#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <string>

#include <ctime>

#include <Windows.h> // Libraries of code because we are lazy.

using namespace std;

void OneDimtoTwoDim(int size, string objects[], string TwoDim[][8]);

int ChooseRow(int size);

int ChooseCol(int size);

void printBoard(string board[][8], int size);

int main()

{

string sGridTheme[8][8];

string objects[50] = { "chair", "table", "hat", "ball", "candle", "yo-yo", "pencil", "yarn", "cup", "pants",

"glasses", "fruit", "stick", "glove", "paddle", "spoon", "plate", "guitar", "sponge",

"hair", "pin", "dice", "thimble", "needle", "shoe", "card", "bench", "foot", "grass",

"court", "hammer", "shirt", "pants", "kilt", "bone", "tea", "ice", "water", "boat",

"bridge", "quilt", "brain", "vase", "rope", "garbage", "silk", "watch", "floor", "plant",

"computer" };

int iGame = 0;

int iSize = 0;

int iBoard = 0;

int iLength = 0;

int idifficulty = 0;

int ireveal = 0;

cout << endl;

cout << "~~~~~~~~~~~ Memory Match Mayhem ~~~~~~~~~~~" << endl; // Title of game.

cout << endl;

cout << " New Game = 1" << endl;

cout << " Quit Game = 0" << endl;

cin >> iGame;

system("CLS");

if (iGame == 1) // Start a new game.

{

cout << endl;

}

else

{

return 0; // Exit application.

}

// Input validation.

while (!(iSize >= 1 && iSize <= 3))

{

cout << " Ok! Now please select a difficulty:" << endl; // ask for board size.

cout << " Rookie (4x4) = 1" << endl;

cout << " Veteran (6x6) = 2" << endl;

cout << " God (8x8) = 3" << endl;

cin >> iSize;

}

system("CLS"); // clear the screen.

if (iSize == 1) {

iSize = 4;

}

if (iSize == 2) {

iSize = 6;

}

if (iSize == 3) {

iSize = 8;

}

cout << endl;

iLength = iSize\*iSize;

// Input validation.

while (!(ireveal >= 1 && ireveal <= 3))

{

cout << " And lastly, please select the amount" << endl; // ask for the time interval.

cout << " of time the cards will be visible:" << endl;

cout << " Rookie 5 seconds = 1" << endl;

cout << " Veteran 3 seconds = 2" << endl;

cout << " God 1 second = 3" << endl;

cin >> ireveal;

}

system("CLS");

cout << endl;

if (ireveal == 1) {

ireveal = 5000;

}

if (ireveal == 2) {

ireveal = 3000;

}

if (ireveal == 3) {

ireveal = 1000;

}

cout << " Let's begin!" << endl;

cout << endl;

Sleep(1500); // To pause the program for 1.5 sec.

system("CLS");

cout << endl;

for (int column = 0; column < iSize; column++) {

cout << setw(10) << column;

}

cout << endl;

for (int row = 0; row < iSize; row++) { // initialize and print out the question mark board.

cout << " " << row << "|";

for (int col = 0; col < iSize; col++) {

sGridTheme[row][col] = "????????";

cout << setw(10) << sGridTheme[row][col];

}

cout << endl;

}

srand(time(NULL));

string temp;

int iTemp;

for (int x = 0; x < 50; x++) { // randomize the theme array (in this case objects).

iTemp = rand() % 49;

temp = objects[x];

objects[x] = objects[iTemp];

objects[iTemp] = temp;

}

string sPlayList[64];

int pairs = iLength / 2;

for (int y = 0, x = 0 ; y < iLength/2; y++, x = x+2) { // populate the sPlayList array.

sPlayList[x] = objects[y];

sPlayList[x+1] = objects[y];

}

// create randomizer for sPlayList look at temp.

string sPlayBoard[8][8];

string temp1;

int iTemp1;

for (int x = 0; x < iLength; x++) {

iTemp1 = rand() % iLength;

temp1 = sPlayList[x];

sPlayList[x] = sPlayList[iTemp1];

sPlayList[iTemp1] = temp1;

}

OneDimtoTwoDim(iSize, sPlayList, sPlayBoard); // change a one dim array to a two dim array.

//printBoard(sPlayBoard, iSize); // Board to verify answers.

int row1, row2, col1, col2;

int foundPairs = 0; // declare variables to be used in the following.

while (true) {

while (true) { // choose the first box on the board already printed.

row1 = ChooseRow(iSize);

col1 = ChooseCol(iSize);

if (sGridTheme[row1][col1] == "????????") // condition statements.

break;

}

system("CLS");

cout << endl;

sGridTheme[row1][col1] = sPlayBoard[row1][col1]; // print the new board and clear screen.

printBoard(sGridTheme, iSize);

while (true) {

row2 = ChooseRow(iSize); // choose the net box.

col2 = ChooseCol(iSize);

if (sGridTheme[row2][col2] == "????????" && (row1 != row2 || col1 != col2)) // validate the boxes.

break;

}

system("CLS");

cout << endl;

sGridTheme[row2][col2] = sPlayBoard[row2][col2]; // Display the new board with the two selections

printBoard(sGridTheme, iSize); // on it.

if (sGridTheme[row1][col1] == sGridTheme[row2][col2]) { // If they match, remove them and count variable.

Sleep(1000);

foundPairs++;

sGridTheme[row1][col1] = " ";

sGridTheme[row2][col2] = " ";

}

else {

Sleep(ireveal);

sGridTheme[row1][col1] = "????????"; // if not a match, replace.

sGridTheme[row2][col2] = "????????";

}

system("CLS");

cout << endl;

printBoard(sGridTheme, iSize); // reprint board to get a new selection.

if (foundPairs == pairs) { // game ends.

break;

}

}

system("CLS");

cout << endl;

cout << "Congratulations! You cleared the board!" << endl;

cout << endl;

cout << "Thanks for playing!" << endl;

cout << endl;

system("pause");

return 0;

}

void OneDimtoTwoDim(int size, string objects[], string TwoDim[][8])

{

for (int row1 = 0, oneDimPos1 = 0; row1 < size; row1++) { // make a one dim into a two dim.

for (int col1 = 0; col1 < size; col1++, oneDimPos1++) {

TwoDim[row1][col1] = objects[oneDimPos1];

}

}

}

int ChooseRow(int size) {

int row = 9;

while (!(row >= 0 && row <= size - 1)) { // input validation.

cout << "Please choose a valid row: ";

cin >> row;

}

return row;

}

int ChooseCol(int size) {

int col = 9;

while (!(col >= 0 && col <= size - 1)) { // input validation.

cout << "Please choose a valid column: ";

cin >> col;

}

return col;

}

void printBoard(string board[][8], int size) {

for (int l = 0; l < size; l++) { // print the board.

cout << setw(10) << l;

}

cout << endl;

cout << "------------------------------------------------------------" << endl;

for (int row = 0; row < size; row++) {

cout << " " << row << "|";

for (int col = 0; col < size; col++) {

cout << setw(9) << board[row][col] << " ";

}

cout << endl;

}

}