Project 2

For Projects 2 and 3 you will be implementing a grid based “move and match” game. For Project 2, you will be implementing the game logic with an ASCII game interface. For Project 3 you will be creating GUI interface.

Preliminary assignment: visit the website http://www.cs.sonoma.edu/~cs215/cs215f17/Project3/ and play the prototype games posted there. (You will need to do this from a Windows platform.) Also, after you copy the starter files from the course pickup folder, play the demo games included in the folder.

Specifications

1. Copy the folder called Project2 from the course pickup directory. The folder will contain (at least) the following files:
   ascii.cpp  CGame.h    colorloop.cpp demo2  Proj2app.cpp  SetLimits.h
   CGame.cpp  color.cpp  dem01 input  makefile  SetLimits.cpp
2. Carefully review each of these files. ascii.cpp, color.cpp, and colorloop.cpp are all standalone programs that you can use for experimentation. In addition to looking at the code in these programs, you should also compile and run each of these programs.
3. Modify the makefile to make it yours. To do this, you should replace all instances of “yln” with your last name.
4. Compile and run the starter program by entering:
made run
5. Complete the file documentation block at the beginning of each of the files CGame.h, CGame.cpp, and Proj2app.cpp.
6. Implement the functions in CGame.cpp. The suggested order for implementing the functions is:
   a. void CGame::Instructions (ostream & outs, istream & ins);
      This function should display the instructions for playing your game. The title of your game should be displayed as part of the instructions.
   b. void CGame::Display (ostream & outs);
      This function should display the current state of the game using ASCII characters. A simple version of this function has been provided. You should modify it to enhance the appearance of the game interface.
   c. void CGame::CGameSquare::Display (ostream & outs);
      This function should display the symbol associated with a single game square. You may use any set of symbols you would like. A simple version of this function has
been provided. You should modify it to enhance the appearance of the game interface.

d. **void CGame::CGameSquare::Fill ();**
   This function should fill in the individual game square. The whatSymbol variable should be initialized with a random integer value representing which symbol will be displayed for the square. You should use the rand() function for selecting a “pseudo random” value; information about this function can be found at: http://www.cplusplus.com/reference/cstdlib/rand/

e. **void CGame::Move (char direction);**
   This function will be called when ‘a’, ‘w’, ‘s’, or ‘d’ is entered. It should swap the “ME” squared with one of the squares adjacent to it as long as the “ME” square remains in the grid. This function should decrement the movesLeft counter if the move is valid.

f. **void CGame::Match ();**
   This function will be called when an ‘m’ is entered. It should look for and mark “matches” in the grid. A match is defined as 3 or more squares with the same symbol adjacent to each other in a row or column. This function should also appropriately update the player’s score.

g. **void CGame::Update ();**
   This function will be called when a ‘u’ is entered. It should update the grid by removing the matched squares and replacing them with new squares.

h. **CGame::CGame (const CGame & other);**
   This function should make a copy of an existing game object called “other”.

i. **CGame & CGame::operator = (const CGame & other);**
   This function should copy an existing game object called “other” into ”this” game object.

j. **void CGame::Restart ();**
   This function will be called when an ‘r’ is entered. It should replace the current game with a copy of the game in its initial state. This function will use the CGame class assignment operator.

k. **bool CGame::Done ();**
   This function should return true if the number of moves remaining has hit zero.

l. **void CGame::Message (bool done, ostream & outs);**
   This function should display a message when the prayer completes the game by using all of the allocated moves or elects to quit the game by entering ‘q’.

m. **void CGame::Init (int R, int C, int M);**
   This function should initialize all of the values in the existing game object based on the input parameters, R (number of rows), C (number of columns), and M (number of permitted moves). Most of this function has already been implemented.

n. **void CGame::FillIn ();**
   This function should be called to fill in the squares of a new game. Most of this function has already been implemented.*
o. CGame::CGame ();
   This function should initialize a new game object. Most of this function has already been implemented.*

p. CGame::CGameSquare::CGameSquare ();
   This function should initialize a new game square object. Most of this function has already been implemented.*

q. CGame::~CGame ();
   This function should release the space allocated for a game object. Most of this function has already been implemented.*

* You should only need to modify these functions if you modify the contents of the class.

7. Verify that your program is running correctly before you submit it. You may incorporate color in your program if you have correctly completed all of the program functions.

Date Due: Sunday, 22 October 2017, 11:59 pm
To Turn In: YourLastNameP2.tgz containing a tarred and zipped version of a directory called YourLastNameP2. This should only contain the well documented files required to reconstruct your executable (P2.out). You must also include a file called README.txt in which you describe what works and what does not work in your submission. You should consider using the submit target in the makefile provided with the other starter files for this project. You will need to change all of the

/******************************************************************************
* Assignment: Project 2 - ASCII Move and Match Game
* Author: 
* Date: Fall 2017
* File: CGame.h
* 
* Description: This file contains the
* ***************************************************************
*/

#ifndef CGAME_H
#define CGAME_H

#include <iostream>
using namespace std;

/******************************************************************************
* Type: sType
* 
* Description: The sType enumerated type is used to identify the type of the
* information stored in a CGameSquare object.
* ***************************************************************
*/

enum sType {BORDER, ME, SYMBOL, EMPTY};
class CGame
{
   public:
      CGame ();
      CGame (const CGame & other);
      ~CGame ();
      CGame & operator = (const CGame & other);
      void Init (int R, int C, int M);
      void Instructions (ostream & outs, istream & ins);
      void Display (ostream & outs);
      void Move (char direction);
      void Match ();
      void Update ();
      void Restart ();
      bool Done ();
      void Message (bool done, ostream & outs);

   private:
      struct CGameSquare
      {
         CGameSquare ();
         void Display (ostream & outs);
         void Fill ();
         sType what;
         int whichSymbol;
         bool matched;
         int value;
      };
      void FillIn ();
      CGameSquare ** grid;
      int numRows, numCols;
      int myRow, myCol;
      int movesLeft;
      int score;
};

#if 0
#endif