Project 1

<Tic-Tac-Toe>

Name: Huda Milbes

Subject: Intro to C++

Date: 7 / 18 / 2014

**Introduction**

**Title:** Tic-Tac-Toe

**How to play the game:** Tic-Tac-Toe is a game in which there is a 3 x 3 grid. The grid is filled with X’s and O’s. Player 1 is X and chooses a box to place his/her X in. After player 1 chooses a box, player 2 chooses a different box to place his/her O in. This goes on until there is a horizontal, vertical, or diagonal line of all X’s or O’s. The player that accomplishes this first is our winner. If all boxes become full and there is no horizontal, vertical, or diagonal line with all X’s or O’s, the game has no winner.

**Pseudo code**

*Initialize*

*Draw tic-tac-toe board*

*Fill vector a – i*

*Ask player where to put his/her mark*

*If available*

*Place mark*

*If not available*

*Ask player where to put his/her mark*

*Repeat until board is full*

*Output winner / no winner*

**Program**

#include <vector>

#include <iostream>

using namespace std;

const bool CLEAR\_SCREEN = true;

// Utilizes an escape character sequence to clear the screen

void clearScreen()

{

cout << endl;

if (CLEAR\_SCREEN)

{

cout << "\033c";

}

cout << endl;

}

// Draws the provided tic-tac-toe board to the screen

void drawBoard(const vector < char >&board)

{

clearScreen();

for (int i = 0; i < 9; i += 3)

{

cout << " " << board.at(i) << " | " << board.at(i + 1) << " | "

<< board.at(i + 2) << " " << endl;

if (i < 6)

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

}

cout << endl;

}

// Fill vector with characters starting at lower case a.

// If the vector is size 3 then it will have characters a to c.

// If the vector is size 5 then it will have characters a to e.

// If the vector is size 26 then it will have characters a to z.

// Vector size will never be over 26

void initVector(vector <char> &v)

{

// Implement Function

for (int i = 0; i < v.size(); i++)

{

v.at(i) = 97 + i;

}

}

// Convert a character representing a cell to associated vector index

// Position is converted to a vector index

// Integer index in the vector, should be 0 to (vector size - 1)

int convertPosition(char position)

{

// Implement function

if(position == 'a')

{

return 0;

}

else if(position == 'b')

{

return 1;

}

else if(position == 'c')

{

return 2;

}

else if(position == 'd')

{

return 3;

}

else if(position == 'e')

{

return 4;

}

else if(position == 'f')

{

return 5;

}

else if(position == 'g')

{

return 6;

}

else if(position == 'h')

{

return 7;

}

else if(position == 'i')

{

return 8;

}

else

{

return -1;

}

}

// Indicate function to determine if a spot on the board is available.

bool validPlacement(const vector <char> &board, int position)

{

// Implement function

if( (position < 0) || (position > 8) )

{

return false;

}

if(board.at(position) != 'X' && board.at(position) != 'O')

{

return true;

}

else

{

return false;

}

}

// Acquire a play from the user as to where to put her mark

int getPlay(const vector <char> &board)

{

// Implement function

char choice = ' ';

int choice\_num = 0;

bool validPosition = false;

while (!validPosition)

{

cout << "Please choose a position: " << endl;

cin >> choice;

choice\_num = convertPosition(choice);

validPosition = validPlacement(board, choice\_num);

}

return choice\_num;

}

// Indicates function to determine if the game has been won

// Winning conditions in tic-tac-toe require three marks from same

// player in a single row, column or diagonal.

bool gameWon(const vector <char> &board)

{

// Implement function

if ( (board.at(0) == board.at(1) ) && (board.at(1) == board.at(2) ) )

{

return true;

}

else if ( (board.at(3) == board.at(4) ) && (board.at(4) == board.at(5) ) )

{

return true;

}

else if ( (board.at(6) == board.at(7) ) && (board.at(7) == board.at(8) ) )

{

return true;

}

else if ( (board.at(0) == board.at(3) ) && (board.at(3) == board.at(6) ) )

{

return true;

}

else if ( (board.at(1) == board.at(4) ) && (board.at(4) == board.at(7) ) )

{

return true;

}

else if ( (board.at(2) == board.at(5) ) && (board.at(5) == board.at(8) ) )

{

return true;

}

else if ( (board.at(0) == board.at(4) ) && (board.at(4) == board.at(8) ) )

{

return true;

}

else if ( (board.at(2) == board.at(4) ) && (board.at(4) == board.at(6) ) )

{

return true;

}

else

{

return false;

}

}

//Indicates function to determine if the board is full

bool boardFull(const vector <char> &board)

{

// Implement function

for (int i = 0; i < board.size(); i++)

{

if ( ( board.at(i) != 'X' ) && ( board.at(i) != 'O' ) )

{

return false;

}

}

return true;

}

// Global constants for player representation

const int PLAYER1 = 0;

const int PLAYER2 = 1;

int main(int argc, char\*\* argv) {

// Variables that you may find useful to utilize

vector <char> board(9);

int curPlay = 0;

int turn = PLAYER1; // Player 1 always goes first and is 'X'

bool fullBoard = false;

bool gameNotWon = false;

int positionCell = 0;

bool player1wins = false;

bool player2wins = false;

bool gameEnd = false;

// Initialize board to empty state

initVector( board );

// Display empty board

drawBoard ( board );

// Play until game is over

while ( (!gameEnd) && (!player1wins && !player2wins) )

{

// Get a play

positionCell = getPlay (board);

// Set the play on the board

if (turn == PLAYER1)

{

board.at( positionCell ) = 'X';

}

else if ( turn == PLAYER2 )

{

board.at ( positionCell ) = 'O';

}

// Switch the turn to the other player

if (turn == PLAYER1)

{

turn = PLAYER2;

}

else if (turn == PLAYER2)

{

turn = PLAYER1;

}

// Output the updated board

drawBoard ( board );

{

fullBoard = boardFull (board);

gameEnd = gameWon( board );

if (fullBoard && turn == PLAYER1)

{

player1wins = true;

}

else if (fullBoard && turn == PLAYER2)

{

player2wins = true;

}

else if( fullBoard )

{

gameEnd = true;

gameNotWon = true;

}

}

}

// Determine winner and output appropriate message

if (player1wins)

{

cout << "Player 1 Wins!" << endl;

}

else if (player2wins)

{

cout << "Player 2 Wins!" << endl;

}

else if (gameNotWon)

{

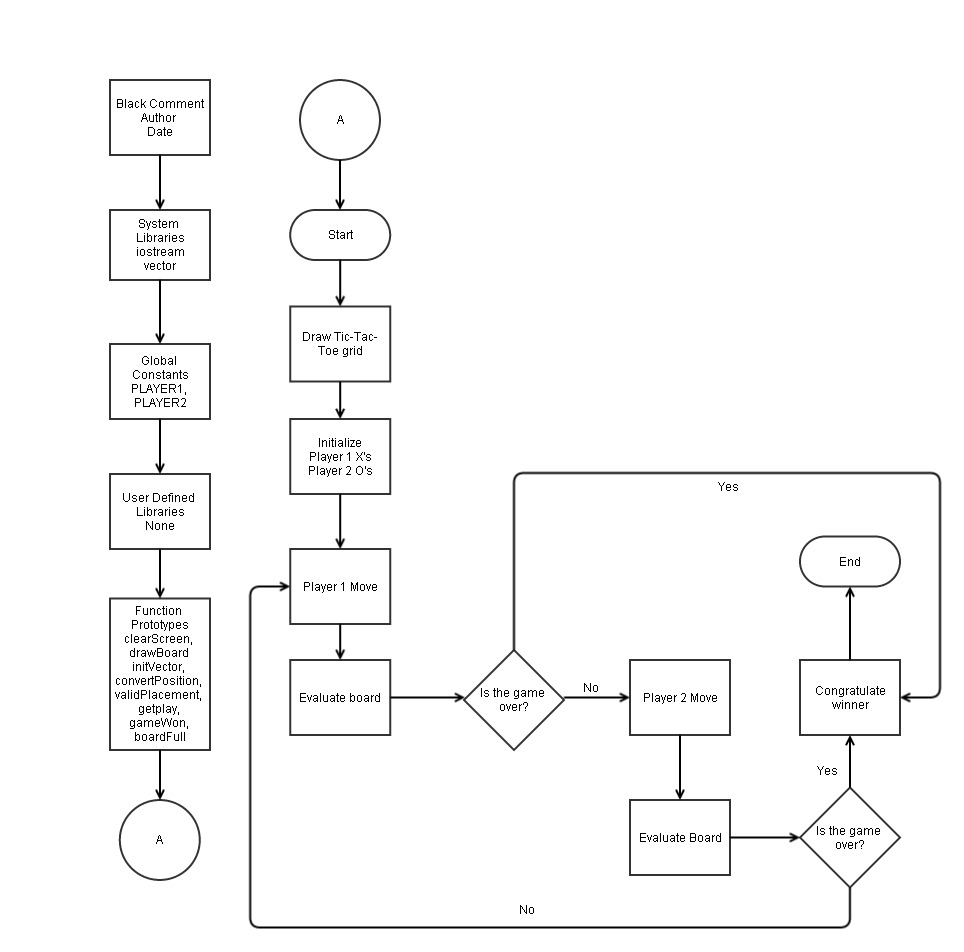
cout << "Nobody Wins!";

}

return 0;

}

**Flowchart**

**