**The laboratory work 10**

**Minimax function for Tic Tac Toe game**

|  |
| --- |
| #pragma once#include <iostream>#include <algorithm>#include <tuple>#include <iterator>#include <vector>#include "Search.h"using namespace std;int winner(int board[9]){ int i, j; int threes[8][3] = { { 1,2,3 },{ 4,5,6 },{ 7,8,9 },{ 1,4,7 },{ 2,5,8 },{ 3,6,9 },{ 1,5,9 },{ 3,5,7 } }; int total = 0; for (i = 0; i < 8; i++) { total = 0; for (j = 0; j < 3; j++) { total += board[threes[i][j] - 1]; } if (total == -3) return -1; else if (total == 3) return 1; } return 0;}bool movesLeft(int board[9]){ int i; for (i = 0; i < 9; i++) { if (board[i] == 0) return true; } return false;}tuple<int,int> minimax(int board[9], int realvalue, int mysymbol, int depth, int baseValue) { vector<int> scores; vector<int> moves; int optimal\_score = 0; int GameWinner = winner(board); if (GameWinner == mysymbol) return{ 10, -1000 }; else if (GameWinner == -mysymbol) return{ -10, -1000 }; else if (movesLeft(board) == false) return{ 0, -1000 }; for (int i = 0; i < 9; i++) { if (board[i] == 0) { int newboard[9]; for (int j = 0; j < 9; j++) { newboard[j] = board[j]; } newboard[i] = realvalue; int score, move; tie(score, move) = minimax(newboard, -realvalue, mysymbol, depth + 1, baseValue); if (depth == 0) { baseValue = max(baseValue, score); } else if (score < baseValue) { return{ score, -1000 }; } scores.push\_back(score); moves.push\_back(i); } } int max = scores[0]; int min = scores[0]; for (int i = 1; i < scores.size(); i++) { if (scores[i] > max) max = scores[i]; } for (int i = 1; i < scores.size(); i++) { if (scores[i] < min) min = scores[i]; } if (realvalue == mysymbol) optimal\_score = find(scores.begin(), scores.end(), max) - scores.begin(); else optimal\_score = find(scores.begin(), scores.end(), min) - scores.begin(); return {scores[optimal\_score], moves[optimal\_score]};}int mymove(int board[9], char mysymbol){ int optimal\_score; int optimal\_move; int s\_symbol; int baseValue = -1000; cout << "Board as seen by the machine " << endl; cout << "[ "; for (int i = 0; i < 9; i++) cout << board[i] << " "; cout << " ]" << endl; cout << "The machine is playing: "; cout << mysymbol << endl; if (mysymbol == 'X') { s\_symbol = 1; } if (mysymbol == 'O') { s\_symbol = -1; } tie(optimal\_score, optimal\_move) = minimax(board, s\_symbol, s\_symbol, 0, baseValue); return optimal\_move;} |