

# cis1.5-spring2008-azhar, lab V, part 1

## instructions

- This is the first part of the lab/homework assignment for unit V.
- The entire assignment will be worth 9 points.
- The first part is worth 6 points and will be distributed and worked on in class on Monday April 28.
- The second part is worth 3 points and will be distributed and worked on in class on Thursday May 1.
- **Both parts together are due on Monday May 5** and must be submitted by email (as below).
- **Follow these emailing instructions:**
  1. Create a mail message addressed to `mqazhar@sci.brooklyn.cuny.edu` with the subject line **cis1.5 hw5**.
  2. Attach **ONLY** the **.cpp** files for each part, as outlined below.  
DO NOT ATTACH THE **.cbp** (CodeBlocks Project) files!
  3. Failure to follow these instructions will result in points being taken away from your grade. The number of points will be in proportion to the extent to which you did not follow instructions... (which can make it a lot harder for me to grade your work)

## two-dimensional arrays.

We have already talked about *one-dimensional* arrays

- this is a data structure that groups together multiple elements of the same data type
- the grouping is done in “one dimension”, e.g.:  
`mygrades =`

|   |   |   |    |    |
|---|---|---|----|----|
| 1 | 3 | 2 | 99 | 27 |
|---|---|---|----|----|

  
which would be declared as:  
`int mygrades[5];`

But sometimes you want to group elements together using two (or more) dimensions

- *two-dimensional* arrays are when you group elements together in two dimensions, like a matrix or a spreadsheet, e.g.:

`allgrades =`

|   |   |   |   |   |
|---|---|---|---|---|
| 9 | 3 | 7 | 9 | 8 |
| 5 | 6 | 7 | 8 | 3 |
| 6 | 4 | 0 | 1 | 8 |
| 2 | 9 | 1 | 9 | 7 |

which would be declared as:  
`int allgrades[4][5];`

- the entries in the two-dimensional array are referred to as *rows* and *columns*
- the *rows* run horizontally
- the *columns* run vertically

- in the example above, there are 4 rows and 5 columns
- note that in the declaration, the row dimension is declared first, as in `int allgrades[4][5];`
- you could also declare an array with 5 rows and 4 columns, like this: `int allgrades[5][4];`
- you address the elements in a two-dimensional array using two indexes, with the row index coming first, e.g.: `allgrades[0][0]` refers to the entry in the upper left corner of the array; in this case, it has the value 9
- Sample code to display all elements from a two-dimensional array

```
int ROW_SIZE=5;
int COL_SIZE=4;
int allgrades[ROW_SIZE][COL_SIZE];

/*display all elements from a two -dimensional array*/

/*scan row*/
for (int row=0; row<ROW_SIZE; row++ ) {
    /*scan column */
    for ( int col=0; x<COL_SIZE; col++ ) {
        cout << allgrades[row][col];
    }
}
```

For this assignment, you will write a program that generates a two-dimensional array of characters and uses that array to implement a two-player tic-tac-toe game. You can make your own software design decisions about whether you want to write separate functions to do each step of if you want to do all the steps inside the `main()`.

Make sure you refer to the examples we discussed in class for this unit!

Name your program: **ttt.cpp** and submit ONLY this file.

1. Define a two-dimensional  $3 \times 3$  array of characters. (0.5 points)
2. Initialize values in the array to dots (the "." character). (1 point)
3. Display on the screen the values stored in the array, like this: (0.5 points)

```
...
...
...
```

(turn the page over)

4. Display on the screen the values stored in the array, but also include extra characters so that the array looks like a tic-tac-toe board, like this: (1 point)

```
. | . | .  
-+-+--  
. | . | .  
-+-+--  
. | . | .
```

5. Now turn your program into a 2-player tic-tac-toe game. Pretend that each square is numbered 0 through 8, like this:

```
0|1|2  
-+-+--  
3|4|5  
-+-+--  
6|7|8
```

The user who goes first will be "X". The user who goes second will be "O". Create a loop that asks the users, in turn, to enter a number (0 through 8) and then "mark" the array with the user's character. So if you define your array like this:

```
char grid[3][3];
```

and the first user enters 4, then you would set

```
grid[1][1] = 'X';
```

(1 point)

6. When each user enters a number, make sure that the spot in the grid hasn't already been marked. (1 point)  
*Hint:* check if the grid position is set to '.' or not.
7. After each user's mark has been entered in the grid, check to see if the user has won. (1 point)  
*Hint:* check for three in a row across each row, down each column and the two diagonals.

Compile, build and run your program to make sure it works as you expect it to.