## Содержание

[**Общая постановка задачи** 2](#_Toc111607)

[**Требования** 3](#_Toc111608)

[**Тест план** 4](#_Toc111609)

[**Алгоритм бота** 5](#_Toc111610)

[**Диаграмма классов** 6](#_Toc111611)

[**Заключение** 7](#_Toc111612)

[**Список литературы** 8](#_Toc111613)

[**Приложение 1** 9](#_Toc111614)

[**Приложение 2** 41](#_Toc111615)

# Общая постановка задачи

1. Разработать и реализовать алгоритм игры Крестики нолики. Объяснить выбор стратегии.
2. Программа должна быть написана в соответствии со стандартом программирования: C++ Programming Style Guidelines (http://geosoft.no/development/cppstyle.html).

# Требования

Функциональные требования:

1. Передвижение курсора осуществляется за счет стрелок на клавиатуре, при нажатии любых других клавиш, срабатывать не будет
2. Чтобы поставить «крестик» или «нолик» нужно нажать ENTER, при нажатии другой клавиши ничего срабатывать не будет
3. Имя должно состоять из одного слова (желательно на латинице, но не обязательно)

Нефункциональные требования:

1. Используется технология ООП 2. Класс «Игрок» должен содержать:

 Конструктор без параметров, деструктор

1. Для сортировки используется функция, которая выполняет сортировку по возрастанию по одному из полей класса(очкам)
2. Результат выводиться в консоль после того, как игрок больше не хочет играть
3. Программа должна быть написана в соответствии со стандартом программирования: C++ Programming Style Guidelines

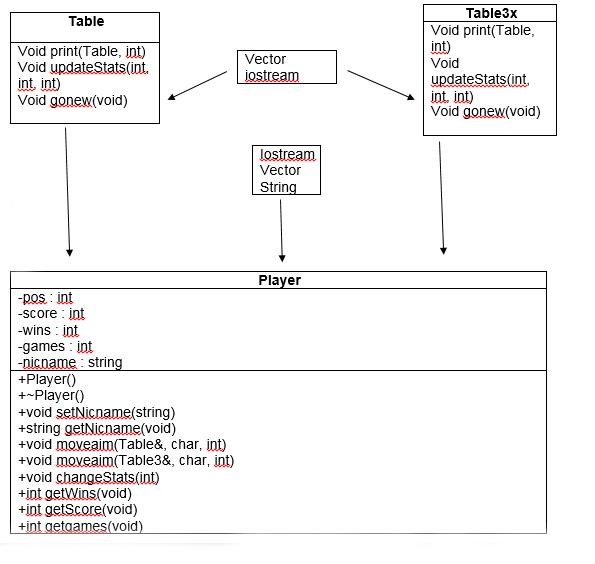
# Тест план

|  |  |  |  |
| --- | --- | --- | --- |
| № | Данные | Ожидаемый результат |  |
| 1 | Выиграл первый игрок (игра 3х3) | Сообщение: «Имя первого игрока» IS WINNER | + |
|  | Выиграл второй игрок (игра 3х3) | Сообщение: «Имя второго игрока» IS WINNER | + |
|  | Ничья (игра 3х3) | Сообщение: DRAW | + |
|  | Выиграл первый игрок (игра 5х5) | Сообщение: «Имя первого игрока» IS WINNER | + |
|  | Выиграл второй игрок (игра 5х5) | Сообщение: «Имя второго игрока» IS WINNER | + |
|  | Выиграл бот (игра 5х5) | Сообщение: BOT IS WINNER | + |
|  | Ничья (игра 5х5) | Сообщение: DRAW | + |

# Алгоритм бота

1. Считывание всего поля
2. Первый ход рандомный
3. Считывание диагоналей, затем строк и столбцов
4. Если в диагонали, строке, столбце есть 3 фишки соперника, бот ставит свою фишку в эту диагональ, столбец или строку
5. Опасность – 3 фишки соперника в диагонали, столбце или строке
6. Если опасности нет, бот заполняет одну диагональ, столбец или строку
7. Если опасности нет, и в другой диагонали, строке или строке у бота больше всего фишек он ставит в ту диагональ, столбец или строку, чтобы выиграть 8. Если есть опасность (3 фишки соперника) и у бота осталось 2 поля в своей строке или диагонали, строке или столбце, он заполняет свою диагональ, строку или столбец

# Диаграмма классов



# Заключение

На общей постановке задачи разработаны классы Player, Table, Table3x, в которых хранятся данные об игроке, и состоянии доски. Был разработан алгоритм бота, которого обыграть невозможно. Изучен алгоритм вывода доски в консоль, который обновляется после какого-то действия. Был разработан алгоритм и функция по нему, которая определяем победителя, и присваивает очки игроку.

После чего в консоль появляется список игроков, которые играли, и их очки.

# Список литературы

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня.- СПб.:Питер, 2002 – 464с.
2. Прата, Стивен. Язык программирования С++. Лекции и упражнения, 6-е изд.: Пер.с англ. – М.: ООО “И.Д. Вильямс”, 2013.- 1248 с.
3. Галина Ваныкина, Татьяна Сундукова. Структуры и алгоритмы компьютерной обработки данных – URL:

<http://www.intuit.ru/studies/courses/648/504/info>

1. Бьерн Страуструп. Язык программирования C++ для профессионалов.– URL: <http://www.intuit.ru/studies/courses/98/98/info>
2. Р. Лафоре. Объектно-ориентированное программирование изд.4, СПб.Питер, 2004
3. Крестики-нолики. Непобедимый искусственный интеллект. URL: <https://toster.ru/q/422134>
4. Игра "Крестики-нолики" формат 5х5. URL: http://www.cyberforum.ru/cppbeginners/thread1876533.html

# Приложение 1

### AI\_Player.h

#pragma once

#include "Table.h"

#include <ctime>

#include <cstdlib> #include <string>

#include <fstream>

#include <iostream>

#include <vector>

class AI\_Player

{

int move\_num = 0; public:

std::ofstream log\_file; std::ofstream mov\_f; std::vector<int> dda; int team; AI\_Player(); ~AI\_Player(); void setteam(int i) { team = i; } int anydanger(Table&); void nextMove(Table&); int findmax(Table&); void init\_log(void); void set\_team(int a) { team = a; } int checkline(Table&, int&);

int myp(Table&, int&);

};

### pch.h

#ifndef PCH\_H

#define PCH\_H

#endif

### Player.h

#pragma once

#include <string>

#include <stdio.h>

#include <cstdlib>

#include <ctime>

#include <cstdlib>

#include <Windows.h>

#include <conio.h>

#include "Table.h"

#include "Table3x.h"

#pragma warning(disable : 4996)

class Player {

int pos = 0; int score; int wins; int games; std::string nicname; public:

Player(); ~Player();

void setNicname(std::string);

std::string getNicname(void) { return nicname; } void moveaim(Table&, char, int); void moveaim(Table3&, char, int); void changeStats(int); int getWins(void) { return wins; } int getScore(void) { return score; } int getgames(void) { return games; }

};

### Table.h

#pragma once

#include <iostream>

#include <vector>

#pragma warning(disable : 4996)

class Table //5x5

{ public:

char table[25];

int i1 = 0, i2 = 0, i3 = 0, j1 = 0, j2 = 0, j3 = 0, d1 = 0, d2 = 0; std::vector<int> data;

Table(); ~Table();

friend void print(Table, int); void updateStats(int, int, int); void gonew(void);

};

### Table3x.h

#pragma once

#include <iostream>

#include <vector>

#pragma warning(disable : 4996)

class Table3 //3x3

{

public:

char table[9];

int i1 = 0, i2 = 0, i3 = 0, j1 = 0, j2 = 0, j3 = 0, d1 = 0, d2 = 0; std::vector<int> data;

Table3();

~Table3();

friend void print(Table3, int);

void updateStats(int, int, int);

void gonew(void);

};

### AI\_Player.cpp

#include "pch.h"

#include "AI\_Player.h"

using namespace std;

AI\_Player::AI\_Player()

{

init\_log(); //открыть лог.тхт

}

AI\_Player::~AI\_Player()

{

} void AI\_Player::init\_log(void){

log\_file.open("log.txt");

}

int AI\_Player::anydanger(Table &t) { // return -1 -> no danger проверяет полностью все значения(строки, столбцы, диагонали), если есть угроза.

//Если в ряду больше 3 и нет его фишек, он ставит

int code = 0;

if (team == 1) { // 1 -> X

for (auto it = t.data.begin(); it != t.data.end(); ++it) { if (\*it <= -3 && myp(t, code) == 0){

return code;

}

++code;

}

return -1;

} if (team == -1) { // -1 -> O for (auto it = t.data.begin(); it != t.data.end(); ++it) {

if (\*it >= 3 && myp(t, code) == 0) {

return code;

}

++code;

}

return -1;

}

}

int AI\_Player::findmax(Table &t) { //где больше его фишек, чтоб выйграть, больше на диагональ уклон

int code = 0; int ret = -1; int max = 0; int min = 0; if (team == 1) {

|  |  |
| --- | --- |
| for (auto it = t.data.begin(); it != t.data.end(); ++it) { | |
|  | if (\*it == max && max != 0) { |
|  | if (checkline(t, code) != 1) { |
|  | ret = code; |
|  | } |
|  | } |
|  | if (\*it > max) { |
|  | if (checkline(t, code) != 1) { |
|  | ret = code; |
|  | } |
|  | } |
| } | ++code; |
| log\_file << endl << "move number " << move\_num << " findmax | |

ret " << ret << " value on that line " << t.data[abs(ret)] << endl << "TEAM " << team

<< endl;

return ret;

} if (team == -1) {

|  |  |
| --- | --- |
| for (auto it = t.data.begin(); it != t.data.end(); ++it) { | |
|  | if (\*it == min && min != 0) { |
|  | srand(time(0)); |
|  | int ch = (rand() % 100); |
|  | if (checkline(t, code) != 1) { |
|  | ret = code; |
|  |  |
|  | } |
|  | } |
|  | if (\*it < min) { |
|  | if (checkline(t, code) != 1) { |
|  | ret = code; |
|  | } |
|  | } |
| } | ++code; |
| log\_file << endl << "move number " << move\_num << "findmax ret | |

" << ret << " value on that line " << t.data[abs(ret)] << endl << "TEAM " << team << endl;

return ret;

}

}

int AI\_Player::checkline(Table &t, int &line) { //возвращает сколько фишек в ряду(1-полный ряд, 3-хотя бы одна вражеская, 0 - нет вражеских и есть пустое место)

int mycounter = 0; int enmycount = 0; int neutral = 0; if (team == 1) {

if (line >= 0 && line < 5) { for (int i = 0; i < 5; ++i) { if (t.table[line \* 5 + i] == 'O') {

++enmycount;

} if (t.table[line \* 5 + i] == 'X') {

|  |  |
| --- | --- |
| ++mycounter;  }  if (t.table[line \* 5 + i] == '@') {  ++neutral;  }  }  }  if (line >= 5 && line < 10) { | |
|  | for (int i = 0; i < 5; ++i) { |
|  | if (t.table[(5 \* i) + (line % 5)] == 'O') { |
|  | ++enmycount; |
|  | } |
|  | if (t.table[(5 \* i) + (line % 5)] == 'X') { |
|  | ++mycounter; |
|  | } |
|  | if (t.table[(5 \* i) + (line % 5)] == '@') { |
|  | ++neutral; |
|  | } |
| } | } |
| if (line == 10) { | |
|  | for (int i = 0; i < 5; ++i) { |
|  | if (t.table[i + 5 \* i] == 'O') { |
|  | ++enmycount; |
|  | } |
|  | if (t.table[i + 5 \* i] == 'X') { |
|  | ++mycounter; |
|  | } |
|  | if (t.table[i + 5 \* i] == '@') { |
|  | ++neutral; |
|  | } |
| } | } |

if (line == 11) {

|  |  |  |
| --- | --- | --- |
|  | for (int i = 0; i < 5; ++i) { | |
|  |  | if (t.table[i \* 4 + 4] == 'O') { |
|  |  | ++enmycount; |
|  |  | } |
|  |  | if (t.table[i \* 4 + 4] == 'X') { |
|  |  | ++mycounter; |
|  |  | } |
|  |  | if (t.table[i \* 4 + 4] == '@') { |
|  |  | ++neutral; |
|  |  | } |
| } | } |  |

}

if (team == -1) {

|  |  |  |
| --- | --- | --- |
| if (line >= 0 && line < 5) { | | |
|  | for (int i = 0; i < 5; ++i) { | |
|  |  | if (t.table[line \* 5 + i] == 'X') { |
|  |  | ++enmycount; |
|  |  | } |
|  |  | if (t.table[line \* 5 + i] == 'O') { |
|  |  | ++mycounter; |
|  |  | } |
|  |  | if (t.table[line \* 5 + i] == '@') { |
|  |  | ++neutral; |
|  |  | } |
| } | } |  |
| if (line >= 5 && line < 10) { for (int i = 0; i < 5; ++i) { if (t.table[line + (i \* 5) - 5] == 'X') {  ++enmycount; | | |

} if (t.table[line + (i \* 5) - 5] == 'O') {

++mycounter;

} if (t.table[line + (i \* 5) - 5] == '@') {

|  |  |
| --- | --- |
| ++neutral;  }  }  }  if (line == 10) { | |
|  | for (int i = 0; i < 5; ++i) { |
|  | if (t.table[i + 5 \* i] == 'X') { |
|  | ++enmycount; |
|  | } |
|  | if (t.table[i + 5 \* i] == 'O') { |
|  | ++mycounter; |
|  | } |
|  | if (t.table[i + 5 \* i] == '@') { |
|  | ++neutral; |
|  | } |
| } | } |
| if (line == 11) { for (int i = 0; i < 5; ++i) { if (t.table[i \* 4 + 4] == 'X') {  ++enmycount;  }  if (t.table[i \* 4 + 4] == 'O') {  ++mycounter;  }  if (t.table[i \* 4 + 4] == '@') {  ++neutral;  }  }  } | |

}

log\_file << endl << "LINE IS " << line << "

values(my/enemy/neutral): " << mycounter << " " << enmycount << " " << neutral << endl; if (neutral == 0) { log\_file << endl << "RETURN " << 1 << endl; return 1;

}

if (enmycount != 0) {

log\_file << endl << "RETURN " << 3 << endl;

return 3;

}

log\_file << endl << "RETURN " << 0 << endl;

return 0;

} void AI\_Player::nextMove(Table &t) {

int i = anydanger(t); bool dangdone = false;

for (auto it = dda.begin(); it != dda.end(); ++it) {

if (\*it == i && i != -1) {

dangdone = true;

}

}

int qq = findmax(t); //линия где выгодней ставить log\_file << "Before move " << endl; for (int i = 0; i < 25; ++i) { log\_file << t.table[i]; if ((i % 5) == 4) { log\_file << endl;

}

}

if (move\_num != 0) {

if (i == -1 || dangdone) { // no danger log\_file << endl << "no danger " << endl; if (qq >= 0 && qq < 5) { //max -> i

srand(time(0)); int delta = (rand() % 5); bool go\_on = true; //----------------------------------------------

-------------------

while (go\_on) {

if (t.table[5 \* qq + delta] == '@') {

if (team == 1) {

t.table[5 \* qq + delta] =

'X';

}

else {

t.table[5 \* qq + delta] =

'O';

}

int i = qq; int j = delta;

t.updateStats(i, j, team);

log\_file << "i= " << i << " j=

" << j << endl;

go\_on = false;

}

else { ++delta;

if (delta > 4) {

delta = 0;

}

}

}

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

-------------------

}

if (qq >= 5 && qq < 10 && !dangdone) { // max -> j

srand(time(0));

int delta = (rand() % 5); bool go\_on = true;

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

-------------------

while (go\_on) {

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| '@') { |  |  |  |  | if (t.table[ (qq % 5) + (delta \* 5) ] == | |
|  |  |  |  |  |  | if (team == 1) { |
| 5) ] = 'X'; |  |  |  |  |  | t.table[ (qq % 5) + (delta \* |
|  |  |  |  |  |  | } |
|  |  |  |  |  |  | else { |
| 5)] = 'O'; |  |  |  |  |  | t.table[ (qq % 5) + (delta \* |
|  |  |  |  |  |  | } |
|  |  |  |  |  |  | int i = delta; |
|  |  |  |  |  |  | int j = qq - 5; |
|  |  |  |  |  |  | t.updateStats(i, j, team); |
| " << j << endl; |  |  |  |  |  | log\_file << "i= " << i << " j= |
|  |  |  |  |  |  | go\_on = false; |
|  |  |  |  |  | } |  |
|  |  |  |  |  | else { | |
|  |  |  |  |  | ++delta; | |
|  |  |  |  |  | if (delta > 4) { | |
|  |  |  |  |  | delta = 0; | |
|  |  |  |  |  | } | |
|  |  |  |  |  | } | |
|  |  |  |  | } |  | |
| ------------------- |  |  |  | //---------------------------------------------- | | |
|  |  |  | } |  | | |
|  |  |  | if (qq == 10) { //max -> main diagonal | | | |
|  |  |  | srand(time(0)); | | | |
|  |  |  | int delta = (rand() % 5); | | | |
|  |  |  | bool go\_on = true; | | | |
|  |  |  | while (go\_on) { | | | |
|  |  |  | if (t.table[6 \* delta] == '@') { if (team == 1) {  t.table[6 \* delta] = 'X';  } | | | |

else {

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | t.table[6 \* delta] = 'O'; | |
|  | |  |  |  | } | |
|  | |  |  |  | go\_on = false; | |
|  | |  |  |  | t.updateStats(delta, delta, team); | |
| j= " << delta << endl; | |  |  |  | log\_file << "i= " << delta << " | |
|  |  |  |  |  | } | |
|  |  |  |  |  | else { | |
|  |  |  |  |  | ++delta; | |
|  |  |  |  |  | if (delta > 4) { | |
|  |  |  |  |  | delta = 0; | |
|  |  |  |  |  | } | |
|  |  |  |  |  | } | |
|  |  |  |  | } |  | |
|  |  |  | } |  |  | |
|  |  |  | if (qq == 11) { //max -> side diagonal | | | |
|  |  |  |  | srand(time(0)); | | |
|  |  |  |  | int delta = (rand() % 5); | | |
|  |  |  |  | bool go\_on = true; | | |
|  |  |  |  | while (go\_on) { | | |
|  |  |  |  | if (t.table[4 \* (delta + 1)] == '@') { | | |
|  |  |  |  |  | | if (team == 1) { |
| 'X'; |  |  |  |  | | t.table[4 \* (delta + 1)] = |
|  |  |  |  |  | | } |
|  |  |  |  |  | | else { |
| 'O'; |  |  |  |  | | t.table[4 \* (delta + 1)] = |
|  |  |  |  |  | | } |
|  |  |  |  |  | | go\_on = false; |
| team); |  |  |  |  | | t.updateStats(delta, 4 - delta, |
|  |  |  |  |  | | log\_file << "i= " << delta << " |
| j= " << 4 - delta << endl; | | |
|  | | |  | } | |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | else {  ++delta; | | | | |  |
|  |  | if (delta > 4) { | | | | |  |
|  |  | delta = 0; | | | | |  |
|  |  | } | | | | |  |
|  |  | } | | | | |  |
|  |  | } | | | | |  |
|  |  | } | | | | |  |
|  |  | } | | | | |  |
|  |  | else { // there is danger | | | | |  |
|  |  | log\_file << endl << "Danger " << i << endl; | | | | |  |
|  |  |  | if (i >= 0 && i < 5) { | | | | //max -> i |
|  |  |  |  | srand(time(0)); | | |  |
|  |  |  |  | int delta = (rand() % 5); | | |  |
|  |  |  |  | bool go\_on = true; | | |  |
|  |  |  |  |  | | |  |
| ------------------- |  |  |  | //---------------------------------------------- | | | |
|  |  |  |  | while (go\_on) { | | | |
|  |  |  |  |  | if (t.table[5 \* i + delta] == '@') { | | |
|  |  |  |  |  |  | if (team == 1) { | |
| 'X'; |  |  |  |  |  | t.table[5 \* i + delta] = | |
|  |  |  |  |  |  | } | |
|  |  |  |  |  |  | else { | |
| 'O'; |  |  |  |  |  | t.table[5 \* i + delta] = | |
|  |  |  |  |  |  | } | |
|  |  |  |  |  |  | int tt = i; | |
|  |  |  |  |  |  | int j = delta; | |
|  |  |  |  |  |  | t.updateStats(tt, j, team); | |
|  |  |  |  |  |  | go\_on = false; | |
|  |  |  |  |  | } |  | |
|  |  |  |  |  | else { | | |
|  |  |  |  |  | ++delta; | | |
|  |  |  |  |  | if (delta > 4) { | | |

delta = 0;

}

}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | } | | | |
| ------------------- |  |  | //---------------------------------------------- | | | |
|  |  |  | } | | | |
|  |  |  | if (i >= 5 && i < 10) { // max -> j | | | |
|  |  |  |  | srand(time(0)); | | |
|  |  |  |  | int delta = (rand() % 5); | | |
|  |  |  |  | bool go\_on = true; | | |
|  |  |  |  |  | | |
| ------------------- |  |  |  | //---------------------------------------------- | | |
|  |  |  |  | while (go\_on) { | | |
|  |  |  |  |  | if (t.table[5 \* delta + i] == '@') { | |
|  |  |  |  |  |  | if (team == 1) { |
| 'X'; |  |  |  |  |  | t.table[5 \* delta + i] = |
|  |  |  |  |  |  | } |
|  |  |  |  |  |  | else { |
| 'O'; |  |  |  |  |  | t.table[5 \* delta + i] = |
|  |  |  |  |  |  | } |
|  |  |  |  |  |  | int tt = delta; |
|  |  |  |  |  |  | int j = i; |
|  |  |  |  |  |  | t.updateStats(tt, j, team); |
|  |  |  |  |  |  | go\_on = false; |
|  |  |  |  |  | } |  |
|  |  |  |  |  | else { | |
|  |  |  |  |  | ++delta; | |
|  |  |  |  |  | if (delta > 4) { | |
|  |  |  |  |  | delta = 0; | |
|  |  |  |  |  | } | |
|  |  |  |  |  | } | |
|  |  |  |  | } |  | |

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

}

if (i == 10) { //max -> main diagonal

srand(time(0));

int delta = (rand() % 5); bool go\_on = true; while (go\_on) {

if (t.table[6 \* delta] == '@') {

if (team == 1) {

t.table[6 \* delta] = 'X';

}

else {

t.table[6 \* delta] = 'O';

}

go\_on = false;

t.updateStats(delta, delta, team);

}

else { ++delta;

if (delta > 4) {

delta = 0;

}

}

}

}

if (i == 11) { //max -> side diagonal

srand(time(0));

int delta = (rand() % 5); bool go\_on = true; while (go\_on) {

if (t.table[4 \* (delta + 1)] == '@') {

if (team == 1) {

t.table[4 \* (delta + 1)] =

'X';

}

else {

t.table[4 \* (delta + 1)] =

'O';

}

go\_on = false;

t.updateStats(delta, 4 - delta, team);

}

else { ++delta;

if (delta > 4) {

delta = 0;

}

}

}

}

++move\_num;

}

}

else { srand(time(0)); int delta = rand(); bool go\_on = true; while (go\_on) {

int uu = delta % 25; if (t.table[uu] == '@') {

if (team == 1) {

t.table[uu] = 'X';

}

else {

t.table[uu] = 'O';

}

int i = uu / 5; int j = uu % 5;

t.updateStats(i, j, team); ++move\_num; go\_on = false;

}

|  |  |
| --- | --- |
|  | else { |
|  | delta = delta \* 3; |
|  | delta -= qq; |
|  | delta += delta % 17; |
|  |  |
| }  } | } |
|  | log\_file << endl << "After move" << endl; |
|  | for (int i = 0; i < 25; ++i) { |
|  | log\_file << t.table[i]; |
|  | if ((i % 5) == 4) { |
|  | log\_file << endl; |
|  | } |
|  | } |

} int AI\_Player::myp(Table& t, int& line) { //возвращает кол-во фишек игрока

int mycounter = 0; int enmycount = 0; int neutral = 0; if (team == 1) {

if (line >= 0 && line < 5) { for (int i = 0; i < 5; ++i) { if (t.table[line \* 5 + i] == 'O') {

++enmycount;

}

if (t.table[line \* 5 + i] == 'X') {

++mycounter;

}

if (t.table[line \* 5 + i] == '@') {

++neutral;

}

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| if (line >= 5 && line < 10) { | | | |
|  | for (int i = 0; i < 5; ++i) { | | |
|  | if (t.table[(5 \* i) + (line % 5)] == 'O') { | | |
|  | ++enmycount; | | |
|  | } | | |
|  | if (t.table[(5 \* i) + (line % 5)] == 'X') { | | |
|  | ++mycounter; | | |
|  | } | | |
|  | if (t.table[(5 \* i) + (line % 5)] == '@') { | | |
|  | ++neutral; | | |
|  | } | | |
| } | } | | |
| if (line == 10) { for (int i = 0; i < 5; ++i) { if (t.table[i + 5 \* i] == 'O') {  ++enmycount;  }  if (t.table[i + 5 \* i] == 'X') {  ++mycounter;  }  if (t.table[i + 5 \* i] == '@') {  ++neutral;  }  }  }  if (line == 11) { for (int i = 0; i < 5; ++i) { if (t.table[i \* 4 + 4] == 'O') {  ++enmycount;  } | | | |
|  |  | if (t.table[i \* 4 + 4] == 'X') { |
|  |  | ++mycounter; |
|  |  | } |
|  |  | if (t.table[i \* 4 + 4] == '@') { |
|  |  | ++neutral; |
|  |  | } |
| } | } |  |

}

if (team == -1) {

|  |  |  |
| --- | --- | --- |
| if (line >= 0 && line < 5) { | | |
|  | for (int i = 0; i < 5; ++i) { | |
|  |  | if (t.table[line \* 5 + i] == 'X') { |
|  |  | ++enmycount; |
|  |  | } |
|  |  | if (t.table[line \* 5 + i] == 'O') { |
|  |  | ++mycounter; |
|  |  | } |
|  |  | if (t.table[line \* 5 + i] == '@') { |
|  |  | ++neutral; |
|  |  | } |
| } | } |  |
| if (line >= 5 && line < 10) { for (int i = 0; i < 5; ++i) { if (t.table[line + (i \* 5) - 5] == 'X') {  ++enmycount;  }  if (t.table[line + (i \* 5) - 5] == 'O') {  ++mycounter;  }  if (t.table[line + (i \* 5) - 5] == '@') {  ++neutral;  }  } | | |

} if (line == 10) {

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | for (int i = 0; i < 5; ++i) { | |
|  |  | if (t.table[i + 5 \* i] == 'X') { | |
|  |  | ++enmycount; | |
|  |  | } | |
|  |  | if (t.table[i + 5 \* i] == 'O') { | |
|  |  | ++mycounter; | |
|  |  | } | |
|  |  | if (t.table[i + 5 \* i] == '@') { | |
|  |  | ++neutral; | |
|  |  | } | |
|  |  | } | |
|  | } |  | |
|  | if (line == 11) { | | |
|  |  | for (int i = 0; i < 5; ++i) { | |
|  |  |  | if (t.table[i \* 4 + 4] == 'X') { |
|  |  |  | ++enmycount; |
|  |  |  | } |
|  |  |  | if (t.table[i \* 4 + 4] == 'O') { |
|  |  |  | ++mycounter; |
|  |  |  | } |
|  |  |  | if (t.table[i \* 4 + 4] == '@') { |
|  |  |  | ++neutral; |
|  |  |  | } |
|  |  | } |  |
| } | } |  |  |
| return mycounter; | | | |

}

### Source.cpp

#include "pch.h"

#include "Player.h"

#include "Table.h"

#include "Table3x.h"

#include <iostream>

#include <vector>

#include <string>

#include "AI\_Player.h"

#pragma warning(disable : 4996)

using namespace std;

bool isWin(Table t) //таблица(аргумент)

{

for (auto it = t.data.begin(); it != t.data.end(); ++it)

{

if (abs(\*it) == 5)

{

return false;

}

}

return true;

}

bool isWin3(Table3 t) //таблица(аргумент)

{

for (auto it = t.data.begin(); it != t.data.end(); ++it)

{

if (abs(\*it) == 3)

{

return false;

}

}

return true;

}

void Game(Player &p1, Player &p2, Table &t)

{

int moves = 0; bool draw = false;

while (isWin(t) && !draw)

{

++moves;

if (moves == 25)

{

cout << "DRAW" << endl; p1.changeStats(0); p2.changeStats(0); draw = true;

}

if (!draw)

{

p1.moveaim(t, 'X', 1);

if (!isWin(t))

{

cout << p1.getNicname() << " IS WINNER" << endl;

p1.changeStats(1);

p2.changeStats(-1);

return;

}

}

++moves;

if (moves == 25)

{

cout << "DRAW" << endl; p1.changeStats(0); p2.changeStats(0); draw = true;

}

if(!draw)

{

p2.moveaim(t, 'O', -1);

if (!isWin(t))

{

cout << p2.getNicname() << " IS WINNER" << endl;

p2.changeStats(1);

p1.changeStats(-1);

return;

}

}

}

}

void Game3(Player& p1, Player& p2, Table3& t)

{

int moves = 0; bool draw = false;

while (isWin3(t) && !draw)

{

++moves;

if (moves == 9)

{

cout << "DRAW" << endl; p1.changeStats(0); p2.changeStats(0);

draw = true;

}

if (!draw)

{

p1.moveaim(t, 'X', 1);

if (!isWin3(t))

{

cout << p1.getNicname() << " IS WINNER" << endl;

p1.changeStats(1); p2.changeStats(-1); return;

} }

++moves;

if (moves == 9)

{

cout << "DRAW" << endl; p1.changeStats(0); p2.changeStats(0);

draw = true;

}

if (!draw)

{

p2.moveaim(t, 'O', -1); if (!isWin3(t))

{

cout << p2.getNicname() << " IS WINNER" << endl; p2.changeStats(1);

p1.changeStats(-1);

return;

}

}

}

}

void Game(Player &p1, Table &t)

{

AI\_Player dumb;

int moves = 0; int q;

cout << "Your team?(1-X)"; cin >> q; bool draw = false;

if (q == 1)

{

dumb.team = -1;

while (isWin(t) && !draw)

{

++moves;

if (moves >= 25)

{

cout << "DRAW" << endl; p1.changeStats(0);

draw = true;

}

if (!draw)

{

p1.moveaim(t, 'X', 1); if (!isWin(t))

{

cout << p1.getNicname() << " IS WINNER" << endl;

p1.changeStats(1);

return;

} }

++moves;

if (moves >= 25)

{

cout << "DRAW" << endl;

//p1.changeStats(0);

draw = true;

}

if (!draw)

{

dumb.nextMove(t); if (!isWin(t))

{

cout << "BOT IS WINNER" << endl; p1.changeStats(-1); return;

}

}

}

} else

{ dumb.team = 1;

while (isWin(t) && !draw)

{

++moves;

if (moves >= 25)

{

cout << "DRAW" << endl; p1.changeStats(0); draw = true;

}

if (!draw) { dumb.nextMove(t);

if (!isWin(t))

{

cout << "BOT IS WINNER" << endl; p1.changeStats(-1);

return;

} }

++moves;

if (moves >= 25)

{

cout << "DRAW" << endl; p1.changeStats(0);

draw = true;

}

if (!draw)

{

p1.moveaim(t, 'O', -1);

if (!isWin(t))

{

cout << p1.getNicname() << " IS WINNER" << endl;

p1.changeStats(1);

return;

}

}

}

}

}

int main()

{

char enter;

cout << "Hello! You can play X & O"<<endl<<"Press ENTER to continue"; enter=cin.get();

if (enter == '\n')

{

system("cls");

}

cout << "The rules: " << endl <<

"Choose a board on which you want to play, 3x3 or 5x5." << endl <<

"If you chose a 3x3 board, then you can only play against a friend, if 5x5, both against a friend and against a computer."<<endl<<

"Press ENTER to continue";

enter = cin.get(); if (enter == '\n')

{

system("cls");

}

cout << "control (3x3): field number and enter" << endl <<

"control (5x5): movement of arrows and enter"<<endl<<

"Points: 0 - lose"<<endl<<

" 5 - draw"<<endl<<

" 10 - win"<<endl<<"Press ENTER to continue"; enter = cin.get(); if (enter == '\n')

{

system("cls");

}

int check = 0;

while (check != 3 && check != 5)

{

cout << "3x3 or 5x5??(3 for 3x3 and 5 for 5x5): ";

cin >> check;

}

if (check == 5)

{

//5x5

vector<Player> players; Player dummy; string text; Table t; int i = 0;

char responce;

//add player

cout << "enter nick : " << endl; cin >> text;

players.push\_back(dummy);

players[i].setNicname(text);

++i;

bool cont = true;

int a1; int a2;

while (cont)

{

t.gonew(); //новая доска(все пустое) cout << "Play a game?(1 - pvp; 2 - pve(bot); 0 - end Game)" << endl; cin >> a1;

if (a1 == 1)

{

cout << "enter nick : " << endl;

cin >> text;

players.push\_back(dummy);

players[i].setNicname(text);

++i;

cout << "Number of player to play as X ";

cin >> a1;

cout << "Number of player to play as O "; cin >> a2;

--a1;

--a2;

if (a1 != a2 && a1 > -1 && a1 < i && a2 > -1 && a2 < i)

//номер игрока, проверка

{

Game(players[a1], players[a2], t);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | } | |
|  |  | } |  | |
|  |  | else |  | |
|  |  | { |  | |
|  |  |  | if (a1 == 2) | |
|  |  |  | { |  |
|  |  |  |  | if (i > 1) |
|  |  |  |  | { |
|  |  |  |  | cout << "NUMBER OF PLAYER AGAINST BOT"; |
|  |  |  |  | cin >> a1; |
|  |  |  |  | --a1; |
|  |  |  |  | if (a1 > -1 && a1 < i) |
|  |  |  |  | { |
|  |  |  |  | Game(players[a1], t); |
|  |  |  |  | } |
|  |  |  |  | } |
|  |  |  |  | else |
|  |  |  |  | { |
|  |  |  |  | Game(players[0], t); |
|  |  |  |  | } |
|  |  |  | } |  |
|  |  |  | else |  |
|  |  |  | { |  |
|  |  |  |  | cont = false; |
|  |  |  | } |  |
|  |  | } |  |  |
|  | } |  |  |  |
|  | for (auto it = players.begin(); it != players.end(); ++it) | | | |
|  | { | | | |
|  | dummy = \*it; | | | |
| endl; | cout << dummy.getNicname() << " SCORE " << dummy.getScore() << | | | |
| } | } | | | |

if (check == 3)

{

vector<Player> players; Player dummy; string text; Table3 t3; int i = 0;

//add player

cout << "enter nick : " << endl; cin >> text;

players.push\_back(dummy); players[i].setNicname(text);

++i;

cout << "enter nick : " << endl; cin >> text;

players.push\_back(dummy); players[i].setNicname(text);

++i;

int a1,a2;

cout << "Number of player to play as X "; cin >> a1;

cout << "Number of player to play as O "; cin >> a2;

--a1; --a2;

if (a1 != a2 && a1 > -1 && a1 < i && a2 > -1 && a2 < i) //номер игрока,

проверка

{

Game3(players[a1], players[a2], t3);

}

}

}

Pch.cpp

#include "pch.h"

### Player.cpp

#include "pch.h"

#include "Player.h"

using namespace std;

Player::Player() :score(0), wins(0), games(0)

{

}

Player::~Player()

{

}

void Player::setNicname(string text) { nicname = text;

}

void Player::changeStats(int key) {

games++; switch (key)

{

case(0):

score += 5; break; case(1): score += 10; ++wins; break; case(-1):

break;

}

} void Player::moveaim(Table& t, char c, int uuu) {

int i = 0, j = 0, pos = 0; bool goon = true; while (goon) { char input = getch(); int code = input; if (code == 72) { i--;

} if (code == 80) {

i++;

} if (code == 77) {

j++;

}

if (code == 75) {

j--;

}

if (code == 13) {

if (t.table[pos] == '@'){ pos = j + i \* 5;

t.table[pos] = c;

goon = false;

t.updateStats(i, j, uuu);

}

}

if (j < 0) { j++; } if (j > 4) { j--; } if (i > 4) { i--; } if (i < 0) { i++; } pos = j + i \* 5; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl;

print(t, pos);

}

}

void Player::moveaim(Table3& t, char c, int uuu) { int i = 0, j = 0, pos = 0; bool goon = true; while (goon) { char input = getch(); int code = input; if (code == 72) {

i--;

}

if (code == 80) {

i++;

}

if (code == 77) {

j++;

}

if (code == 75) {

j--;

}

if (code == 13) { if (t.table[pos] == '@') { pos = j + i \* 3;

t.table[pos] = c; goon = false;

t.updateStats(i, j, uuu);

}

}

if (j < 0) { j++; } if (j > 2) { j--; } if (i > 2) { i--; } if (i < 0) { i++; } pos = j + i \* 3; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl;

cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl; cout << endl;

print(t, pos);

}

}

### Table.cpp

#include "pch.h"

#include "Table.h"

using namespace std;

Table::Table() {

for (int i = 0; i < 25; ++i) {

table[i] = '@';

}

for (int i = 0; i < 12; ++i) { data.push\_back(0);

}

}

Table::~Table()

{

}

void print(Table t, int pos) {

for (int i = 0; i < 25; ++i) {

if (i == pos) {

cout << "\*";

}

else {

cout << t.table[i];

}

if ((i % 5) == 4) { cout << endl;

}

}

} void Table::updateStats(int i, int j, int upd){ if (i == j) {

data[10] += upd;

}

if (i == 0) { data[0] += upd; }

if (i == 1) { data[1] += upd;

}

if (i == 2) {

data[2] += upd;

}

if (i == 3) {

data[3] += upd;

}

if (i == 4) { data[4] += upd;

}

if (j == 0) { data[5] += upd;

}

if (j == 1) { data[6] += upd;

}

if (j == 2) {

data[7] += upd;

}

if (j == 3) { data[8] += upd;

}

if (j == 4) { data[9] += upd;

}

if (i + j == 4) { data[11] += upd;

}

}

void Table::gonew(void) { for (auto it = data.begin(); it != data.end(); ++it) {

\*it = 0;

}

for (int i = 0; i < 25; ++i) { table[i] = '@';

}

}

### Table3x.cpp

#include "pch.h"

#include "Table3x.h"

using namespace std;

Table3::Table3()

{

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

table[i] = '@';

}

for (int i = 0; i < 12; ++i) { data.push\_back(0);

}

}

Table3::~Table3()

{

}

void print(Table3 t, int pos) { for (int i = 0; i < 9; ++i) {

if (i == pos) {

cout << "\*";

}

else {

cout << t.table[i];

}

if ((i % 3) == 2) { cout << endl;

}

}

}

void Table3::updateStats(int i, int j, int upd) {

if (i == j) { data[10] += upd;

}

if (i == 0) { data[0] += upd;

}

if (i == 1) { data[1] += upd;

}

if (i == 2) {

data[2] += upd;

}

if (j == 0) { data[5] += upd;

}

if (j == 1) { data[6] += upd;

}

if (j == 2) { data[7] += upd;

}

if (i + j == 2) {

data[11] += upd;

}

}

void Table3::gonew(void) {

for (auto it = data.begin(); it != data.end(); ++it) { \*it = 0;

}

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

table[i] = '@';

}

}

# Приложение 2

