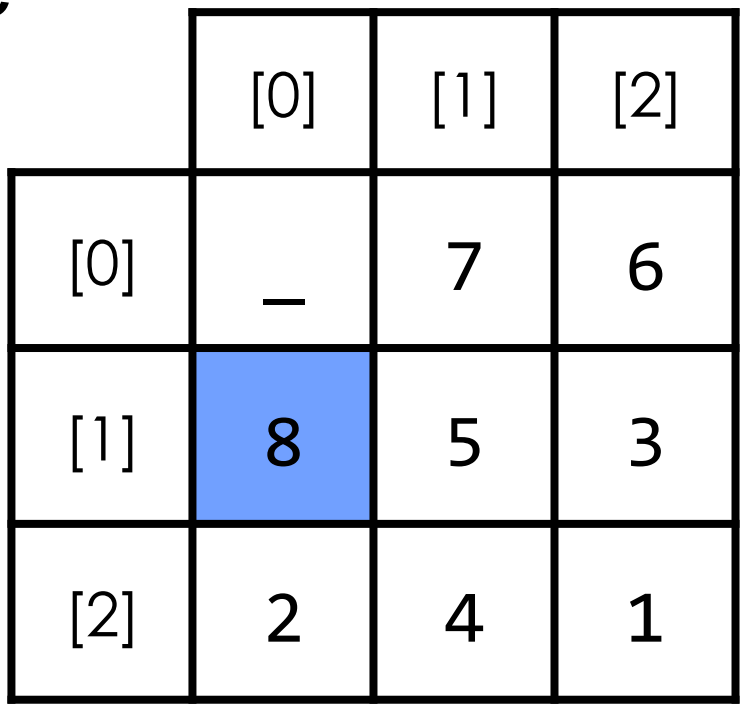
move

- edit the board array
  - … only if the move is legal



move

- swap tile and blank tile
- keep track of blank tile location

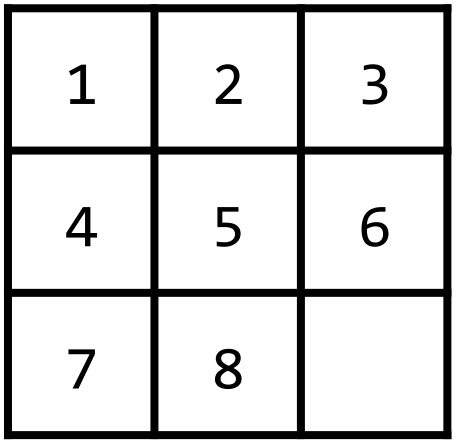


TODO

- ☑ `init`
- ☑ `draw`
- ☑ `move`
- ☐ `won`

won

- returns `true` if the game is won, `false` otherwise
- game is won when tiles are in increasing order
  - left to right
  - top to bottom



won

- iterate over `board` and check the values
- if any value is incorrect, return `false`
- return `true` once all tiles are checked

TODO

- ☑ `init`
- ☑ `draw`
- ☑ `move`
- ☑ `won`

this was fifteen