# Exercise 1: TicTacToe

Write a two-player tic-tac-toe program. Prompt the players one at a time for their move (a number 1 through 9 for squares left to right and top to bottom). Print the board after each move. Make sure moves are legal and detect a winning move and congratulate the winner!

Suggestions:

1. Use a global variable array of 9 ints to represent the 9 squares left to right and top to bottom. Initialize the array to all 0’s. The values of the squares are the states of the squares:
   1. 0 means empty
   2. 1 means X
   3. 2 means Y
2. Create a “printBoard” function that prints the state of the board on the screen. Use “|” and “-“ to make an ascii-art image of the screen. Use a helper function to translate 0,1, and 2 to SPACE, X, and Y.
3. Create a “getLegalMove” function that asks the player for a square number and returns the move if the square is valid. Otherwise the function prints an error and asks until the move is legal.
4. Add an infinite loop to main and ask player 1 for a move and then player 2 for a move. Set the state of the board array after every move.
5. Create a “checkForWin” function that returns true if the board is in a “win” state and false if not. You can be as fancy as you like here or make a brute-force 16-if routine.
6. Add check-for-win after every move.

## Step 1 and 2

#include <iostream>

using namespace std;

int board[] = {0,0,0, 0,0,0, 0,0,0};

char getPrintCharacter(int contents) {

if(contents==0) return ' ';

if(contents==1) return 'X';

if(contents==2) return 'O';

return '?';

}

void printBoard() {

cout << getPrintCharacter(board[0]) << " | "

<< getPrintCharacter(board[1]) << " | "

<< getPrintCharacter(board[2]) << endl;

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

cout << getPrintCharacter(board[3]) << " | "

<< getPrintCharacter(board[4]) << " | "

<< getPrintCharacter(board[5]) << endl;

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

cout << getPrintCharacter(board[6]) << " | "

<< getPrintCharacter(board[7]) << " | "

<< getPrintCharacter(board[8]) << endl;

}

void main() {

printBoard();

system("pause");

}

## Step 3

int getLegalMove() {

int move;

while(true) {

cout << "Pick a square: ";

cin >> move;

if(move>=1 && move<=9 && board[move]==0) {

return move-1; // Return 0 to 8

}

cout << "Invalid move" << endl;

}

}

## Step 4

void main() {

while(true) {

cout << endl;

printBoard();

cout << "Your move player 1 (X)." << endl;

int m = getLegalMove();

board[m] = 1; // 1 for X

cout << endl;

printBoard();

cout << "Your move player 2 (O)." << endl;

m = getLegalMove();

board[m] = 2; // 2 for O

}

system("pause");

}

## Step 5

bool checkForWin() {

for(int p=1;p<3;++p) {

// Horizontal

if(board[0]==p && board[1]==p && board[2]==p) return true;

if(board[3]==p && board[4]==p && board[5]==p) return true;

if(board[6]==p && board[7]==p && board[8]==p) return true;

// Vertical

if(board[0]==p && board[3]==p && board[6]==p) return true;

if(board[1]==p && board[4]==p && board[7]==p) return true;

if(board[2]==p && board[5]==p && board[8]==p) return true;

// Diagonal

if(board[0]==p && board[4]==p && board[8]==p) return true;

if(board[2]==p && board[4]==p && board[6]==p) return true;

}

return false;

}

## Step 6

cout << endl;

printBoard();

cout << "Your move player 1 (X)." << endl;

int m = getLegalMove();

board[m] = 1; // 1 for X

if(checkForWin()) {

cout << endl << "X wins!" << endl << endl;

break;

}

## Final

#include <iostream>

using namespace std;

// Use a flat array for the board. 0=empty, 1=X, 2=O

int board[] = {0,0,0, 0,0,0, 0,0,0};

/\*\*

\* This fuction maps a board-square state to a printed

\* character.

\* @param contents the contents of the square

\* @return the character representation to print

\*/

char getPrintCharacter(int contents) {

if(contents==0) return ' ';

if(contents==1) return 'X';

if(contents==2) return 'O';

return '?';

}

/\*\*

\* This function prints the tic-tac-toe game board on the screen

\* in ASCII art.

\*/

void printBoard() {

cout << getPrintCharacter(board[0]) << " | "

<< getPrintCharacter(board[1]) << " | "

<< getPrintCharacter(board[2]) << endl;

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

cout << getPrintCharacter(board[3]) << " | "

<< getPrintCharacter(board[4]) << " | "

<< getPrintCharacter(board[5]) << endl;

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

cout << getPrintCharacter(board[6]) << " | "

<< getPrintCharacter(board[7]) << " | "

<< getPrintCharacter(board[8]) << endl;

}

/\*\*

\* This function gets a legal move from the player. The function prints an

\* error and loops until the player enters a valid move.

\* @return the move 0-8

\*/

int getLegalMove() {

int move;

while(true) {

cout << "Pick a square: ";

cin >> move;

// TODO more detailed reporting here:

// - The move is not valid (must be 1-8)

// - The square is taken

if(move>=1 && move<=9 && board[move-1]==0) {

return move-1; // Return 0 to 8

}

cout << "Invalid move" << endl;

}

}

/\*\*

\* This method checks the game board for a win configuration. If

\* the board is a "win" the last player to move just won.

\* @return true if won or false if not

\*/

bool checkForWin() {

for(int p=1;p<3;++p) { // Check player 1 then player 2

// Horizontal

if(board[0]==p && board[1]==p && board[2]==p) return true;

if(board[3]==p && board[4]==p && board[5]==p) return true;

if(board[6]==p && board[7]==p && board[8]==p) return true;

// Vertical

if(board[0]==p && board[3]==p && board[6]==p) return true;

if(board[1]==p && board[4]==p && board[7]==p) return true;

if(board[2]==p && board[5]==p && board[8]==p) return true;

// Diagonal

if(board[0]==p && board[4]==p && board[8]==p) return true;

if(board[2]==p && board[4]==p && board[6]==p) return true;

}

return false;

}

void main() {

while(true) {

cout << endl;

printBoard();

cout << "Your move player 1 (X)." << endl;

int m = getLegalMove();

board[m] = 1; // 1 for X

if(checkForWin()) {

cout << endl << "X wins!" << endl << endl;

break;

}

cout << endl;

printBoard();

cout << "Your move player 2 (O)." << endl;

m = getLegalMove();

board[m] = 2; // 2 for O

if(checkForWin()) {

cout << endl << "O wins!" << endl << endl;

break;

}

}

printBoard();

system("pause");

}