Dots and Boxes Top Down Design

user description: make a program to play a dots and boxes game, one human player against a computer player

++

STEP :0

pseudo code:

```
in main()
        play game
                // call to stages of the game
                        pre game stage
                        game
                        post game
```

++

STEP :1

dotsAndBoxes.cpp (Main)

Call to game.cpp

*game.cpp // runs the game
- **pregame
- **game in progress
- **post game

*grid.cpp// displays the game board
- ** display grid
- * *players moves

*box.cpp// adds the C or H
- **display claimed box
- **determine degrees of freedom
- ** determine score

*human
- **assign player

*computer// moves the computer player
- ** four Rules check

++

STEP :2

dotsAndBoxes.cpp (Main)

Call to game.cpp

*game.cpp // runs the game
- **pregame
  - ***call setup game
- **game in progress
  - *** assign turns
  - *** check for status of game
- **post game
  - ***display final scores

*grid.cpp

** display grid

*** constructors

*** initialize edges

* *players moves

***check if a legal move

*box.cpp

**display claimed box

***initialize box values to " "

**determine degrees of freedom

*** check if dot 1 and dot are legal

*** check north, south, east, west positions form dot1 and dot2

** determine score

***if player symbol (C or H) is assigned, update score

*human

**assign player

***check whose turn

*computer

** four Rules check

***rule one function

***rule two function

***rule three function

***rule four function

++

STEP :3

dotsAndBoxes.cpp (Main)

Call to game.cpp

***game.cpp // runs the game**

pregame member function **(fx- abbreviated)

***call setup game

****call display grid fx

**game in progress member fx

*** assign turns

**** set turn function

*** check for status of game

****Boolean board game not full fx

**** Boolean human player quit game fx

**post game member fx

***display final scores

****display final grid

*****for(inti=0; i< gridsizeRows; i++) //outerloop

//alphabet label prints here

******for(int j=0; gridsixeColumns; j++)// innerloop

array Horizontal(get value)

for(int j=0; gridsixeColumns; j++)// innerloop vertical

array vertical(get value) + Box check(get value)

// END LOOP

number label goes here

****display final scores
*****cout<<"Final Score: "<<endl;
cout<<"You :" << humanScore<<endl;
cout<<"Me :" << computerScore<<endl;

****display final message
if conditions You == me ; then tie msg rematch
else if you < me; then congats msg
else ; better luck next time msg

***grid.cpp// displays the game board**

** display grid
*** constructors
****initialize edges
*** initialize edges
****forloop horizontal
****forloop vertical
****print row letters
****print column numbers
* *players moves
***check if a legal move
****check if human player quits
****check is data in bounds
****check is data correct data type
****check is board space is available

*box.cpp// adds the C or H

**display claimed box
***initialize box values to " "
****array or class obj ??

PROBLEM> complex, trying to decide between a struct of top, bottom, left and right or to handle with an array that will cover the gird blank spaces only

**determine degrees of freedom
*** check if dot 1 and dot are legal
****check horizontal edges (is taken?)
****check vertical edge (is taken?)
*** check north, south, east, west positions form dot1 and dot2
****assign coordinates to be checked check

PROBLEM> complex, trying to decide how to convert Dot1 and Dot2 input from user to matching horizontal_edge [][] and vertical_edge[][] from grid.cpp. Using if conditions I could check if the letters from Dot 1 and Dot2 are equal; then I can assume a horizontal edge and check that the Dot1 num <the Dot2 number. Then consider if this is good input, check if the array element has a blank space(not taken) and update the array with the correct "-" symbol. Such that horizontal_edge[0][0] = '-', for the first box of the grid (top left, assigned array value 0,0). Then

send this back out to the grid.cpp to display the current grid. I would repeat for the vertical edge as well.

** determine score

***if player symbol (C or H) is assigned, update score

**** display letter and display grid

***take another turn or end turn?

PROBLEM> complex trying to use a boolean value [true for player1=human and false for computer] not sure how to handle not sure to use human.cpp or box.cpp for this problem

***human.cpp**

**assign player

***Boolean check whose turn

***computer.cpp // moves the computer player**

** four Rules check

PROBLEM> I noted the conditions for each rule would need to be checked first; then fill the edge and assign the turn. those two functions were changed form level 4 to level 5 here. Level 4 was re vamped here. **fx = function**

***rule one function

****if boxes w/3 edges exist, do filledge and exit loop, exit 4 rule fx

****else if boxes w? !3 edges exist, choose an edge to fill, exit 4 rule fx

****else select and empty edge

*****return a Boolean value to assign turn to player

*****fill the edge fx

***** assign turn to move

***rule two function

****list degrees of freedom fx

*****calculate the degrees of freedom

****sort degrees of freedom and ranks

*****use temp_array for sorted items

******if array[0] = array[1] exit loop to rule3

******else fill the edge

***** and assign the turn to move

PROBLEM> Do I use 2 arrays 1 for horizontal & one for vertical? (to able to pass to the next 2 if conditions as x & y arrays to sort also?) Or treat as 3 separate searches?

***rule three function

****if(x-cordinateSort())

// takes and sorts from highest to lowest edges in boxes

*****array, sort, return choice

*****filledge

*****assign the turn to move (Boolean player assign)

****else go to rule 4

***rule four function

****if (y_cordinateSort ()

// takes and sorts from highest to lowest edges in boxes

*****array, sort, return choice

*****filledge

*****assign the turn to move (Boolean player assign)

*****exit loop and function

NOTE TO GRADER: I was not sure how to handle the situation to show all my knowledge. I also did not want the basically streamline code above to have too many low level code ideas.

When making a class, the variables used by the class are often private, therefore to access these the .h file must declare the gets and sets function prototypes and the .cpp file will then define the functions. Unless you use the inline reserved word, then both can be in the .h file and allows the developer to use these functions with the class identifier. So, above in each class you will use a constructor to initialize these values, each time an object (instance) is called. For example Grid grid; will invoke the constructor of the class Grid, and make an object (like a variable of the class datatype). The sets, gets, and constructor functions are needed but I did not list each one individually. For example the user will input a size for the game grid, then the size variable goes into the constructor(size), the value gets initialized; then is used in the member functions as n_rows and n_cols. However,the public functions and variables are able to be accessed by the main.cpp, the class, and other classes.

When I re-read the directions for top-down design documentation, it was open to interpretation, using step-wise, pseudo-code, real code, and asked for paragraphs explaining problems and to note unsolved areas. I included my initial step-wise as a way to see the levels progress and the changes I made as I was re-thinking my approach.

In my program code, I used Professor Kennedy's trick of testing code in main.cpp then changing that code to be used in classes (move out of main.cpp and into class.h and class.cpp files). Though I have not placed my game function call in dotsAndBoxes.cpp (main.cpp); I was working towards that goal. I wanted few lines of code indotsAndBoxes.cpp file (less than 10), then the game stages would be managed in game.cpp. Each stage would call other classes, like pre-game would use the grid.cpp class functions. I realize, that is the better approach for solving this problem.

Overall, what I found to be most challenging is the idea to move from a one file approach to a multi-file approach. The other challenge I faced was logically I am used to stepping through a program as it is being executed, such as from line 0 to line 300 and solving line by line. I have used few nested functions and classes, and found I was not using all my tools in my toolbox effectively. The step-wise exercise really forces you to think in a different way and I can see the advantages of using this approach.