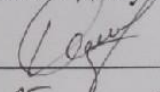


МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

Работа допущена к защите

 Руководитель
« 25 » 02 2019 г.

КОНТРОЛЬНАЯ РАБОТА

по дисциплине «Проектирование пользовательского интерфейса»

на тему: «Реализация игры «Крестики-нолики» с помощью алгоритма
«Minimax»»

Студент  Шорин В.Д.

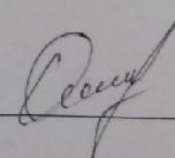
Шифр 171406

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Направленность (профиль) Промышленная разработка программного
обеспечения

Группа 71-ПГ

Преподаватель  Олькина Е.В.

Оценка: « зачтено » Дата 25.02.19

Орел 2019

Задание на контрольную работу

Реализовать игру «Крестики-нолики» против компьютера, действующего по алгоритму «Minimax». Использовать графический интерфейс. Все действия осуществляются мышью.

Листинг программы

```
#include "windows.h"#include <stdio>#include <algorithm> #include <limits>
struct Move{int x;int y;};
BOOL RegClass(WNDPROC, LPCTSTR, UINT);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
HINSTANCE hInstance;
char szMainClass[] = "MainClass";
char szMainTitle[] = "Cross-Zero";
void DrawLine(HDC, int, int, int, int);
void DrawGameField(HDC);
void DrawCross(HDC, int, int);
void ComputerTurn(HDC);
BOOL isSquareFree(int, int);
BOOL isTie();
BOOL isWinner(char);
Move Minimax();
int MaxSearch();
int MinSearch();
HWND hwnd;
char gameField[3][3] = {' ',' ',' ',' ',' ',' ',' ',' ',' '};
char symbol;
int xMouse, yMouse;
bool isComputerTurn = false;
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow){
    MSG msg;
    hInstance = hInst;
    if (!RegClass(WndProc, szMainClass, COLOR_WINDOW)) { return
FALSE; }
    const int w = 308, h = 330;
    int x = (GetSystemMetrics(SM_CXSCREEN) - w) / 2;
    int y = (GetSystemMetrics(SM_CYSCREEN) - h) / 2;
    hwnd = CreateWindow(szMainClass, szMainTitle,
        WS_VISIBLE | WS_SYSMENU | WS_MINIMIZEBOX,
        x, y, w, h, 0, 0, hInstance, NULL);
```

```

    if (!hwnd) { return FALSE; }
    while (GetMessage(&msg, 0, 0, 0)) {
        if (isComputerTurn) {
            HDC hdc = GetDC(hwnd);
            ComputerTurn(hdc);
        }
        if (isWinner('X')) {
            MessageBox(hwnd, "Player wins", "Winner", MB_OK);
            SendMessage(hwnd, WM_CLOSE, 0, 0);
        }
        if (isWinner('O')) {
            MessageBox(hwnd, "Computer wins", "Winner", MB_OK);
            SendMessage(hwnd, WM_CLOSE, 0, 0);
        }
        if (isTie()) {
            MessageBox(hwnd, "Its tie", "Tie", MB_OK);
            SendMessage(hwnd, WM_CLOSE, 0, 0);
        }
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}
}6
BOOL RegClass(WNDPROC Proc, LPCTSTR szName, UINT brBackground){
    WNDCLASS wc;
    wc.style = CS_HREDRAW | CS_VREDRAW;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.lpfWndProc = Proc;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH)(brBackground + 1);
    wc.lpszMenuName = (LPCTSTR)NULL;
    wc.lpszClassName = szName;
    return (RegisterClass(&wc) != 0);
}
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM
wParam, LPARAM lParam){
    HDC hdc; PAINTSTRUCT ps;
    switch (msg) {
        case WM_LBUTTONDOWN: {
            xMouse = LOWORD(lParam);
            yMouse = HIWORD(lParam);

```

```

        hdc = GetDC(hWnd);
        if (isSquareFree(xMouse, yMouse))
        {
            DrawCross(hdc, xMouse / 100, yMouse / 100);
            symbol = 'X';
            gameField[xMouse / 100][yMouse / 100] = symbol;
        } else {
            char str[64];
            sprintf(str, "There is already cross/zero \nPlease, try
again", xMouse / 100, yMouse / 100);
            MessageBox(hWnd, (LPCTSTR)str, "XY", MB_OK);
        }
        isComputerTurn = true;
        break;
    }
    case WM_PAINT:
    {
        hdc = BeginPaint(hWnd, &ps);
        DrawGameField(hdc);           //Отрисовка поля
        EndPaint(hWnd, &ps);
        break;
    }
    case WM_DESTROY: {PostQuitMessage(0);return 0; }
}
return DefWindowProc(hWnd, msg, wParam, lParam);
}
//Отрисовка линии
void DrawLine(HDC hdc, int left, int top, int right, int bottom) {
    MoveToEx(hdc, left, top, NULL);LineTo(hdc, right, bottom);}

//Отрисовка игрового поля
void DrawGameField(HDC hdc) {
    DrawLine(hdc, 100, 0, 100, 300);DrawLine(hdc, 200, 0, 200, 300);
    DrawLine(hdc, 0, 100, 300, 100);DrawLine(hdc, 0, 200, 300, 200);
}
//Отрисовка креста
void DrawCross(HDC hdc, int left, int top){
    DrawLine(hdc, left * 100, top * 100, (left + 1) * 100, (top + 1) * 100);
    DrawLine(hdc, left * 100, (top + 1) * 100, (left + 1) * 100, top * 100);
}
//Ход компьютера
void ComputerTurn(HDC hdc){
    if (isTie()) {
        MessageBox(hWnd, "Its tie", "Tie", MB_OK);
        SendMessage(hWnd, WM_CLOSE, 0, 0);
    }
}

```

```

    Move computerMove = Minimax();
    Ellipse(hdc, computerMove.x * 100, computerMove.y * 100,
(computerMove.x + 1) * 100, (computerMove.y + 1) * 100);
    symbol = 'O';
    gameField[computerMove.x][computerMove.y] = symbol;
    isComputerTurn = false;
}
BOOL isSquareFree(int _xMouse, int _yMouse){
    if (gameField[_xMouse / 100][_yMouse / 100] != 'O'
        && gameField[_xMouse / 100][_yMouse / 100] != 'X')
    {
        return true;
    }
    else {
        return false;
    }
}
BOOL isTie(){
    for (int i = 0; i < 3; i++) {
        if (gameField[i][0] == ' ' || gameField[i][1] == ' ' || gameField[i][2] == ' ')
        {
            return false;
        }
    }
    return true;
}
BOOL isWinner(char player){
    for (int i = 0; i < 3; i++) {
        for (int i = 0; i < 3; i++) {
            if (gameField[i][0] == player && gameField[i][1] == player
&& gameField[i][2] == player)
            {
                return true;
            }

            if (gameField[0][i] == player && gameField[1][i] == player
&& gameField[2][i] == player)
            {
                return true;
            }

            if (gameField[0][0] == player && gameField[1][1] == player &&
gameField[2][2] == player)
            {
                return true;
            }
            if (gameField[0][2] == player && gameField[1][1] == player &&
gameField[2][0] == player)
            {
                return true;
            }
            return false;
        }
    }
}
Move Minimax(){
    int score = (std::numeric_limits<int>::max)();
    Move move;
    for (int i = 0; i < 3; i++) {

```

```

        for (int j = 0; j < 3; j++) {
            if (gameField[i][j] == ' ') {
                gameField[i][j] = 'O';
                int temp = MaxSearch();
                if (temp < score) {
                    score = temp;
                    move.x = i;
                    move.y = j;
                }
                gameField[i][j] = ' ';
            }
        }
    }
    return move;
}

int MaxSearch(){
    if (isWinner('X')) { return 10; }
    else if (isWinner('O')) { return -10; }
    else if (isTie()) { return 0; }
    int score = (std::numeric_limits<int>::min)();
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (gameField[i][j] == ' ') {
                gameField[i][j] = 'X';
                score = (std::max)(score, MinSearch());
                gameField[i][j] = ' ';
            }
        }
    }
    return score;
}

int MinSearch(){
    if (isWinner('X')) { return 10; }
    else if (isWinner('O')) { return -10; }
    else if (isTie()) { return 0; }
    int score = (std::numeric_limits<int>::max)();
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (gameField[i][j] == ' ') {
                gameField[i][j] = 'O';
                score = (std::min)(score, MaxSearch());
                gameField[i][j] = ' ';
            }
        }
    }
}

```

```
    }  
    return score;  
}
```