#include "time.h"

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <list>

using namespace std;

static const int INF = 100;

static enum { playing, owin, xwin, draw } state;

typedef struct {

char symbol;

int move;

bool selected;

bool win;

}

player;

void display\_board(); // отображение

void get\_move(); // поиск(получение) хода

void update\_game(); // следующий шаг игры

bool free\_square(); // проверка пустоты ячейки

void update\_board(); // обновление позиций на доске

void verify\_move(); // проверка валидности хода(0-8 и не занята ячейка)

void generate\_moves(char \_board[9], list<int> &move\_list); // генерация всех ходов для minmax

void check\_game\_state(char board[9]); // проверка постояния игры

int evaluate\_position(char \_board[9], player \_player); // оценка положения

int MiniMax(char \_board[9], player \_player);

int MinMove(char \_board[9], player \_player);

int MaxMove(char \_board[9], player \_player);

static player player1, player2, cplayer;

static char board[9] = { 0 }; // обозначение доски для игры

static char symbol; // символ для текущего игрока

static int nmove; // последний шаг сделанный действующим игроком

int q = 0;

//-------------------------------------------------------------

int main() {

setlocale(LC\_ALL, "rus");

srand((unsigned)time(NULL));

// Get player sumbol

int selection = rand() % 2;

if (selection == 0) {

player2.symbol = 'x';

player1.symbol = 'o';

}

else if (selection == 1) {

player1.symbol = 'x';

player2.symbol = 'o';

}

state = playing;

// Play

while (state == playing) {

get\_move();

update\_game();

}

// если игра закончилась

if (state == xwin || state == owin || state == draw) {

if (state == xwin) player1.win = 1;

else if (state == owin) player2.win = 1;

if (player1.win) cout << "Игрок одержал победу!" << endl;

else if (player2.win) cout << "Компьютер одежал победу!" << endl;

else if (player1.win == 0 && player2.win == 0) cout << "Ничья!." << endl;

}

system("PAUSE");

return 0;

}

//-------------------------------------------------------------

void get\_move() {

player1.move = -1;

player2.move = -1;

cin.sync(); //очищение буфера стандартного ввода

if (player1.selected) {

cout << endl;

cout << " Пожалуйста выберите номер поля от (1-9): "<<endl;

cin >> player1.move;

nmove = player1.move;

symbol = player1.symbol;

cplayer = player1;

player1.selected = 0;

player2.selected = 1;

}

else if (player2.selected) {

player2.move = MiniMax(board, player2);

nmove = player2.move;

symbol = player2.symbol;

cplayer = player2;

player1.selected = 1;

player2.selected = 0;

state = playing;

}

verify\_move();

if ((state == xwin || state == owin || state == draw)) return;

}

//-------------------------------------------------------------

void update\_board() {

if (state == playing) {

if (player1.move != -1 && player2.move == -1) board[player1.move - 1] = player1.symbol;

else if (player2.move != -1) board[player2.move - 1] = player2.symbol;

}

}

//-------------------------------------------------------------

void update\_game() {

update\_board();

display\_board();

check\_game\_state(board);

}

//-------------------------------------------------------------

bool free\_square() {

if (player1.move != -1 && player2.move == -1)

return board[player1.move - 1] == 0;

else if (player2.move != -1)

return board[player2.move - 1] == 0;

return 0;

}

//-------------------------------------------------------------

void display\_board() {

cout << endl;

cout << " " << board[0] << " | " << board[1] << " | " << board[2] << endl;

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

cout << " " << board[3] << " | " << board[4] << " | " << board[5] <<endl;

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

cout << " " << board[6] << " | " << board[7] << " | " << board[8] << endl;

cout << endl;

}

//-------------------------------------------------------------

void verify\_move() {

if (!(nmove > 0 && nmove < 10) || !free\_square()) {

cout << "Неверный ход. Повторите ход." << endl;

cout << "" << endl;

cout << "" << endl;

cout << "Пример" << endl;

cout << "1 | " << "2 | " << "3 | " << endl;

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

cout << "4 | " << "5 | " << "6 | " << endl;

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

cout << "7 | " << "8 | " << "9 | " << endl;

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

player1.selected = 1;

player2.selected = 0;

get\_move();

}

}

//-------------------------------------------------------------

void check\_game\_state(char board[9]) {

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

(board[3] == symbol && board[4] == symbol && board[5] == symbol) ||

(board[6] == symbol && board[7] == symbol && board[8] == symbol) ||

(board[0] == symbol && board[3] == symbol && board[6] == symbol) ||

(board[1] == symbol && board[4] == symbol && board[7] == symbol) ||

(board[2] == symbol && board[5] == symbol && board[8] == symbol) ||

(board[0] == symbol && board[4] == symbol && board[8] == symbol) ||

(board[2] == symbol && board[4] == symbol && board[6] == symbol)) {

if (symbol == 'x') state = xwin;

else if (symbol == 'o') state = owin;

}

else {

state = draw;

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

if (board[i] == 0) {

state = playing;

break;

}

}

}

}

//-------------------------------------------------------------

void generate\_moves(char \_board[9], list<int> &move\_list) {

for (int i = 0; i < 9; i++) if (\_board[i] == 0) move\_list.push\_back(i);

}

//-------------------------------------------------------------

int evaluate\_position(char \_board[9], player \_player) {

check\_game\_state(\_board);

if ((state == xwin || state == owin || state == draw)) {

if ((state == xwin && \_player.symbol == 'x') || (state == owin && \_player.symbol == 'o')) return +INF;

else if ((state == xwin && \_player.symbol == 'o') || (state == owin && \_player.symbol == 'x')) return -INF;

else if (state == draw) return 0;

}

return -1;

}

//-------------------------------------------------------------

int MiniMax(char \_board[9], player \_player) {

int best\_val = -INF, index = 0;

list<int> move\_list;

char best\_moves[9] = { 0 };

generate\_moves(\_board, move\_list); // генерить все ходы

while (!move\_list.empty()) { // по всем

\_board[move\_list.front()] = \_player.symbol; // заполнение ячейки символом игрока

symbol = \_player.symbol;

int val = MinMove(\_board, \_player); // поиск MinMove

if (val > best\_val) { // выбор наибольшего

best\_val = val;

index = 0;

best\_moves[index] = 1 + move\_list.front();

}

else if (val == best\_val)

best\_moves[++index] = 1 + move\_list.front(); // если несколько наибольших - тоже заносим

printf("\nminimax(Отслеживание): %3d(%1d) ", 1 + move\_list.front(), val);

\_board[move\_list.front()] = 0; // отбрасываем ход, идем далее по циклу

move\_list.pop\_front();

}

if (index > 0) index = rand() % index; // выбираем случайный из нескольких ходов (random)

printf("\nminimax best: %3d(%1d) ", best\_moves[index], best\_val);

printf("Шаги подсчитаны: %d", q);

q = 0;

return best\_moves[index];

}

//---------------------------------------------------------------------------------

int MinMove(char \_board[9], player \_player) {

int pos\_value = evaluate\_position(\_board, \_player); // проверить состояние игры для текущего игрока

if (pos\_value != -1) return pos\_value;

q++;

int best\_val = +INF;

list<int> move\_list;

generate\_moves(\_board, move\_list); // генерить все ходы

while (!move\_list.empty()) { // по всем

\_player.symbol == 'x' ? symbol = 'o' : symbol = 'x'; // расчет для противника

\_board[move\_list.front()] = symbol; // front - ссылка на первый элемент, заполнение ячейки символом

int val = MaxMove(\_board, \_player); // подсчет MaxMove

if (val < best\_val) {

best\_val = val; // выбрать с наименьшим

//printf("%3d(%d) ",move\_list.front()+1,best\_val);

}

\_board[move\_list.front()] = 0;

move\_list.pop\_front();

}

return best\_val;

}

//---------------------------------------------------------------------------------

int MaxMove(char \_board[9], player \_player) {

int pos\_value = evaluate\_position(\_board, \_player);

if (pos\_value != -1) return pos\_value;

q++;

int best\_val = -INF;

list<int> move\_list;

generate\_moves(\_board, move\_list);

while (!move\_list.empty()) {

\_player.symbol == 'x' ? symbol = 'x' : symbol = 'o'; // расчет для себя

\_board[move\_list.front()] = symbol;

int val = MinMove(\_board, \_player);

if (val > best\_val) {

best\_val = val;

//printf("%3d(%d) ",move\_list.front()+1,best\_val);

}

\_board[move\_list.front()] = 0;

move\_list.pop\_front();

}

return best\_val;

system("pause");

}

