МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное автономное образовательное учреждение высшего образования "КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО" ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ Кафедра компьютерной инженерии и моделирования  
  
​

**Отчёт по лабораторной работе № 4  
по дисциплине "Программирование"**

студента 2 курса группы ПИ-б-о-191(1)  
Спичека Игоря Дмитриевича  
направления подготовки 09.03.04 "Программная инженерия"  
​

|  |  |  |
| --- | --- | --- |
| Научный руководитель старший преподаватель кафедры компьютерной инженерии и моделирования | (оценка) | Чабанов В.В. |

​Симферополь, 2020

**Цель:**

1. Закрепить навыки работы с перечислениями;
2. Закрепить навыки работы с структурами;
3. Освоить методы составления многофайловых программ.

**Ход работы:**

1. Создал вспомогательный файл содержащий реализацию основных функций игры.
2. Создал заголовочный файл для связи главного и вспомогательного файлов.
3. Создал игру "Иксики-нолики" используя функции описанные в вспомогательном файле.
4. Загрузил весь проект свой репозиторий.

**КОД:**

XO.cpp (main):

#include "pch.h"

#include "XOutils.h"

#include <iostream>

int main()

{

setlocale(LC\_ALL, "Russian");

char c;

std::cout << "Введите ваш символ (X или O или любой другой символ): " << std::endl;

std::cin >> c;

XO::Game game = initGame(c);

updateDisplay(game);

while (!updateGame(&game)) {

updateDisplay(game);

}

return 0;

}

XOutils.h:

#pragma once

#ifndef XO\_H

#define XO\_H

namespace XO {

enum Status {

PLAY,

USER\_WIN,

BOT\_WIN,

NOT\_WIN

};

struct Game {

char board[3][3] = {};

bool isUserTurn;

char userChar;

char botChar;

int playerScore = 0;

int botScore = 0;

Status status;

};

}

XO::Game initGame(char);

void updateDisplay(const XO::Game);

bool updateGame(XO::Game\*);

#endif

XOutils.cpp:

#include "pch.h"

#include <iostream>

#include "XOutils.h"

#include <ctime>

#include <algorithm>

XO::Game initGame(char userChar) {

//system("CLS");

srand(unsigned(time(0)));

XO::Game gm;

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

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

gm.board[i][j] = '.';

}

}

gm.isUserTurn = rand() % 2;

gm.userChar = userChar;

gm.botChar = (userChar=='O'?'X':'O');

gm.status = XO::PLAY;

return gm;

}

void updateDisplay(const XO::Game game) {

system("CLS");

std::cout << game.playerScore << " - " << game.botScore << std::endl;

std::cout << "-------" << std::endl;

std::cout << " 1 2 3" << std::endl;

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

std::cout << i + 1 << ' ';

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

std::cout << game.board[i][j] << ' ';

}

std::cout << std::endl;

}

}

bool boardEmpty(XO::Game\* game) {

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

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

if (game->board[i][j] != '.') return false;

}

}

return true;

}

bool checkWinMoves(XO::Game\* game, char Char) {

int rows[3] = {};

int cols[3] = {};

int diags[2] = {};

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

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

if (game->board[i][j] == Char) rows[i]++;

}

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

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

if (game->board[j][i] == Char) cols[i]++;

}

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

if (game->board[i][i] == Char) diags[0]++;

}

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

if (game->board[2 - i][i] == Char) diags[1]++;

if (diags[0] == 2) {

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

if (game->board[i][i] == '.') {

game->board[i][i] = game->botChar;

return true;

}

}

}

if (diags[1] == 2) {

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

if (game->board[2 - i][i] == '.') {

game->board[2 - i][i] = game->botChar;

return true;

}

}

}

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

if (rows[i] == 2) {

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

if (game->board[i][j] == '.') {

game->board[i][j] = game->botChar;

return true;

}

}

}

}

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

if (cols[i] == 2) {

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

if (game->board[j][i] == '.') {

game->board[j][i] = game->botChar;

return true;

}

}

}

}

return false;

}

void botTurn(XO::Game\* game) {

if (boardEmpty(game)) {

game->board[1][1] = game->botChar;

return;

}

if (!checkWinMoves(game, game->botChar)) {

if (!checkWinMoves(game, game->userChar)){

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

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

if (game->board[i][j] == '.') {

game->board[i][j] = game->botChar;

return;

}

}

}

}

}

return;

}

void userTurn(XO::Game\* game) {

int row, col;

while (1) {

std::cin >> col >> row;

col--; row--;

if (game->board[row][col] == '.' && row >= 0 && row <= 2 && col >= 0 && col <= 2) {

game->board[row][col] = game->userChar;

break;

}

else {

std::cout << "Введите правильную координату точки!" << std::endl;

}

}

}

bool won(XO::Game\* game, char c) {

int a = 0;

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

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

if (game->board[i][j] == c) a++;

if (a == 3) return true;

}

a = 0;

}

a = 0;

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

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

if (game->board[j][i] == c) a++;

if (a == 3) return true;

}

a = 0;

}

a = 0;

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

if (game->board[i][i] == c) a++;

if (a == 3) return true;

}

a = 0;

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

if (game->board[2 - i][i] == c) a++;

if (a == 3) {

return true;

}

}

return false;

}

bool draw(XO::Game\* game) {

bool full = true;

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

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

if (game->board[i][j] == '.') full = false;

}

}

return (full && !won(game, game->userChar) && !won(game, game->botChar));

}

bool quit() {

char a = 0;

while (1) {

std::cout << "Еще раз? (1 - да, 2 - нет): ";

std::cin >> a;

switch (a) {

case '1': return false;

case '2': return true;

}

}

}

void restart(XO::Game\* game) {

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

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

game->board[i][j] = '.';

}

}

updateDisplay(\*game);

game->status = XO::PLAY;

}

bool updateGame(XO::Game\* game) {

if (won(game, game->userChar)) {

game->status = XO::USER\_WIN;

std::cout << "Вы выиграли!" << std::endl;

game->playerScore++;

if (quit()) return true;

restart(game);

}

if (won(game, game->botChar)) {

game->status = XO::BOT\_WIN;

std::cout << "Вы проиграли!" << std::endl;

game->botScore++;

if(quit()) return true;

restart(game);

}

if (draw(game)) {

game->status = XO::NOT\_WIN;

std::cout << "Ничья!" << std::endl;

if(quit()) return true;

restart(game);

}

if (game->isUserTurn) {

userTurn(game);

game->isUserTurn = false;

}

else {

botTurn(game);

game->isUserTurn = true;

}

return false;

}

**Вывод**

Закрепил навыки работы с перечислениями, закрепил навыки работы с структурами, освоил методы составления многофайловых программ.