Intro to Computing—CSCI-1310

Mon, Apr 6, 2015 · Lab

Lab 12—References and streams

Objectives

Use `references` as parameters to functions.
Use `output stream(ostream)` as parameter instead of `cout`

Use the checker board program of `Lab 11` and build on to it.

Part 1:

Modify the `displayBoard()` so that it takes in cout as a reference. Then use the reference variable instead of cout statements.

Syntax:

```
void displayBoard(ostream &,char board[8][8]);
```

Part 2:

Write a function named `move` which allows the checker pieces to move around on the board. It takes `six` parameters:

rowSource- The row number from which the checker piece has to be moved
columnSource- The column number from which the checker piece has to be moved
rowDestination- The row number to which the checker piece has to be moved
columnDestination- The column number to which the checker piece has to be moved
success- An `int` reference variable which indicates whether there was a successful move or not
board- The `char` array that contains the checker board

Syntax:

```
void move(int rowSource ,int columnSource, int rowDestination, int columnDestination, int& success, c
                                                                {
                                                    //Check for the conditions to move the pieces.
                                                                }
```

move.cc hosted with ❤ by **GitHub** **view raw**

`move()` should check for the following conditions:

Check if the location the piece is moving to is already taken.
Check if location entered by the user contains a piece to be moved.
Check if the piece is not moving diagonally.

If there is an unsuccessful move that is if any of the above conditions are not satisfied then set value of the reference variable, success to 0.

Only if there is a successful move that is value of success is 1, then call the `displayBoard()` else print out `"Illegal move!!"`.

`main()`:

`main()` does the following:

Accepts rowSource, columnSource, rowDestination and columnDestination as input from user.
Initialize the value of success to 1 and then pass a reference of it to `move()`.
Call `move()` and pass the parameters to it.

Output1:

```
  0 |1 |2 |3 |4 |5 |6 |7 |
0 *| b| *| b| *| b| *| b|
1 b| *| b| *| b| *| b| *|
2 *| b| *| b| *| b| *| b|
3 *| *| *| *| *| *| *| *|
4 *| *| *| *| *| *| *| *|
5 r| *| r| *| r| *| r| *|
6 *| r| *| r| *| r| *| r|
7 r| *| r| *| r| *| r| *|
Select the piece to be moved:
       Move piece from
            Row: 5
         Column: 1
           To box
            Row: 3
         Column: 1
       Illegal move!!
```

Output1 hosted with ❤ by **GitHub** **view raw**

Output2:

```
  0 |1 |2 |3 |4 |5 |6 |7 |
0 *| b| *| b| *| b| *| b|
1 b| *| b| *| b| *| b| *|
2 *| b| *| b| *| b| *| b|
3 *| *| *| *| *| *| *| *|
4 *| *| *| *| *| *| *| *|
5 r| *| r| *| r| *| r| *|
6 *| r| *| r| *| r| *| r|
7 r| *| r| *| r| *| r| *|
Select the piece to be moved:
      Move piece from
          Row: 5
         Column: 0
          To box
          Row: 4
         Column: 1
  0 |1 |2 |3 |4 |5 |6 |7 |
0 *| b| *| b| *| b| *| b|
1 b| *| b| *| b| *| b| *|
2 *| b| *| b| *| b| *| b|
3 *| *| *| *| *| *| *| *|
4 *| r| *| *| *| *| *| *|
5 *| *| r| *| r| *| r| *|
6 *| r| *| r| *| r| *| r|
7 r| *| r| *| r| *| r| *|
```

Output2 hosted with ❤ by **GitHub** **view raw**

Zip and submit

To get credit for this lab exercise:

Submit the file to Moodle as a zip file named Firstname_Lastname_Lab12.zip
Show the TA your code and run your program.

Structs & classes Strings & vectors