Федеральное агенство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования

"Сибирский государственный университет телекоммуникаций и информатики"

Курсовая работа

по дисциплине "Программирование"

ВАРИАНТ 30

Выполнил: студент группы ИП - 813

Бурдуковский Илья Александрович

Проверил: ассистент кафедры ПМиК

Дьячкова И.С.

Новосибирск 2018

**Содержание:**

1. **Введение**
2. **Описание алгоритма**
3. **Код программы на C++**
4. **Результат работы программы**

**1. Введение**

**Задача:**

Выполнить Курсовую работу по дисциплине "Программирование" за 1 семестр

**Описание:**

Согласно заданию для моего варианта необходимо Написать программу для игры c компьютером в крестики-нолики

**2. Описание алгоритма**

char pole[3][3] – двумерный массив, для записи состояния клетки

**Функция int oponentturn(int team)**

Просчитывает ходы бота в ситуации хода первым/вторым.

Int team – какую стороны выбрал игрок

int svit - используется для переключения между пунктами меню

float scrX, float scrY – размер окна

int MenuNum – нахождение мыши в области спрайта

int chod – какой ход по счёту делает бот (не считая случаи, где перекрывает выигрышную ситуацию игрока)

int draw – переключатель с условием для ничьи

int g – условие для кнопки continue (нельзя продолжить, если игра закончилась или ещё не начиналась)

int ok – переключатель для хождения

3**. Код программы на C++**

#include <SFML/Graphics.hpp>

#include<Windows.h>

#include <iostream>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

using namespace std;

using namespace sf;

int chod = 0;

int turn(int team);

int oponentturn(int team);

char pole[3][3] = {

{' ',' ',' '},

{' ',' ',' '},

{' ',' ',' '},

};

int main() {

float scrX = GetSystemMetrics(SM\_CXSCREEN);

float scrY = GetSystemMetrics(SM\_CYSCREEN);

RenderWindow window(VideoMode(scrX, scrY), "X-O", Style::Fullscreen);

RectangleShape polosa(Vector2f(20, scrY));

polosa.setFillColor(Color::Black);

polosa.setPosition(scrX / 2 - 5, 0);

int svit = 0;

int team = 0;

int g = 0;

while (window.isOpen())

{

Event event;

while (window.pollEvent(event)) {

if (event.type == Event::Closed)

window.close();

if (Keyboard::isKeyPressed(Keyboard::Escape)) {

window.close();

}

}

if (svit == 0) {

Texture start\_texture, continue\_texture, exit\_texture;

start\_texture.loadFromFile("images/start.png");

continue\_texture.loadFromFile("images/continue.png");

exit\_texture.loadFromFile("images/exit.png");

Sprite start\_sprite(start\_texture), continue\_sprite(continue\_texture), exit\_sprite(exit\_texture);

start\_sprite.setPosition(scrX / 2.5, scrY / 4);

continue\_sprite.setPosition(scrX / 2.8, scrY / 2.7);

exit\_sprite.setPosition(scrX / 2.4, scrY / 2);

int MenuNum = 0;

if (IntRect(scrX / 2.5, (scrY / 4) + 50, 320, 195).contains(Mouse::getPosition(window))) { MenuNum = 1; }

if (IntRect(scrX / 2.8, (scrY / 2.7) + 50, 530, 190).contains(Mouse::getPosition(window))) { MenuNum = 2; }

if (IntRect(scrX / 2.4, (scrY / 2) + 50, 220, 190).contains(Mouse::getPosition(window))) { MenuNum = 3; }

if (Mouse::isButtonPressed(Mouse::Left))

{

if (MenuNum == 1) {

svit = 1;

Sleep(300);

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

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

pole[i][j] = ' ';

}

}

}

if ((MenuNum == 2) && (g != 0)) { svit = 2; Sleep(300); }

if (MenuNum == 3) { window.close(); Sleep(300); }

}

window.clear(Color::White);

window.draw(start\_sprite);

window.draw(continue\_sprite);

window.draw(exit\_sprite);

window.display();

}

if (svit == 1) {

chod = 0;

Texture krestik\_texture, nolik\_texture, choose\_texture, way\_texture, exitmenu\_texture;

krestik\_texture.loadFromFile("images/krestikhead.png");

nolik\_texture.loadFromFile("images/nolikhead.png");

choose\_texture.loadFromFile("images/choose.png");

way\_texture.loadFromFile("images/way.png");

exitmenu\_texture.loadFromFile("images/exitmenu.png");

Sprite krestik\_sprite(krestik\_texture), nolik\_sprite(nolik\_texture), choose\_sprite(choose\_texture), way\_sprite(way\_texture), exitmenu\_sprite(exitmenu\_texture);

krestik\_sprite.setPosition(scrX / 8 \* 5, scrY / 5);

nolik\_sprite.setPosition(scrX / 8, scrY / 5);

choose\_sprite.setPosition(200, 50);

way\_sprite.setPosition(scrX / 2 + 200, 50);

exitmenu\_sprite.setPosition(scrX - 250, scrY - 140);

int MenuNum = 0;

if (IntRect(scrX - 250, scrY - 140, scrX, scrY).contains(Mouse::getPosition(window))) { MenuNum = 1; }

if (IntRect(300, 340, 400, 640).contains(Mouse::getPosition(window))) { MenuNum = 2; }

if (IntRect(1250, 340, 400, 640).contains(Mouse::getPosition(window))) { MenuNum = 3; }

if (Mouse::isButtonPressed(Mouse::Left))

{

//cout << Mouse::getPosition().x;

//cout << Mouse::getPosition().y;

if (MenuNum == 1) { svit = 0; Sleep(300); }

if (MenuNum == 2) { svit = 2; team = 0; pole[1][1] = 'x'; Sleep(300); }

if (MenuNum == 3) { svit = 2; team = 1; Sleep(300); }

}

window.clear(Color::White);

window.draw(polosa);

window.draw(krestik\_sprite);

window.draw(nolik\_sprite);

window.draw(choose\_sprite);

window.draw(way\_sprite);

window.draw(exitmenu\_sprite);

window.display();

}

if (svit == 2) {

int draw = 0;

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

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

if (pole[i][j] == ' ') {

draw = 1;

}

}

}

if (draw == 0) {

svit = 5;

}

g = 1;

Texture krestik\_texture, nolik\_texture, exitmenu\_texture;

krestik\_texture.loadFromFile("images/krestikhead.png");

nolik\_texture.loadFromFile("images/nolikhead.png");

exitmenu\_texture.loadFromFile("images/exitmenu.png");

Sprite krestik\_sprite(krestik\_texture), nolik\_sprite(nolik\_texture), exitmenu\_sprite(exitmenu\_texture);

krestik\_sprite.setPosition(610, 160);

nolik\_sprite.setPosition(600, 375);

exitmenu\_sprite.setPosition(scrX - 250, scrY - 140);

nolik\_sprite.scale(0.5f, 0.5f);

krestik\_sprite.scale(0.5f, 0.5f);

int MenuNum = 0, ok = 0;

if (IntRect(600, 160, 225, 240).contains(Mouse::getPosition())) { MenuNum = 1; }

if (IntRect(840, 160, 225, 240).contains(Mouse::getPosition(window))) { MenuNum = 2; }

if (IntRect(1080, 160, 240, 240).contains(Mouse::getPosition(window))) { MenuNum = 3; }

if (IntRect(600, 420, 225, 235).contains(Mouse::getPosition())) { MenuNum = 4; }

if (IntRect(855, 420, 225, 235).contains(Mouse::getPosition(window))) { MenuNum = 5; }

if (IntRect(1080, 420, 240, 235).contains(Mouse::getPosition(window))) { MenuNum = 6; }

if (IntRect(600, 670, 225, 240).contains(Mouse::getPosition())) { MenuNum = 7; }

if (IntRect(840, 670, 225, 240).contains(Mouse::getPosition(window))) { MenuNum = 8; }

if (IntRect(1080, 670, 240, 240).contains(Mouse::getPosition(window))) { MenuNum = 9; }

if (IntRect(scrX - 250, scrY - 140, scrX, scrY).contains(Mouse::getPosition(window))) { MenuNum = 10; }

if (Mouse::isButtonPressed(Mouse::Left)) {

if (MenuNum == 1) { ok = 1; Sleep(300); }

if (MenuNum == 2) { ok = 1; Sleep(300); }

if (MenuNum == 3) { ok = 1; Sleep(300); }

if (MenuNum == 4) { ok = 1; Sleep(300); }

if (MenuNum == 5) { ok = 1; Sleep(300); }

if (MenuNum == 6) { ok = 1; Sleep(300); }

if (MenuNum == 7) { ok = 1; Sleep(300); }

if (MenuNum == 8) { ok = 1; Sleep(300); }

if (MenuNum == 9) { ok = 1; Sleep(300); }

if (MenuNum == 10) { svit = 0; Sleep(300); }

}

window.clear(Color::White);

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

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

if (pole[i][j] == 'o') {

cout << i << j;

nolik\_sprite.setPosition(615 + (i \* 240), 140 + (245 \* j));

window.draw(nolik\_sprite);

}

if (pole[i][j] == 'x') {

cout << i << j;

krestik\_sprite.setPosition(615 + (i \* 240), 140 + (245 \* j));

window.draw(krestik\_sprite);

}

}

}

int retur = 0;

if (ok == 1) {

switch (team) {

case 0:

if ((MenuNum == 1) && (pole[0][0] == ' ')) {

nolik\_sprite.setPosition(615, 140);

window.draw(nolik\_sprite);

pole[0][0] = 'o';

//retur=turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 2) && (pole[1][0] == ' ')) {

nolik\_sprite.setPosition(855, 140);

window.draw(nolik\_sprite);

pole[1][0] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 3) && (pole[2][0] == ' ')) {

nolik\_sprite.setPosition(1095, 140);

window.draw(nolik\_sprite);

pole[2][0] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 4) && (pole[0][1] == ' ')) {

nolik\_sprite.setPosition(615, 385);

window.draw(nolik\_sprite);

pole[0][1] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 5) && (pole[1][1] == ' ')) {

nolik\_sprite.setPosition(855, 385);

window.draw(nolik\_sprite);

pole[1][1] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 6) && (pole[2][1] == ' ')) {

nolik\_sprite.setPosition(1095, 385);

window.draw(nolik\_sprite);

pole[2][1] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 7) && (pole[0][2] == ' ')) {

nolik\_sprite.setPosition(615, 630);

window.draw(nolik\_sprite);

pole[0][2] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 8) && (pole[1][2] == ' ')) {

nolik\_sprite.setPosition(855, 630);

window.draw(nolik\_sprite);

pole[1][2] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 9) && (pole[2][2] == ' ')) {

nolik\_sprite.setPosition(1095, 630);

window.draw(nolik\_sprite);

pole[2][2] = 'o';

//retur = turn(team);

retur = oponentturn(team);

}

/\*if (retur == 1) {

svit = 6;

Sleep(300);

}\*/

if (retur == 2) {

svit = 7;

}

break;

case 1:

if ((MenuNum == 1) && (pole[0][0] == ' ')) {

krestik\_sprite.setPosition(615, 140);

window.draw(krestik\_sprite);

pole[0][0] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 2) && (pole[1][0] == ' ')) {

krestik\_sprite.setPosition(855, 140);

window.draw(krestik\_sprite);

pole[1][0] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 3) && (pole[2][0] == ' ')) {

krestik\_sprite.setPosition(1095, 140);

window.draw(krestik\_sprite);

pole[2][0] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 4) && (pole[0][1] == ' ')) {

krestik\_sprite.setPosition(615, 385);

window.draw(krestik\_sprite);

pole[0][1] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 5) && (pole[1][1] == ' ')) {

krestik\_sprite.setPosition(855, 385);

window.draw(krestik\_sprite);

pole[1][1] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 6) && (pole[2][1] == ' ')) {

krestik\_sprite.setPosition(1095, 385);

window.draw(krestik\_sprite);

pole[2][1] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 7) && (pole[0][2] == ' ')) {

krestik\_sprite.setPosition(615, 630);

window.draw(krestik\_sprite);

pole[0][2] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 8) && (pole[1][2] == ' ')) {

krestik\_sprite.setPosition(855, 630);

window.draw(krestik\_sprite);

pole[1][2] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

if ((MenuNum == 9) && (pole[2][2] == ' ')) {

krestik\_sprite.setPosition(1095, 630);

window.draw(krestik\_sprite);

pole[2][2] = 'x';

//retur = turn(team);

retur = oponentturn(team);

}

/\*if (retur == 1) {

svit = 6;

}\*/

if (retur == 2) {

svit = 7;

}

break;

}

}

sf::RectangleShape line(Vector2f(740, 15));

line.setFillColor(Color::Black);

line.rotate(90);

line.setPosition(840, 160);

window.draw(line);

line.setPosition(1080, 160);

window.draw(line);

line.rotate(-90);

line.setPosition(600, 405);

window.draw(line);

line.setPosition(600, 655);

window.draw(line);

window.draw(exitmenu\_sprite);

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

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

if (pole[i][j] == 'o') {

cout << i << j;

nolik\_sprite.setPosition(615 + (i \* 240), 140 + (245 \* j));

window.draw(nolik\_sprite);

}

if (pole[i][j] == 'x') {

cout << i << j;

krestik\_sprite.setPosition(615 + (i \* 240), 140 + (245 \* j));

window.draw(krestik\_sprite);

}

}

}

if (svit == 7) {

line.setFillColor(Color::Magenta);

switch (team) {

case 0:

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

if ((pole[0][i] == 'x') && (pole[1][i] == 'x') && (pole[2][i] == 'x')) {

line.setPosition(600, 280 + 125 \* i);

window.draw(line);

}

}

line.rotate(90);

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

if ((pole[i][0] == 'x') && (pole[i][1] == 'x') && (pole[i][2] == 'x')) {

line.setPosition(720 + 240 \* i, 160);

window.draw(line);

}

}

line.rotate(-45);

if ((pole[0][0] == 'x') && (pole[1][1] == 'x') && (pole[2][2] == 'x')) {

line.setPosition(600, 160);

line.setSize(Vector2f(1080, 15));

window.draw(line);

}

line.rotate(90);

if ((pole[2][0] == 'x') && (pole[1][1] == 'x') && (pole[0][2] == 'x')) {

line.setPosition(1320, 160);

line.setSize(Vector2f(1080, 15));

window.draw(line);

}

break;

case 1:

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

if ((pole[0][i] == 'o') && (pole[1][i] == 'o') && (pole[2][i] == 'o')) {

line.setPosition(600, 280 + 125 \* i);

window.draw(line);

}

}

line.rotate(90);

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

if ((pole[i][0] == 'o') && (pole[i][1] == 'o') && (pole[i][2] == 'o')) {

line.setPosition(720 + 240 \* i, 160);

window.draw(line);

}

}

line.rotate(-45);

if ((pole[0][0] == 'o') && (pole[1][1] == 'o') && (pole[2][2] == 'o')) {

line.setPosition(600, 160);

line.setSize(Vector2f(1080, 15));

window.draw(line);

}

line.rotate(90);

if ((pole[2][0] == 'o') && (pole[1][1] == 'o') && (pole[0][2] == 'o')) {

line.setPosition(1320, 160);

line.setSize(Vector2f(1080, 15));

window.draw(line);

}

break;

}

}

window.display();

if ((svit == 5) || (svit == 6) || (svit == 7)) {

Sleep(1000);

}

}

//ничья

if (svit == 5) {

int MenuNum = 0;

Texture playagain\_texture, draw\_texture;

playagain\_texture.loadFromFile("images/play-again.png");

draw\_texture.loadFromFile("images/draw.png");

Sprite playagain\_sprite(playagain\_texture), draw\_sprite(draw\_texture);

draw\_sprite.setPosition(770, 325);

playagain\_sprite.setPosition(770, 540);

if (IntRect(770, 540, 384, 123).contains(Mouse::getPosition(window))) { MenuNum = 1; }

if (Mouse::isButtonPressed(Mouse::Left)) {

if (MenuNum == 1) {

svit = 0;

g = 0;

Sleep(300);

}

}

window.clear(Color::White);

window.draw(draw\_sprite);

window.draw(playagain\_sprite);

window.display();

}

//выигрыш

/\*

if (svit == 6) {

int MenuNum = 0;

Texture playagain\_texture, win\_texture;

playagain\_texture.loadFromFile("images/play-again.png");

win\_texture.loadFromFile("images/win.png");

Sprite playagain\_sprite(playagain\_texture), win\_sprite(win\_texture);

win\_sprite.setPosition(680, 325);

playagain\_sprite.setPosition(770, 540);

if (IntRect(770, 540, 384, 123).contains(Mouse::getPosition(window))) { MenuNum = 1; }

if (Mouse::isButtonPressed(Mouse::Left)) {

if (MenuNum == 1) {

svit = 0;

g = 0;

Sleep(300);

}

}

window.clear(Color::White);

window.draw(win\_sprite);

window.draw(playagain\_sprite);

window.display();

}

\*/

//луз

if (svit == 7) {

int MenuNum = 0;

Texture playagain\_texture, lose\_texture;

playagain\_texture.loadFromFile("images/play-again.png");

lose\_texture.loadFromFile("images/lose.png");

Sprite playagain\_sprite(playagain\_texture), lose\_sprite(lose\_texture);

lose\_sprite.setPosition(700, 325);

playagain\_sprite.setPosition(770, 540);

if (IntRect(770, 540, 384, 123).contains(Mouse::getPosition(window))) { MenuNum = 1; }

if (Mouse::isButtonPressed(Mouse::Left)) {

if (MenuNum == 1) {

svit = 0;

g = 0;

Sleep(300);

}

}

window.clear(Color::White);

window.draw(lose\_sprite);

window.draw(playagain\_sprite);

window.display();

}

}

return 0;

}

/\*

int turn(int team) {

int condition = 0;

int oponent=0;

switch (team) {

case 0:

if ((pole[0][0] == 'o') && (pole[0][1] == 'o') && (pole[0][2] == 'o')) {

condition = 1;

}

if ((pole[1][0] == 'o') && (pole[1][1] == 'o') && (pole[1][2] == 'o')) {

condition = 1;

}

if ((pole[2][0] == 'o') && (pole[2][1] == 'o') && (pole[2][2] == 'o')) {

condition = 1;

}

if ((pole[0][0] == 'o') && (pole[1][0] == 'o') && (pole[2][0] == 'o')) {

condition = 1;

}

if ((pole[0][1] == 'o') && (pole[1][1] == 'o') && (pole[2][1] == 'o')) {

condition = 1;

}

if ((pole[0][2] == 'o') && (pole[1][2] == 'o') && (pole[2][2] == 'o')) {

condition = 1;

}

if ((pole[0][0] == 'o') && (pole[1][1] == 'o') && (pole[2][2] == 'o')) {

condition = 1;

}

if ((pole[2][0] == 'o') && (pole[1][1] == 'o') && (pole[0][2] == 'o')) {

condition = 1;

}

if (condition == 1) {

return 1;

}

oponent = oponentturn(team);

if (oponent == 2) {

return 2;

}

return 0;

break;

case 1:

if ((pole[0][0] == 'x') && (pole[0][1] == 'x') && (pole[0][2] == 'x')) {

condition = 1;

}

if ((pole[1][0] == 'x') && (pole[1][1] == 'x') && (pole[1][2] == 'x')) {

condition = 1;

}

if ((pole[2][0] == 'x') && (pole[2][1] == 'x') && (pole[2][2] == 'x')) {

condition = 1;

}

if ((pole[0][0] == 'x') && (pole[1][0] == 'x') && (pole[2][0] == 'x')) {

condition = 1;

}

if ((pole[0][1] == 'x') && (pole[1][1] == 'x') && (pole[2][1] == 'x')) {

condition = 1;

}

if ((pole[0][2] == 'x') && (pole[1][2] == 'x') && (pole[2][2] == 'x')) {

condition = 1;

}

if ((pole[0][0] == 'x') && (pole[1][1] == 'x') && (pole[2][2] == 'x')) {

condition = 1;

}

if ((pole[2][0] == 'x') && (pole[1][1] == 'x') && (pole[0][2] == 'x')) {

condition = 1;

}

if (condition == 1) {

return 1;

}

oponent = oponentturn(team);

if (oponent == 2) {

return 2;

}

return 0;

break;

}

}

\*/

int oponentturn(int team) {

switch (team) {

case 0:

//проверка на победу

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

//горизонталь

if ((pole[0][i] == 'x') && (pole[1][i] == 'x') && (pole[2][i] == ' ')) { pole[2][i] = 'x'; return 2; }

if ((pole[0][i] == 'x') && (pole[2][i] == 'x') && (pole[1][i] == ' ')) { pole[1][i] = 'x'; return 2; }

if ((pole[1][i] == 'x') && (pole[2][i] == 'x') && (pole[0][i] == ' ')) { pole[0][i] = 'x'; return 2; }

//вертикаль

if ((pole[i][0] == 'x') && (pole[i][1] == 'x') && (pole[i][2] == ' ')) { pole[i][2] = 'x'; return 2; }

if ((pole[i][0] == 'x') && (pole[i][2] == 'x') && (pole[i][1] == ' ')) { pole[i][1] = 'x'; return 2; }

if ((pole[i][1] == 'x') && (pole[i][2] == 'x') && (pole[i][0] == ' ')) { pole[i][0] = 'x'; return 2; }

}

//диагональ1

if ((pole[0][0] == 'x') && (pole[1][1] == 'x') && (pole[2][2] == ' ')) { pole[2][2] = 'x'; return 2; }

if ((pole[0][0] == 'x') && (pole[2][2] == 'x') && (pole[1][1] == ' ')) { pole[1][1] = 'x'; return 2; }

if ((pole[1][1] == 'x') && (pole[2][2] == 'x') && (pole[0][0] == ' ')) { pole[0][0] = 'x'; return 2; }

//диагональ2

if ((pole[2][0] == 'x') && (pole[1][1] == 'x') && (pole[0][2] == ' ')) { pole[0][2] = 'x'; return 2; }

if ((pole[2][0] == 'x') && (pole[0][2] == 'x') && (pole[1][1] == ' ')) { pole[1][1] = 'x'; return 2; }

if ((pole[0][2] == 'x') && (pole[1][1] == 'x') && (pole[2][0] == ' ')) { pole[2][0] = 'x'; return 2; }

//проверка на угрозу

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

//горизонталь

if ((pole[0][i] == 'o') && (pole[1][i] == 'o') && (pole[2][i] == ' ')) { pole[2][i] = 'x'; return 1; }

if ((pole[0][i] == 'o') && (pole[2][i] == 'o') && (pole[1][i] == ' ')) { pole[1][i] = 'x'; return 1; }

if ((pole[1][i] == 'o') && (pole[2][i] == 'o') && (pole[0][i] == ' ')) { pole[0][i] = 'x'; return 1; }

//вертикаль

if ((pole[i][0] == 'o') && (pole[i][1] == 'o') && (pole[i][2] == ' ')) { pole[i][2] = 'x'; return 1; }

if ((pole[i][0] == 'o') && (pole[i][2] == 'o') && (pole[i][1] == ' ')) { pole[i][1] = 'x'; return 1; }

if ((pole[i][1] == 'o') && (pole[i][2] == 'o') && (pole[i][0] == ' ')) { pole[i][0] = 'x'; return 1; }

}

//диагональ1

if ((pole[0][0] == 'o') && (pole[1][1] == 'o') && (pole[2][2] == ' ')) { pole[2][2] = 'x'; return 1; }

if ((pole[0][0] == 'o') && (pole[2][2] == 'o') && (pole[1][1] == ' ')) { pole[1][1] = 'x'; return 1; }

if ((pole[1][1] == 'o') && (pole[2][2] == 'o') && (pole[0][0] == ' ')) { pole[0][0] = 'x'; return 1; }

//диагональ2

if ((pole[2][0] == 'o') && (pole[1][1] == 'o') && (pole[0][2] == ' ')) { pole[0][2] = 'x'; return 1; }

if ((pole[2][0] == 'o') && (pole[0][2] == 'o') && (pole[1][1] == ' ')) { pole[1][1] = 'x'; return 1; }

if ((pole[0][2] == 'o') && (pole[1][1] == 'o') && (pole[2][0] == ' ')) { pole[2][0] = 'x'; return 1; }

if (chod == 0) {

if ((pole[1][2] == 'o') && (pole[0][0] == ' ')) {

pole[0][0] = 'x';

}

else

if ((pole[2][1] == 'o') && (pole[2][0] == ' ')) {

pole[2][0] = 'x';

}

else

if ((pole[2][0] == 'o') && (pole[1][2] == 'o') && (pole[2][2] == ' ')) {

pole[2][2] = 'x';

}

else

if ((pole[0][0] == 'o') && (pole[1][2] == 'o') && (pole[0][2] == ' ')) {

pole[0][2] = 'x';

}

else

if ((pole[1][0] == 'o') && (pole[0][2] == 'o') && (pole[2][1] == ' ')) {

pole[2][1] = 'x';

}

else

if ((pole[1][0] == 'o') && (pole[2][2] == 'o') && (pole[2][1] == ' ')) {

pole[2][1] = 'x';

}

else

if ((pole[1][0] == 'o') && (pole[0][2] == ' ')) {

pole[0][2] = 'x';

}

else

if ((pole[0][1] == 'o') && (pole[0][0] == ' ')) {

pole[0][0] = 'x';

}

else

f ((pole[2][2] == 'o') && (pole[1][0] == ' ')) {

pole[1][0] = 'x';

}

else

if ((pole[1][0] == 'o') && (pole[0][1] == ' ')) {

pole[0][1] = 'x';

}

else

if ((pole[0][2] == 'o') && (pole[1][0] == ' ')) {

pole[1][0] = 'x';

} else

if ((pole[2][1] == 'o') && (pole[2][2]) == ' ') pole[2[2] = 'x'; } else

if ((pole[0][0] == 'o') && (pole[1][2]) == ' ') {

pole[1][2] = 'x'; }

else

if ((pole[2][0] == 'o') && (pole[1][2]) == ' ') {

pole[1][2] = 'x';

}

return 1;

}

else {

if (chod == 1) {

if ((pole[2][2] == 'o') && (pole[2][0] == ' ')) {

pole[2][0] = 'x';

}

else

if ((pole[0][1] == 'o') && (pole[2][1] == ' ')) {

pole[2][1] = 'x';

}

else

if ((pole[0][2] == 'o') && (pole[0][0] == ' ')) {

pole[0][0] = 'x';

}

else

if ((pole[2][1] == 'o') && (pole[0][1] == ' ')) {

pole[0][1] = 'x';

}

else

if ((pole[0][1] == 'o') && (pole[1][0] == ' ')) {

pole[1][0] = 'x';

}

else

if ((pole[1][2] == 'o') && (pole[1][0] == ' ')) {

pole[0][0] = 'x';

}

else

if ((pole[1][0] == 'o') && (pole[1][2]) == ' ') {

pole[1][2] = 'x';

}

else

if ((pole[2][1] == 'o') && (pole[2][0] == ' ')) {

pole[2][0] = 'x';

}

else

if ((pole[0][1] == 'o') && (pole[0][0] == ' ')) {

pole[0][0] = 'x';

}

else

if ((pole[2][1] == 'o') && (pole[2][2]) == ' ') {

pole[2][2] = 'x';

}

chod++;

return 1;

}

else {

if (chod == 2) {

if ((pole[0][0] == 'o') && (pole[1][2] == ' ')) {

pole[1][2] = 'x';

}

chod++;

return 1;

}

}

}

break;

case 1:

//проверка на победу

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

//горизонталь

if ((pole[0][i] == 'o') && (pole[1][i] == 'o') && (pole[2][i] == ' ')) { pole[2][i] = 'o'; return 2; }

if ((pole[0][i] == 'o') && (pole[2][i] == 'o') && (pole[1][i] == ' ')) { pole[1][i] = 'o'; return 2; }

if ((pole[1][i] == 'o') && (pole[2][i] == 'o') && (pole[0][i] == ' ')) { pole[0][i] = 'o'; return 2; }

//вертикаль

if ((pole[i][0] == 'o') && (pole[i][1] == 'o') && (pole[i][2] == ' ')) { pole[i][2] = 'o'; return 2; }

if ((pole[i][0] == 'o') && (pole[i][2] == 'o') && (pole[i][1] == ' ')) { pole[i][1] = 'o'; return 2; }

if ((pole[i][1] == 'o') && (pole[i][2] == 'o') && (pole[i][0] == ' ')) { pole[i][0] = 'o'; return 2; }

}

//диагональ1

if ((pole[0][0] == 'o') && (pole[1][1] == 'o') && (pole[2][2] == ' ')) { pole[2][2] = 'o'; return 2; }

if ((pole[0][0] == 'o') && (pole[2][2] == 'o') && (pole[1][1] == ' ')) { pole[1][1] = 'o'; return 2; }

if ((pole[1][1] == 'o') && (pole[2][2] == 'o') && (pole[0][0] == ' ')) { pole[0][0] = 'o'; return 2; }

//диагональ2

if ((pole[2][0] == 'o') && (pole[1][1] == 'o') && (pole[0][2] == ' ')) { pole[0][2] = 'o'; return 2; }

if ((pole[2][0] == 'o') && (pole[0][2] == 'o') && (pole[1][1] == ' ')) { pole[1][1] = 'o'; return 2; }

if ((pole[0][2] == 'o') && (pole[1][1] == 'o') && (pole[2][0] == ' ')) { pole[2][0] = 'o'; return 2; }

//проверка на угрозу

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

//горизонталь

if ((pole[0][i] == 'x') && (pole[1][i] == 'x') && (pole[2][i] == ' ')) { pole[2][i] = 'o'; return 1; }

if ((pole[0][i] == 'x') && (pole[2][i] == 'x') && (pole[1][i] == ' ')) { pole[1][i] = 'o'; return 1; }

if ((pole[1][i] == 'x') && (pole[2][i] == 'x') && (pole[0][i] == ' ')) { pole[0][i] = 'o'; return 1; }

//вертикаль

if ((pole[i][0] == 'x') && (pole[i][1] == 'x') && (pole[i][2] == ' ')) { pole[i][2] = 'o'; return 1; }

if ((pole[i][0] == 'x') && (pole[i][2] == 'x') && (pole[i][1] == ' ')) { pole[i][1] = 'o'; return 1; }

if ((pole[i][1] == 'x') && (pole[i][2] == 'x') && (pole[i][0] == ' ')) { pole[i][0] = 'o'; return 1; }

}

//диагональ1

if ((pole[0][0] == 'x') && (pole[1][1] == 'x') && (pole[2][2] == ' ')) { pole[2][2] = 'o'; return 1; }

if ((pole[0][0] == 'x') && (pole[2][2] == 'x') && (pole[1][1] == ' ')) { pole[1][1] = 'o'; return 1; }

if ((pole[1][1] == 'x') && (pole[2][2] == 'x') && (pole[0][0] == ' ')) { pole[0][0] = 'o'; return 1; }

//диагональ2

if ((pole[2][0] == 'x') && (pole[1][1] == 'x') && (pole[0][2] == ' ')) { pole[0][2] = 'o'; return 1; }

if ((pole[2][0] == 'x') && (pole[0][2] == 'x') && (pole[1][1] == ' ')) { pole[1][1] = 'o'; return 1; }

if ((pole[0][2] == 'x') && (pole[1][1] == 'x') && (pole[2][0] == ' ')) { pole[2][0] = 'o'; return 1; }

if (chod == 0) {

if (pole[0][0] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[1][0] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[2][0] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[0][1] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[2][1] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[0][2] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[1][2] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[2][2] == 'x') {

pole[1][1] = 'o';

}

else

if (pole[1][1] == 'x') {

pole[2][2] = 'o';

chod = chod + 7;

}

chod++;

return 1;

}

else {

if (chod == 1) {

if ((pole[2][1] == 'x') && (pole[2][0] == ' ')) {

pole[2][0] = 'o';

}

else

if ((pole[2][0] == 'x') && (pole[1][2] == 'x') && (pole[2][2] == ' ')) {

pole[2][2] = 'o';

}

else

if ((pole[0][0] == 'x') && (pole[1][2] == 'x') && (pole[0][2] == ' ')) {

pole[0][2] = 'o';

}

else

if ((pole[1][0] == 'x') && (pole[0][2] == 'x') && (pole[2][1] == ' ')) {

pole[2][1] = 'o';

}

else

if ((pole[1][0] == 'x') && (pole[2][2] == 'x') && (pole[2][1] == ' ')) {

pole[2][1] = 'o';

}

else

if ((pole[1][0] == 'x') && (pole[0][2] == ' ')) {

pole[0][2] = 'o';

}

else

if ((pole[0][1] == 'x') && (pole[0][2] == ' ')) {

pole[0][2] = 'o';

}

else

if ((pole[2][2] == 'x') && (pole[1][0] == ' ')) {

pole[1][0] = 'o';

}

else

if ((pole[1][0] == 'x') && (pole[0][1] == ' ')) {

pole[0][1] = 'o';

}

else

if ((pole[0][2] == 'x') && (pole[1][0] == ' ')) {

pole[1][0] = 'o';

}

else

if ((pole[2][1] == 'x') && (pole[2][2]) == ' ') {

pole[2][2] = 'o';

}

chod++;

return 1;

}

else {

if (chod == 2) {

if ((pole[0][1] == 'x') && (pole[2][1] == ' ')) {

pole[2][1] = 'o';

}

else

if ((pole[2][1] == 'x') && (pole[0][1] == ' ')) {

pole[0][1] = 'o';

}

else

if ((pole[0][1] == 'x') && (pole[1][0] == ' ')) {

pole[1][0] = 'o';

}

else

if ((pole[1][2] == 'x') && (pole[1][0] == ' ')) {

pole[0][0] = 'o';

}

else

if ((pole[1][0] == 'x') && (pole[1][2]) == ' ') {

pole[1][2] = 'o';

}

chod++;

return 1;

}

else {

if (chod == 3) {

if ((pole[0][0] == 'x') && (pole[1][2] == ' ')) {pole[1][2] = 'o';}

chod++;

return 1;

}

else {

if (chod == 8) {

if ((pole[0][0] == 'x') && (pole[0][2] == 'o')) {

pole[0][1] = 'o';

}

else

if ((pole[0][0] == 'x') && (pole[0][2] == 'x')) { pole[1][0] = 'o';

}

else

if (pole[0][0] == 'x') {pole[0][2] = 'o';}

chod++;

return 1;

}

else {

if (chod == 9) {

if (pole[2][0] == 'x') {pole[2][1] = 'o';}

return 1;

}

}

}

}

}

}

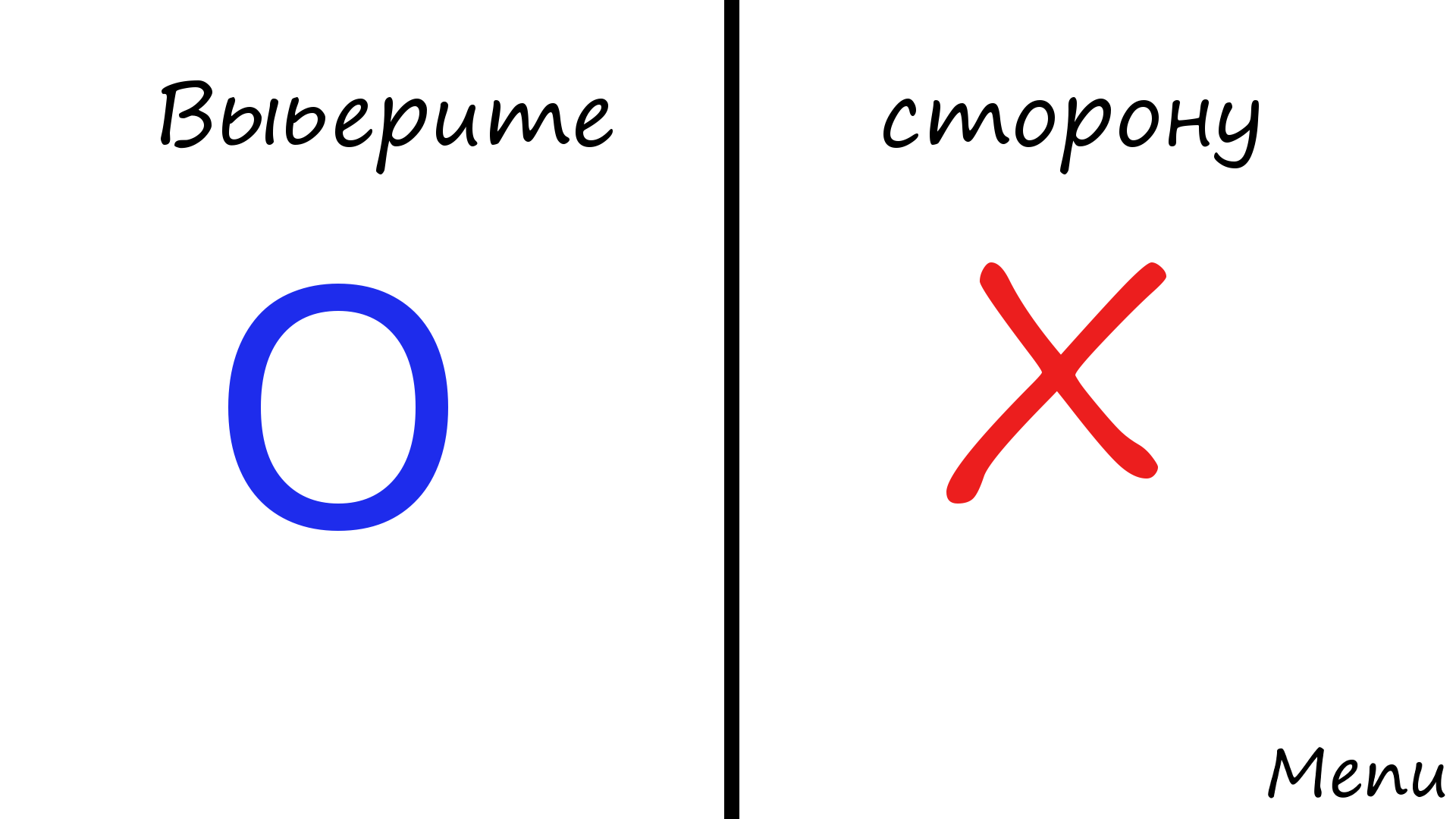
return 1;

break;

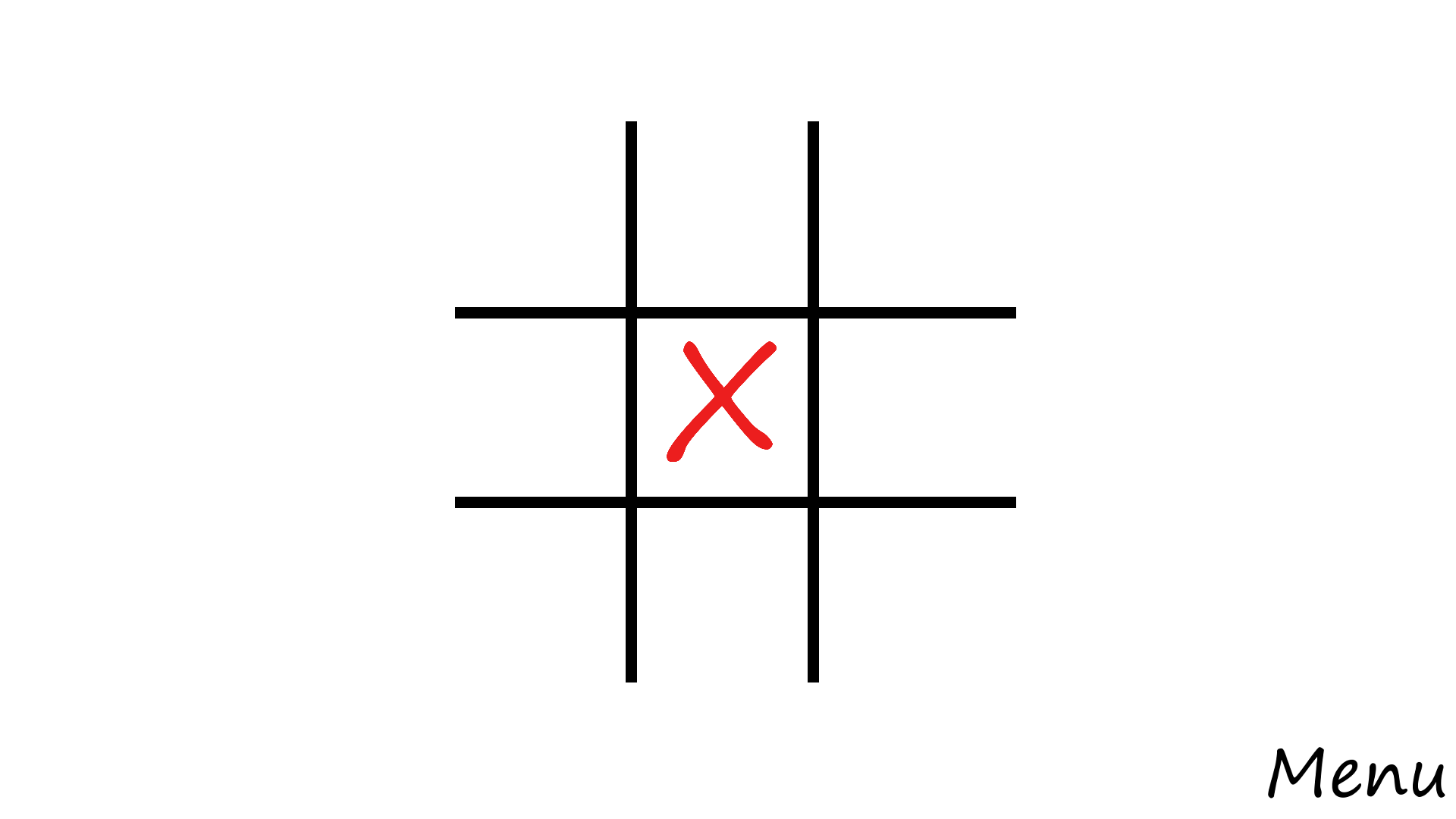
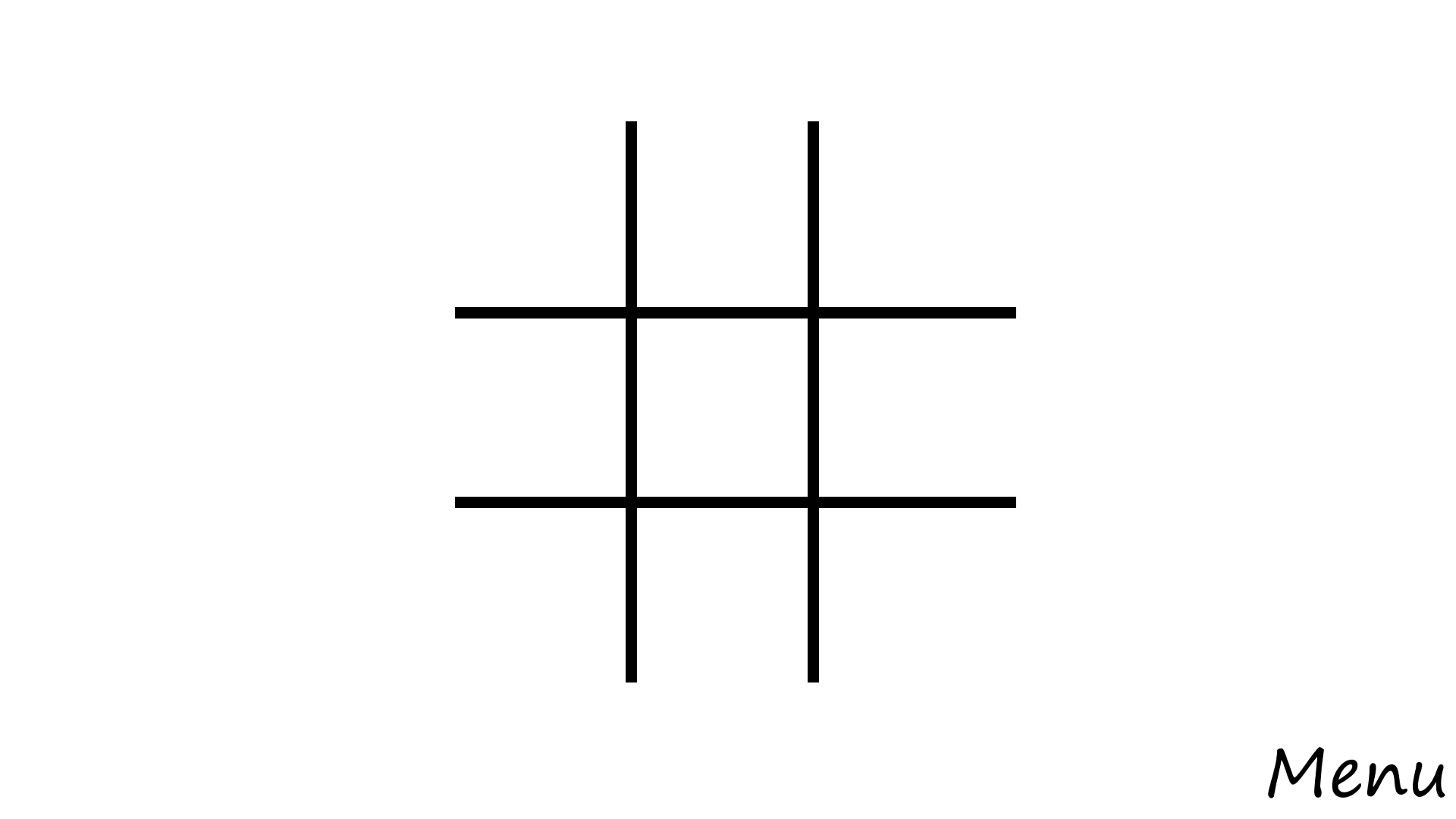
}

}**4. Результат работы программы**

1.Меню 2. Start

3. O (ходит вторым) 4. X (ходит первым)

1. Lose 6. Draw