Homework 9

Version 3 Source Code

// Preston Knibbe

// Version 3

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

class TicTacToe {

public:

TicTacToe();

void driver();

private:

char board[3][3] = {{'1', '4', '7'}, {'2', '5', '8'}, {'3', '6', '9'}};

char whoStarted;

char whoDidntStart;

int turnCount;

char winner;

void printBoard();

bool placeCheck(int, int);

void whoStarts();

bool winCheck();

void Rules(char);

void printMove(char, int, int);

int randomNum(int,int);

void turn(char);

bool block(char);

bool forTheWin(char);

void prompt(char);

char player1;

bool firstMove;

const int TOTAL\_TURNS = 9;

};

TicTacToe::TicTacToe() {

srand(time(0));

whoStarts();

turnCount = 0;

}

int TicTacToe::randomNum(int startNum, int endNum) {

int random = rand() % endNum + startNum;

return random;

}

void TicTacToe::whoStarts() {

if (randomNum(1,2)==1) {

whoStarted = 'X';

whoDidntStart = 'O';

} else {

whoStarted = 'O';

whoDidntStart = 'X';

}

}

void TicTacToe::turn(char player) {

bool test = true;

while (test) {

int xCord = randomNum(0,3);

int yCord = randomNum(0,3);

if (placeCheck(xCord,yCord)) {

printMove(player, xCord, yCord);

board[xCord][yCord] = player;

test = false;

}

}

}

void TicTacToe::prompt(char player) {

char num;

cout << "\n";

cout << "Where is your move?" << endl;

cin >> num;

for (int a = 0; a < 3; a++) {

for (int b = 0; b < 3; b++) {

if (num == board[a][b]) {

if (placeCheck(a,b)) {

board[a][b] = player;

}

}

}

}

}

void TicTacToe::printMove(char player, int x, int y) {

cout << "\n";

cout << player << " marked " << board[x][y] << endl;

cout << "\n";

}

bool TicTacToe::block(char player) {

// Winning move block

//Vertical Block

for (int a = 0; a < 3; a++) {

if (board[a][0] == board[a][1] &&

board[a][0] != player &&

placeCheck(a, 2)) {

printMove(player, a, 2);

board[a][2] = player;

return true;

} else if (board[a][1] == board[a][2] &&

board[a][1] != player &&

placeCheck(a, 0)) {

printMove(player, a, 0);

board[a][0] = player;

return true;

} else if (board[a][0] == board[a][2] &&

board[a][0] != player &&

placeCheck(a, 1)) {

printMove(player, a, 1);

board[a][1] = player;

return true;

}

}

//Horizontal Block

for (int a = 0; a < 3; a++) {

if (board[0][a] == board[1][a] &&

board[0][a] != player &&

placeCheck(2, a)) {

printMove(player, 2, a);

board[2][a] = player;

return true;

} else if (board[1][a] == board[2][a] &&

board[1][a] != player &&

placeCheck(0, a)) {

printMove(player, 0, a);

board[0][a] = player;

return true;

} else if (board[0][a] == board[2][a] &&

board[0][a] != player &&

placeCheck(1, a)) {

printMove(player, 1, a);

board[1][a] = player;

return true;

}

}

//Diagonal Blocks

if (board[0][0] == board[1][1] &&

board[0][0] != player &&

placeCheck(2, 2)) {

printMove(player, 2, 2);

board[2][2] = player;

return true;

} else if (board[2][2] == board [1][1] &&

board[2][2] != player &&

placeCheck(0, 0)) {

printMove(player, 0, 0);

board[0][0] = player;

return true;

} else if (board[0][2] == board [1][1] &&

board[0][2] != player &&

placeCheck(2, 0)) {

printMove(player, 2, 0);

board[2][0] = player;

return true;

} else if (board[2][0] == board [1][1] &&

board[2][0] != player &&

placeCheck(0, 2)) {

printMove(player, 0, 2);

board[0][2] = player;

return true;

} else if (board[0][0] == board [2][0] &&

board[0][0] != player &&

placeCheck(1, 1)) {

printMove(player, 1, 1);

board[1][1] = player;

return true;

} else if (board[0][0] == board [2][2] &&

board[0][0] != player &&

placeCheck(1, 1)) {

printMove(player, 1, 1);

board[1][1] = player;

return true;

}

return false;

}

bool TicTacToe::forTheWin(char player) {

// Win set up

// Attempt at vertical win

for (int a = 0; a < 3; a++) {

if (board[a][0] == board[a][1] &&

board[a][0] == player &&

placeCheck(a, 2)) {

printMove(player, a, 2);

board[a][2] = player;

return true;

} else if (board[a][1] == board[a][2] &&

board[a][1] == player &&

placeCheck(a, 0)) {

printMove(player, a, 0);

board[a][0] = player;

return true;

} else if (board[a][0] == board[a][2] &&

board[a][0] == player &&

placeCheck(a, 1)) {

printMove(player, a, 1);

board[a][1] = player;

return true;

}

}

// Attempt at horizontal win

for (int a = 0; a < 3; a++) {

if (board[0][a] == board[1][a] &&

board[0][a] == player &&

placeCheck(2, a)) {

printMove(player, 2, a);

board[2][a] = player;

return true;

} else if (board[1][a] == board[2][a] &&

board[1][a] == player &&

placeCheck(0, a)) {

printMove(player, 0, a);

board[0][a] = player;

return true;

} else if (board[0][a] == board[2][a] &&

board[0][a] == player &&

placeCheck(1, a)) {

printMove(player, 1, a);

board[1][a] = player;

return true;

}

}

// Attempt at diagonal win

if (board[0][0] == board[1][1] &&

board[0][0] == player &&

placeCheck(2, 2)) {

printMove(player, 2, 2);

board[2][2] = player;

return true;

} else if (board[2][2] == board [1][1] &&

board[2][2] == player &&

placeCheck(0, 0)) {

printMove(player, 0, 0);

board[0][0] = player;

return true;

} else if (board[0][2] == board [1][1] &&

board[0][2] == player &&

placeCheck(2, 0)) {

printMove(player, 2, 0);

board[2][0] = player;

return true;

} else if (board[2][0] == board [1][1] &&

board[2][0] == player &&

placeCheck(0, 2)) {

printMove(player, 0, 2);

board[0][2] = player;

return true;

}

return false;

}

void TicTacToe::Rules(char player) {

bool test = true;

// First player chooses middle of board

if (turnCount==0) {

printMove(player, 1, 1);

board[1][1] = player;

test = false;

}

// Makes the second turn a corner pick

if (turnCount==1) {

int rand = randomNum(1,4);

if (rand==1 && placeCheck(0,0)) {

printMove(player, 0, 0);

board[0][0] = player;

test = false;

} else if (rand==2 && placeCheck(2, 0)) {

printMove(player, 2, 0);

board[2][0] = player;

test = false;

} else if (rand==3 && placeCheck(0, 2)) {

printMove(player, 0, 2);

board[0][2] = player;

test = false;

} else if (rand==4 && placeCheck(2, 2)) {

printMove(player, 2, 2);

board[2][2] = player;

test = false;

}

}

if (block(player)) {

} else if (forTheWin(player)) {

} else if (test) {

turn(player);

}

}

bool TicTacToe::placeCheck(int x, int y) {

if (board[x][y]=='O' || board[x][y]=='X') {

return false;

} else {

return true;

}

}

void TicTacToe::driver() {

cout << "Choose X or O: " << endl;

cin >> player1;

if (player1 == whoStarted) {

cout << "\n";

cout << "You get first move!" << endl;

cout << "\n";

printBoard();

firstMove = true;

} else {

cout << "\n";

cout << "The computer gets first move!" << endl;

cout << "\n";

firstMove = false;

}

while (!winCheck() && turnCount < TOTAL\_TURNS) {

if (firstMove) {

if ((turnCount%2)==0 || turnCount==0) {

prompt(player1);

} else {

Rules(whoDidntStart);

}

printBoard();

turnCount++;

} else {

if ((turnCount%2)==0 || turnCount==0) {

Rules(whoStarted);

} else {

prompt(player1);

}

printBoard();

turnCount++;

}

}

if(turnCount == (TOTAL\_TURNS) && !winCheck()) {

cout << "\n";

cout << "Cats!" << endl;

cout << "\n";

printBoard();

} else if (winCheck()) {

cout << "\n\n";

cout << "The winner is " << winner << endl;

cout << "\n";

printBoard();

}

}

bool TicTacToe::winCheck() {

// Vertical Check

for (int a = 0; a < 3; a++) {

if (board[a][0]==board[a][1] &&

board[a][1]==board[a][2]) {

winner = board[a][0];

return true;

}

}

// Horizontal Check

for (int b = 0; b < 3; b++) {

if (board[0][b]==board[1][b] &&

board[1][b]==board[2][b]) {

winner = board[0][b];

return true;

}

}

// Diagonal Check

if ((board[0][0]==board[1][1] && board[1][1]==board[2][2]) ||

(board[0][2]==board[1][1] && board[1][1]==board[2][0])) {

winner = board[1][1];

return true;

}

return false;

}

void TicTacToe::printBoard() {

cout << "\n";

for (int a = 0; a < 3; a++) {

cout << " " << board[0][a] << " | "

<< board[1][a] << " | "

<< board[2][a] << " "

<< endl;

if (a<2) {

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

}

}

cout << "\n";

}

int main()

{

TicTacToe Game1;

Game1.driver();

char playAgain;

cout << "Would you like to play again? (Y/N)" << endl;

cin >> playAgain;

if (playAgain == 'Y') {

TicTacToe Game2;

Game2.driver();

}

return 0;

}

Version 2 Source Code

// Preston Knibbe

// Version 2

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

class TicTacToe {

public:

TicTacToe();

void driver();

private:

char board[3][3] = {{'1', '4', '7'}, {'2', '5', '8'}, {'3', '6', '9'}};

char whoStarted;

char whoDidntStart;

int turnCount;

char winner;

void printBoard();

bool placeCheck(int, int);

void whoStarts();

bool winCheck();

void Rules(char);

void printMove(char, int, int);

int randomNum(int,int);

void turn(char);

bool block(char);

bool forTheWin(char);

const int TOTAL\_TURNS = 9;

};

TicTacToe::TicTacToe() {

srand(time(0));

whoStarts();

turnCount = 0;

cout << whoStarted << " goes first!" << endl;

}

int TicTacToe::randomNum(int startNum, int endNum) {

int random = rand() % endNum + startNum;

return random;

}

void TicTacToe::whoStarts() {

if (randomNum(1,2)==1) {

whoStarted = 'X';

whoDidntStart = 'O';

} else {

whoStarted = 'O';

whoDidntStart = 'X';

}

}

void TicTacToe::turn(char player) {

bool test = true;

while (test) {

int xCord = randomNum(0,3);

int yCord = randomNum(0,3);

if (placeCheck(xCord,yCord)) {

printMove(player, xCord, yCord);

board[xCord][yCord] = player;

test = false;

}

}

}

void TicTacToe::printMove(char player, int x, int y) {

cout << player << " marked " << board[x][y] << endl;

}

bool TicTacToe::block(char player) {

// Winning move block

//Vertical Block

for (int a = 0; a < 3; a++) {

if (board[a][0] == board[a][1] &&

board[a][0] != player &&

placeCheck(a, 2)) {

printMove(player, a, 2);

board[a][2] = player;

return true;

} else if (board[a][1] == board[a][2] &&

board[a][1] != player &&

placeCheck(a, 0)) {

printMove(player, a, 0);

board[a][0] = player;

return true;

} else if (board[a][0] == board[a][2] &&

board[a][0] != player &&

placeCheck(a, 1)) {

printMove(player, a, 1);

board[a][1] = player;

return true;

}

}

//Horizontal Block

for (int a = 0; a < 3; a++) {

if (board[0][a] == board[1][a] &&

board[0][a] != player &&

placeCheck(2, a)) {

printMove(player, 2, a);

board[2][a] = player;

return true;

} else if (board[1][a] == board[2][a] &&

board[1][a] != player &&

placeCheck(0, a)) {

printMove(player, 0, a);

board[0][a] = player;

return true;

} else if (board[0][a] == board[2][a] &&

board[0][a] != player &&

placeCheck(1, a)) {

printMove(player, 1, a);

board[1][a] = player;

return true;

}

}

//Diagonal Blocks

if (board[0][0] == board[1][1] &&

board[0][0] != player &&

placeCheck(2, 2)) {

printMove(player, 2, 2);

board[2][2] = player;

return true;

} else if (board[2][2] == board [1][1] &&

board[2][2] != player &&

placeCheck(0, 0)) {

printMove(player, 0, 0);

board[0][0] = player;

return true;

} else if (board[0][2] == board [1][1] &&

board[0][2] != player &&

placeCheck(2, 0)) {

printMove(player, 2, 0);

board[2][0] = player;

return true;

} else if (board[2][0] == board [1][1] &&

board[2][0] != player &&

placeCheck(0, 2)) {

printMove(player, 0, 2);

board[0][2] = player;

return true;

}

return false;

}

bool TicTacToe::forTheWin(char player) {

// Win set up

// Attempt at vertical win

for (int a = 0; a < 3; a++) {

if (board[a][0] == board[a][1] &&

board[a][0] == player &&

placeCheck(a, 2)) {

printMove(player, a, 2);

board[a][2] = player;

return true;

} else if (board[a][1] == board[a][2] &&

board[a][1] == player &&

placeCheck(a, 0)) {

printMove(player, a, 0);

board[a][0] = player;

return true;

} else if (board[a][0] == board[a][2] &&

board[a][0] == player &&

placeCheck(a, 1)) {

printMove(player, a, 1);

board[a][1] = player;

return true;

}

}

// Attempt at horizontal win

for (int a = 0; a < 3; a++) {

if (board[0][a] == board[1][a] &&

board[0][a] == player &&

placeCheck(2, a)) {

printMove(player, 2, a);

board[2][a] = player;

return true;

} else if (board[1][a] == board[2][a] &&

board[1][a] == player &&

placeCheck(0, a)) {

printMove(player, 0, a);

board[0][a] = player;

return true;

} else if (board[0][a] == board[2][a] &&

board[0][a] == player &&

placeCheck(1, a)) {

printMove(player, 1, a);

board[1][a] = player;

return true;

}

}

// Attempt at diagonal win

if (board[0][0] == board[1][1] &&

board[0][0] == player &&

placeCheck(2, 2)) {

printMove(player, 2, 2);

board[2][2] = player;

return true;

} else if (board[2][2] == board [1][1] &&

board[2][2] == player &&

placeCheck(0, 0)) {

printMove(player, 0, 0);

board[0][0] = player;

return true;

} else if (board[0][2] == board [1][1] &&

board[0][2] == player &&

placeCheck(2, 0)) {

printMove(player, 2, 0);

board[2][0] = player;

return true;

} else if (board[2][0] == board [1][1] &&

board[2][0] == player &&

placeCheck(0, 2)) {

printMove(player, 0, 2);

board[0][2] = player;

return true;

}

return false;

}

void TicTacToe::Rules(char player) {

bool test = true;

// First player chooses middle of board

if (turnCount==0) {

printMove(player, 1, 1);

board[1][1] = player;

test = false;

}

// Makes the second turn a corner pick

if (turnCount==1) {

int rand = randomNum(1,4);

if (rand==1 && placeCheck(0,0)) {

printMove(player, 0, 0);

board[0][0] = player;

test = false;

} else if (rand==2 && placeCheck(2, 0)) {

printMove(player, 2, 0);

board[2][0] = player;

test = false;

} else if (rand==3 && placeCheck(0, 2)) {

printMove(player, 0, 2);

board[0][2] = player;

test = false;

} else if (rand==4 && placeCheck(2, 2)) {

printMove(player, 2, 2);

board[2][2] = player;

test = false;

}

}

if (block(player)) {

} else if (forTheWin(player)) {

} else if (test) {

turn(player);

}

}

bool TicTacToe::placeCheck(int x, int y) {

if (board[x][y]=='O' || board[x][y]=='X') {

return false;

} else {

return true;

}

}

void TicTacToe::driver() {

while (!winCheck() && turnCount < TOTAL\_TURNS) {

if ((turnCount%2)==0 || turnCount==0) {

Rules(whoStarted);

} else {

Rules(whoDidntStart);

}

turnCount++;

}

if(turnCount == (TOTAL\_TURNS) && !winCheck()) {

cout << "Cats!" << endl;

printBoard();

} else if (winCheck()) {

cout << "The winner is " << winner << endl;

printBoard();

}

}

bool TicTacToe::winCheck() {

// Vertical Check

for (int a = 0; a < 3; a++) {

if (board[a][0]==board[a][1] &&

board[a][1]==board[a][2]) {

winner = board[a][0];

return true;

}

}

// Horizontal Check

for (int b = 0; b < 3; b++) {

if (board[0][b]==board[1][b] &&

board[1][b]==board[2][b]) {

winner = board[0][b];

return true;

}

}

// Diagonal Check

if ((board[0][0]==board[1][1] && board[1][1]==board[2][2]) ||

(board[0][2]==board[1][1] && board[1][1]==board[2][0])) {

winner = board[1][1];

return true;

}

return false;

}

void TicTacToe::printBoard() {

for (int a = 0; a < 3; a++) {

cout << " " << board[0][a] << " | "

<< board[1][a] << " | "

<< board[2][a] << " "

<< endl;

if (a<2) {

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

}

}

}

int main()

{

TicTacToe Game1;

Game1.driver();

return 0;

}

Version 1 Source Code

// Preston Knibbe

// Version 1

#include <iostream>

#include <stdlib.h>

#include <time.h>

using namespace std;

class TicTacToe {

public:

TicTacToe();

void driver();

private:

char board[3][3] = {{'1', '4', '7'}, {'2', '5', '8'}, {'3', '6', '9'}};

char whoStarted;

char whoDidntStart;

int turnCount;

char winner;

void printBoard();

bool placeCheck(int, int);

void whoStarts();

bool winCheck();

int randomNum(int,int);

void turn(char);

const int TOTAL\_TURNS = 9;

};

TicTacToe::TicTacToe() {

srand(time(0));

whoStarts();

turnCount = 0;

cout << whoStarted << " goes first!" << endl;

}

int TicTacToe::randomNum(int startNum, int endNum) {

startNum += 1;

endNum += 1;

int random = rand() % endNum + startNum;

random -= 1;

return random;

}

void TicTacToe::whoStarts() {

if (randomNum(1,2)==1) {

whoStarted = 'X';

whoDidntStart = 'O';

} else {

whoStarted = 'O';

whoDidntStart = 'X';

}

}

void TicTacToe::turn(char player) {

bool test = true;

while (test) {

int xCord = randomNum(0,2);

int yCord = randomNum(0,2);

if (placeCheck(xCord,yCord)) {

cout << player << " marked " << board[xCord][yCord] << endl;

board[xCord][yCord] = player;

test = false;

}

}

}

bool TicTacToe::placeCheck(int x, int y) {

if (board[x][y]=='O' || board[x][y]=='X') {

return false;

} else {

return true;

}

}

void TicTacToe::driver() {

while (!winCheck() && turnCount < TOTAL\_TURNS) {

if ((turnCount%2)==0 || turnCount==0) {

turn(whoStarted);

} else {

turn(whoDidntStart);

}

turnCount++;

}

if(turnCount == (TOTAL\_TURNS) && !winCheck()) {

cout << "Cats!" << endl;

printBoard();

} else if (winCheck()) {

cout << "The winner is " << winner << endl;

printBoard();

}

}

bool TicTacToe::winCheck() {

// Vertical Check

for (int a = 0; a < 3; a++) {

if (board[a][0]==board[a][1] &&

board[a][1]==board[a][2]) {

winner = board[a][0];

return true;

}

}

// Horizontal Check

for (int b = 0; b < 3; b++) {

if (board[0][b]==board[1][b] &&

board[1][b]==board[2][b]) {

winner = board[0][b];

return true;

}

}

// Diagonal Check

if ((board[0][0]==board[1][1] && board[1][1]==board[2][2]) ||

(board[0][2]==board[1][1] && board[1][1]==board[2][0])) {

winner = board[1][1];

return true;

}

return false;

}

void TicTacToe::printBoard() {

for (int a = 0; a < 3; a++) {

cout << " " << board[0][a] << " | "

<< board[1][a] << " | "

<< board[2][a] << " "

<< endl;

if (a<2) {

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

}

}

}

int main()

{

TicTacToe Game1;

Game1.driver();

return 0;

}

Code Output from Version 3

Choose X or O:

X

You get first move!

1 | 2 | 3

-----------

4 | 5 | 6

-----------

7 | 8 | 9

Where is your move?

1

X | 2 | 3

-----------

4 | 5 | 6

-----------

7 | 8 | 9

O marked 3

X | 2 | O

-----------

4 | 5 | 6

-----------

7 | 8 | 9

Where is your move?

9

X | 2 | O

-----------

4 | 5 | 6

-----------

7 | 8 | X

O marked 5

X | 2 | O

-----------

4 | O | 6

-----------

7 | 8 | X

Where is your move?

7

X | 2 | O

-----------

4 | O | 6

-----------

X | 8 | X

O marked 4

X | 2 | O

-----------

O | O | 6

-----------

X | 8 | X

Where is your move?

8

X | 2 | O

-----------

O | O | 6

-----------

X | X | X

The winner is X

X | 2 | O

-----------

O | O | 6

-----------

X | X | X

Would you like to play again? (Y/N)

Y

Choose X or O:

O

The computer gets first move!

X marked 5

1 | 2 | 3

-----------

4 | X | 6

-----------

7 | 8 | 9

Where is your move?

1

O | 2 | 3

-----------

4 | X | 6

-----------

7 | 8 | 9

X marked 7

O | 2 | 3

-----------

4 | X | 6

-----------

X | 8 | 9

Where is your move?

3

O | 2 | O

-----------

4 | X | 6

-----------

X | 8 | 9

X marked 2

O | X | O

-----------

4 | X | 6

-----------

X | 8 | 9

Where is your move?

8

O | X | O

-----------

4 | X | 6

-----------

X | O | 9

X marked 9

O | X | O

-----------

4 | X | 6

-----------

X | O | X

Where is your move?

4

O | X | O

-----------

O | X | 6

-----------

X | O | X

X marked 6

O | X | O

-----------

O | X | X

-----------

X | O | X

Cats!

O | X | O

-----------

O | X | X

-----------

X | O | X

Process returned 0 (0x0) execution time : 70.710 s

Press any key to continue.