# For the Pyramid X\_O Board:

**Class Design:**

* **Properties:**
  + **n\_moves**: Represents the total number of moves made in the game.
  + **n\_cols**: Indicates the number of columns in the pyramid.
  + **n\_rows**: Represents the number of rows in the pyramid.
  + **board**: A dynamically allocated 2D array to store the state of the game board.
* **Constructor:**
  + The constructor initializes **n\_moves**, **n\_cols**, and **n\_rows**.
  + Allocates memory for the **board** using a 2D array of characters.
* **update board Method:**
  + Updates the game board with a player's move.
  + Checks if the game is already over (n\_moves equals 9).
  + Validates the move's coordinates and ensures the selected cell is empty.
  + Updates the cell with the player's mark, increments **n\_moves**, and returns true for a successful move.
* **display\_board Method:**
  + Displays the current state of the game board in a pyramid shape.
  + Utilizes nested loops to traverse the 2D array and print the board's elements.
* **is\_winner Method:**
  + Checks for winning conditions in the game board.
  + Examines the diagonal from bottom-left to top-right and horizontal lines in the bottom row.
  + Returns true if any winning condition is met; otherwise, returns false.
* **is\_draw Method:**
  + Returns true if the number of moves is equal to 9, indicating a draw; otherwise, returns false.
* **game\_is\_over Method:**
  + Returns true if the number of moves is equal to 9, indicating the end of the game; otherwise, returns false.
* **Destructor:**
  + Deallocates memory for the dynamically allocated game board.

# For the connect4 game:

**constructor:**

Setting the board with rows equal to 6 and columns equal to 7 and fill every coordinate with space.

**update board Function:**

It checks whether the user enters the right number to play or not then it changes the board with the play and here we use a vector to calculate the previous x coordinate for each column.

**Display board Function:**

It displays the overall board when we call it.

**Is winner Function:**

Briefly there’s four ways to win in connect four game there’s row win or column win or right to left diagonal or vice versa, each one we will check it alone in a separate for loop we check the possible coordinates to win.

For the row win: we compare each coordinate with the coordinates beside it and check whether the coordinate is free to play or not.

For the column win: we compare each coordinate with the coordinates under it and check whether the coordinate is free to play or not.

For the left to right diagonal win: there’s only 12 possibilities we loop over their coordinates compare each one with the right down coordinate to get the diagonal shape.

For the right to left diagonal win: there’s only 12 possibilities we loop over their coordinates compare each one with the left down coordinate to get the diagonal shape.

**Is draw Function:** checks whether the game ended with draw or not by checking its maximum moves (42) and no one won.

**Is over Function:** checks only the maximum moves (42).

# For the 5\*5 X\_O Board:

**Class Design:**

* **Properties:**
  + **n\_moves**: Represents the total number of moves made in the game.
  + **n\_cols**: Indicates the number of columns in board.
  + **n\_rows**: Represents the number of rows in the board.
  + **board**: A dynamically allocated 2D array to store the state of the game board.
* **Constructor:**
  + The constructor initializes **n\_moves**, **n\_cols**, and **n\_rows**.
  + Allocates memory for the **board** using a 2D array of characters.
* **update board Method:**
  + check if the player’s move is in the coordinates of the board.
  + Updates the game board with a player's move.
  + Updates the cell with the player's mark, increments **n\_moves**, and returns true for a successful move.
* **display\_board Method:**
  + Displays the current state of the game board in a 5\*5 board.
  + Utilizes nested loops to traverse the 2D array and print the board's elements.
* **is\_winner Method:**
  + Checks for winning conditions in the game board.
  + Examines the rows and columns and diagonals for 3 consecutive X’s or O’s.
  + Returns true if any winning condition is met; otherwise, returns false.
* **is\_draw Method:**
  + Returns true if the number of moves is equal to 25, indicating a draw; otherwise, returns false.
* **game\_is\_over Method:**
  + Returns true if the number of moves is equal to 25, indicating the end of the game; otherwise, returns false.
* **Destructor:**
  + Deallocates memory for the dynamically allocated game board.

# Quality code:

1. the connect4 game

A screenshot of a computer program

Description automatically generated

1. pyramid X\_O game // NB: no commenting in the code

A screenshot of a computer

Description automatically generated

1. 5\*5 tic tac toe game // NB: no commenting in the code

A screenshot of a computer

Description automatically generated