Universidad Nacional Experimental de Guayana

Vicerrectorado Académico

Coordinación General de Pregrado

Proyecto de Carrera: Ingeniería Informática

Cátedra: Estructura de Datos

Sección: 1

**Informe – 3 en raya C++**

Profesor: Alumno:

Carlos Abaffy Lorenzo Parra CI: V-28727109

Ciudad Guayana, Junio 2023

**ANALISIS DEL PROBLEMA**

Se desea que se realice el juego 3 en raya también conocido como Tic-Tac-Toe. Bajo los siguientes criterios:

* El juego debe admitir los siguientes modos de juegos:

1. Jugador vs Jugador.
2. Un jugador es humano y el otro es computador
3. Los dos jugadores son computadoras

* Cada jugada del juego debe ser almacenada en un archivo .txt con la siguiente estructura:

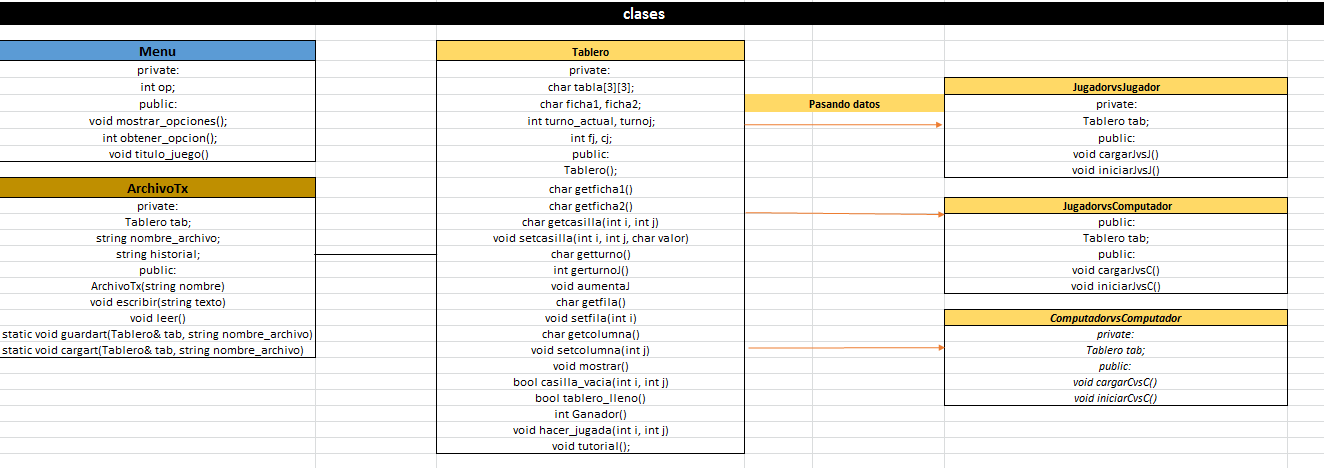
1. #Movimiento, tiempo, Id Jugador, tipo Jugador (Humano o AI), Símbolo Jugador=”X/O” [fila][columna]

Ejemplo:

Move1, 20, 01, Humano, X, [0],[1];

* El programa tiene la característica que un usuario puede cargar un archivo de un juego previo y continuarlo si es posible.
* El programa debe ser realizado usando el modelo de 3 capas (capa de presentación, capa de negocio, capa de datos).

**ALGORITMO GENERAL**



**ESTRUCTURA DE DATOS**

Clase Tablero (clase padre)

|  |  |
| --- | --- |
| Atributos | Metodos |
| char tabla[3][3] | char getficha1()const |
| char ficha1 | char getficha2()const |
| char ficha2 | char getcasilla(int i, int j)const |
| int turno\_actual | void setcasilla(int i, int j, char valor) |
| int turno\_j | char getturno()const |
| int fj | int gerturnoJ() |
| int cj; | void aumentaJ() |
|  | char getfila()const |
|  | void setfila(int i) |
|  | char getcolumna()const |
|  | void setcolumna(int j) |
|  | void mostrar()const |
|  | bool casilla\_vacia(int i, int j) |
|  | bool tablero\_lleno() |
|  | int Ganador() |
|  | void hacer\_jugada(int i, int j) |
|  | void tutorial() |
|  |  |
|  |  |

Clase ArchivoTx

|  |  |
| --- | --- |
| Atributos | Métodos |
| Tablero tab | void escribir(string texto) |
| String nombre\_archivo | void leer() |
| String historial | static void guardart(Tablero& tab, string nombre\_archivo) |
|  | static void cargart(Tablero& tab, string nombre\_archivo) |

Clase Menu

|  |  |
| --- | --- |
| Atributos | Metodos |
| Int op | void titulo\_juego() |
|  | void mostrar\_opciones() |
|  | int obtener\_opcion() |
|  |  |
|  |  |

Clase JugadorvsJugador (Hija de clase Tablero)

|  |  |
| --- | --- |
| Atributos | Métodos |
| Tablero tab | void cargarJvsJ() |
|  | void iniciarJvsJ() |

Clase JugadorvsComputador (Hija de clase Tablero)

|  |  |
| --- | --- |
| Atributos | Métodos |
| Tablero tab | void cargarJvsC() |
|  | void iniciarJvsC() |

Clase ComputadorvsComputador (Hija de clase tablero)

|  |  |
| --- | --- |
| Atributos | Metodos |
| Tablero tab | void cargarCvsC() |
|  | void iniciarCvsC() |

**CODIGO FUENTE**

|  |
| --- |
| */\*PROYECTO DE 3 EN LINEA*  *ALUMNOS:*  *Rosangela Brito*  *Lorenzo Parra*  *Cristian Sierra*  *\*/*  **#include <iostream>**  **#include <ctime>**  **#include <fstream>**  **#include <chrono>**  **using** **namespace** std;  */\*----------------------------------------------------------------------------------------------*  *CLASE TABLERO*  *-------------------------------------------------------------------------------------------------\*/*  **class** Tablero{  private:  **char** tabla[3][3]; *// Matriz que representa el estado actual del tablero*  **char** ficha1, ficha2; *// Ficha del jugador (X o O)*  **int** turno\_actual, turnoj;  **int** fj, cj; *//donde se colocÃ³ la ficha en el Ãºltimo turno*      public:  *// Constructor de la clase*  Tablero(){    *// Inicializar el tablero con todas las casillas vacÃ­as*  **for**(**int** i=0;i<3;i++){  **for**(**int** j=0;j<3;j++){  tabla[i][j]='-';  }  }    *// Inicializar las fichas de los jugadores*  ficha1 = 'X';  ficha2 = 'O';    turno\_actual= 'X';  turnoj=0;  }    *// MÃ©todos para obtener y establecer la ficha del jugador 1*  **char** getficha1()**const**{  **return** ficha1;  }    *// MÃ©todos para obtener y establecer la ficha del jugador 2*  **char** getficha2()**const**{  **return** ficha2;  }    *// MÃ©todospara obtener y establecer el estado de una casilla en el tablero*  **char** getcasilla(**int** i, **int** j)**const**{  **return** tabla[i][j];  }    **void** setcasilla(**int** i, **int** j, **char** valor){  tabla[i][j]= valor;  }    *// MÃ©todos para obtener y establecer el turno actual*  **char** getturno()**const**{  **return** turno\_actual;  }    **int** gerturnoJ(){  **return** turnoj;  }    **void** aumentaJ(){  turnoj++;  }  *// MÃ©todos para obtener y establecer la fila y columna de la Ãºltima jugada*  **char** getfila()**const**{  **return** fj;  }  **void** setfila(**int** i){  fj= i;  }    **char** getcolumna()**const**{  **return** cj;  }    **void** setcolumna(**int** j){  cj= j;  }    *// MÃ©todo para mostrar el estado actual del tablero*  **void** mostrar()**const**{ *//Mostrar (Linea 17 hasta la 25)*  cout<<endl<<endl<<endl;  cout<<"\t\t\t\t "<<tabla[0][0]<<" | "<<tabla[0][1]<<" | "<<tabla[0][2]<<" "; *// - | - | -*  cout<<"\n\t\t\t\t---|---|---\n"; *// ----|---|-----*  cout<<"\t\t\t\t "<<tabla[1][0]<<" | "<<tabla[1][1]<<" | "<<tabla[1][2]<<" "; *// - | - | -*  cout<<"\n\t\t\t\t---|---|---\n"; *// ----|---|-----*  cout<<"\t\t\t\t "<<tabla[2][0]<<" | "<<tabla[2][1]<<" | "<<tabla[2][2]<<" "; *// - | - | -*  cout<<"\n\t\t\t\t---|---|---\n";  cout<<endl<<endl<<endl;  }    *// MÃ©todo para verificar si una casilla estÃ¡ vacÃ­a*  **bool** casilla\_vacia(**int** i, **int** j){  **return** tabla[i][j] == '-';  }  *// MÃ©todo para verificar si el tablero estÃ¡ lleno*  **bool** tablero\_lleno(){  **for** (**int** i = 0; i < 3; i++) {  **for** (**int** j = 0; j < 3; j++) {  **if** (tabla[i][j] == '-') {  **return** false;  }  }  }  **return** true;  }  **int** Ganador(){ *//Funcion que permite analizar si ha terminado la partida.*    *//Jugador X*  **for**(**int** i=0; i<3; i++){ *//Comprobar filas*  **if**((tabla[i][0]== ficha1) && tabla[i][0]== tabla[i][1] && tabla[i][1]== tabla[i][2]){  **return** ficha1; *//Existe ganador en supuesta fila i*  }  }    **for**(**int** j=0; j<3; j++){ *//Comprobar columnas*  **if**((tabla[0][j]== ficha1) && tabla[0][j]== tabla[1][j] && tabla[1][j]== tabla[2][j]){  **return** ficha1; *//Existe ganador en supuesta columnas j*  }  }    *//Comprobar diagonales*    **if**((tabla[0][0]== ficha1) && tabla[0][0]== tabla[1][1] && tabla[1][1]== tabla[2][2]){  **return** ficha1; *//Existe ganador en la primera diagonal*  }    **if**((tabla[2][0]== ficha1) && tabla[2][0]== tabla[1][1] && tabla[1][1]== tabla[0][2]){  **return** ficha1; *//Existe ganador en la segunda diagonal*  }    *//Jugador O*  **for**(**int** i=0; i<3; i++){ *//Comprobar filas*  **if**(( tabla[i][0]== ficha2) && tabla[i][0]== tabla[i][1] && tabla[i][1]== tabla[i][2]){  **return** ficha2; *//Existe ganador en supuesta fila i*  }  }    **for**(**int** j=0; j<3; j++){ *//Comprobar columnas*  **if**((tabla[0][j]== ficha2) && tabla[0][j]== tabla[1][j] && tabla[1][j]== tabla[2][j]){  **return** ficha2; *//Existe ganador en supuesta columnas j*  }  }    *//Comprobar diagonales*    **if**((tabla[0][0]== ficha2) && tabla[0][0]== tabla[1][1] && tabla[1][1]== tabla[2][2]){  **return** ficha2; *//Existe ganador en la primera diagonal*  }    **if**((tabla[2][0]== ficha2) && tabla[2][0]== tabla[1][1] && tabla[1][1]== tabla[0][2]){  **return** ficha2; *//Existe ganador en la segunda diagonal*  }    **return** 0; *//No hay ganador*  }    *// MÃ©todo para hacer una jugada*  **void** hacer\_jugada(**int** i, **int** j) {  *// Colocar la ficha del jugador en la casilla correspondiente*  **if** (turno\_actual == ficha1) {  tabla[i][j] = ficha1;  fj = i;  cj = j;  } **else** {  tabla[i][j] = ficha2;  fj = i;  cj = j;  }  *// Cambiar el turno al siguiente jugador*  **if** (turno\_actual == ficha1) {  turno\_actual = ficha2;  } **else** {  turno\_actual = ficha1;  }  }    **void** tutorial(){  *// cout<<"================================================================================\n";*  cout<<"\n\n\tA continuacion se le muestra todas las posiciones a elegir: "<<endl<<endl;  cout<<"\t\t\t[0][0] | [0][1] | [0][2] "; *// - | - | -*  cout<<"\n\t\t\t-------|--------|-------\n"; *// ----|---|-----*  cout<<"\t\t\t[1][0] | [1][1] | [1][2] "; *// - | - | -*  cout<<"\n\t\t\t-------|--------|-------\n"; *// ----|---|-----*  cout<<"\t\t\t[2][0] | [2][1] | [2][2] "; *// - | - | -*  cout<<"\n\t\t\t-------|--------|-------\n";  cout<<endl<<endl;  *// cout<<"================================================================================\n";*  system ("pause");  }  };  */\*----------------------------------------------------------------------------------------------*  *CLASE ARCHIVO*  *-------------------------------------------------------------------------------------------------\*/*  **class** ArchivoTx{  private:  Tablero tab;  string nombre\_archivo;  string historial;  public:    ArchivoTx(string nombre){  nombre\_archivo= nombre;  historial= "";  }    *// MÃ©todo para escribir el estado actual del tablero en un archivo de texto*  **void** escribir(string texto){  ofstream archivo(nombre\_archivo, std::ios::app);  **if**(archivo.is\_open()){    historial = historial + texto + "\n";  archivo<<historial;    archivo.close();    cout<<"Se guardo el archivo "<<nombre\_archivo<<endl;  }**else**{  cout<<"No se pudo abrir el archivo "<<nombre\_archivo<<endl;  }  }    *// MÃ©todo para leer el estado actual del tablero desde un archivo de texto*  **void** leer(){  ifstream archivo(nombre\_archivo);  **if**(archivo.is\_open()){  string linea;  **while**(getline(archivo, linea)){  cout<<linea<<endl;  }  archivo.close();      cout<<"Se cargo el archivo "<<nombre\_archivo<<endl;  }**else**{  cout<<"No se pudo abrir el archivo "<<nombre\_archivo<<endl;  }  }    **static** **void** guardart(Tablero& tab, string nombre\_archivo){  ofstream archivo1(nombre\_archivo);  **if**(archivo1.is\_open()){  **for**(**int** i=0; i<3; i++){  **for**(**int** j=0; j<3; j++){    **char** valor= tab.getcasilla(i, j);  archivo1<<valor<<" ";    }  archivo1<<endl;  }  archivo1.close();  }**else**{  cout<<"No se pudo abrir el archivo"<<endl;  }  }    **static** **void** cargart(Tablero& tab, string nombre\_archivo){  ifstream archivo1(nombre\_archivo);  **char** estado[3][3];  **if**(archivo1.is\_open()){  **for**(**int** i=0; i<3; i++){  **for**(**int** j=0; j<3; j++){    **char** valor;  archivo1>>valor;  cout<<valor;  **if**(valor=='X' || valor=='O'){  tab.setcasilla(i, j, valor);  }**else**{  tab.casilla\_vacia(i, j);  }    }  cout<<endl;  }    archivo1.close();  }**else**{  cout<<"No se pudo abrir el archivo"<<endl;  }  }      };  */\*----------------------------------------------------------------------------------------------*  *CLASE MENU*  *-------------------------------------------------------------------------------------------------\*/*  **class** Menu{  private:  **int** op;    public:    *// Constructor de la clase*  Menu() {  op = 0;  }    *// MÃ©todo para mostrar las opciones del menÃº*    **void** titulo\_juego(){    cout<<" ####### ######## ## ## ######## ### ## ## ### \n";  cout<<" ## ## ## ### ## ## ## ## ## ## ## ## ## \n";  cout<<" ## ## #### ## ## ## ## ## #### ## ## \n";  cout<<" ####### ###### ## ## ## ######## ## ## ## ## ## \n";  cout<<" ## ## ## #### ## ## ######### ## ######### \n";  cout<<" ## ## ## ## ### ## ## ## ## ## ## ## \n";  cout<<" ####### ######## ## ## ## ## ## ## ## ## ## \n\n\n";  *// cout<<"----------------------------------------------------------------------------\n\n\n";*  }  **void** mostrar\_opciones(){  cout << "====================== Seleccione una opcion ======================" << endl << endl;  cout << "\t\t1.- Jugador vs Jugador" << endl;  cout << "\t\t2.- Jugador vs Computadora" << endl;  cout << "\t\t3.- Computadora vs Computadora" << endl;  cout << "\t\t4.- Instrucciones para jugar" << endl ;  cout << "\t\t5.- Salir" << endl << endl ;  cout << "\t\t Ingrese su opcion: " ;  cin >> op;  }  *// MÃ©todo para obtener la opciÃ³n seleccionada por el usuario*  **int** obtener\_opcion() **const** {  **return** op;  }  };  */\*----------------------------------------------------------------------------------------------*  *CLASE JUGADOR VS JUGADOR*  *-------------------------------------------------------------------------------------------------\*/*  **class** JugadorvsJugador: **public** Tablero{    private:  Tablero tab;    public:    **void** cargarJvsJ(){  **int** conf;    cout<<"Desea cargar una partida anterior Si(1)/No(2)?"<<endl;  cin>>conf;    ArchivoTx archivo1 ("JvsJ.txt");    **switch**(conf){  **case** 1:    archivo1.cargart(tab, "JvsJ.txt");  system("cls");  **break**;    **case** 2:    **break**;  }    }    **void** iniciarJvsJ(){    **int** f, c;  **char** ganador, l;      ArchivoTx archivo("Jugadas.txt");    ArchivoTx archivo1 ("JvsJ.txt");    cargarJvsJ();    system("cls");    **while**(true){    **do**{    tab.mostrar();    cout<<"\t\t\tJugador "<<tab.getturno()<<", es su turno\n"<<endl;  **auto** t1= std::chrono::high\_resolution\_clock::now();    **if**(tab.getturno()=='X'){    cout<<"\t\t\tIngrese la fila(0-2): "; cin>>f;  cout<<"\t\t\tIngrese la columna(0-2): "; cin>>c;    **if**(tab.casilla\_vacia(f, c)){  tab.hacer\_jugada(f, c);      **auto** t2= std::chrono::high\_resolution\_clock::now();  *// Medir el tiempo que tarda el jugador 1 en hacer la jugada*  **double** time1= std::chrono::duration\_cast<std::chrono::duration<**double**>>(t2-t1).count();  **double** time2= 0.0;    **int** nJ= 0;  nJ= tab.gerturnoJ();      string texto= "Mov " + to\_string(tab.gerturnoJ()) + " t1: " + to\_string(time1) + " 1 HUMANO " + "X (" + to\_string(f) + ", " + to\_string(c) + ")" + "\n";    tab.aumentaJ();    archivo.escribir(texto);    }**else**{  cout<<"\t\t\tingrese de nuevo"<<endl;  cout<<"\t\t\t";system("pause");  }  }**else**{    **auto** t1= std::chrono::high\_resolution\_clock::now();    cout<<"\t\t\tIngrese la fila(0-2): "; cin>>f;  cout<<"\t\t\tIngrese la columna(0-2): "; cin>>c;    **if**(tab.casilla\_vacia(f, c)){  tab.hacer\_jugada(f, c);    **auto** t2= std::chrono::high\_resolution\_clock::now();  **double** time1= 0.0;  *// Medir el tiempo que tarda el jugador 2 en hacer la jugada*  **double** time2= std::chrono::duration\_cast<std::chrono::duration<**double**>>(t2-t1).count();    **int** nJ= 0;  nJ= tab.gerturnoJ();      string texto= "Mov " + to\_string(tab.gerturnoJ()) + " t2: " + to\_string(time2) + " 2 HUMANO " + "O (" + to\_string(f) + ", " + to\_string(c) + ")" + "\n";    tab.aumentaJ();    archivo.escribir(texto);    }**else**{  cout<<"\t\t\tingrese de nuevo"<<endl;  cout<<"\t\t\t";system("pause");  }    }    archivo1.guardart(tab, "JvsJ.txt");    system("cls");      **if**(tab.tablero\_lleno()){  cout<<"\t\t\tHa sido un empate"<<endl;  cout<<"\t\t\t";system("pause");  **break**;  }    }**while**(!tab.Ganador());    system("cls");    l= tab.tablero\_lleno();  ganador= tab.Ganador();    **if**(ganador){  cout<<"\t\t\tEl jugador "<<ganador<<" ha ganado"<<endl;  cout<<"\t\t\t";system("pause");  **break**;  }    **if**(l!= '-'){  **break**;  }  }  }  };  */\*----------------------------------------------------------------------------------------------*  *CLASE JUGADOR VS COMPUTADOR*  *-------------------------------------------------------------------------------------------------\*/*  **class** JugadorvsComputador : **public** Tablero{  private:  Tablero tab;    public:    **void** cargarJvsC(){  **int** conf;    cout<<"Desea cargar una partida anterior Si(1)/No(2)?"<<endl;  cin>>conf;    ArchivoTx archivo1 ("JvsC.txt");    **switch**(conf){  **case** 1:    archivo1.cargart(tab, "JvsC.txt");  system("cls");  **break**;    **case** 2:    **break**;  }    }    **void** iniciarJvsC(){  **int** f, c;  **char** ganador, l;    ArchivoTx archivo1 ("JvsC.txt");    cargarJvsC();    system ("cls");    **while**(true){    **do**{  ArchivoTx archivo("Jugadas.txt");    tab.mostrar();    cout<<"\t\t\tJugador "<<tab.getturno()<<", es su turno"<<endl;  **auto** t1= std::chrono::high\_resolution\_clock::now();    **if**(tab.getturno()=='X'){    cout<<"\t\t\tIngrese la fila(0-2): "; cin>>f;  cout<<"\t\t\tIngrese la columna(0-2): "; cin>>c;    **if**(tab.casilla\_vacia(f, c)){  tab.hacer\_jugada(f, c);      **auto** t2= std::chrono::high\_resolution\_clock::now();  *// Medir el tiempo que tarda el jugador 1 en hacer la jugada*  **double** time1= std::chrono::duration\_cast<std::chrono::duration<**double**>>(t2-t1).count();  **double** time2= 0.0;    **int** nJ= 0;  nJ= tab.gerturnoJ();      string texto= "Mov " + to\_string(tab.gerturnoJ()) + " t1: " + to\_string(time1) + " 1 HUMANO " + "X (" + to\_string(f) + ", " + to\_string(c) + ")" + "\n";    tab.aumentaJ();    archivo.escribir(texto);    }**else**{  cout<<"\t\t\tingrese de nuevo"<<endl;  cout<<"\t\t\t";system("pause");  }  }**else**{      cout<<"\t\t\tLa computadora esta pensando.."<<endl;  **auto** t1= std::chrono::high\_resolution\_clock::now();  cout<<"\t\t\t";system("pause");  **do**{  f= rand()%3;  c= rand()%3;    }**while**(!tab.casilla\_vacia(f, c));    tab.hacer\_jugada(f, c);    **auto** t2= std::chrono::high\_resolution\_clock::now();  **double** time1= 0.0;  *// Medir el tiempo que tarda el jugador 2 en hacer la jugada*  **double** time2= std::chrono::duration\_cast<std::chrono::duration<**double**>>(t2-t1).count();    **int** nJ= 0;  nJ= tab.gerturnoJ();    string texto= "Mov " + to\_string(nJ) + " t2: " + to\_string(time2) + " 2 COMPUTADORA " + "O (" + to\_string(f) + ", " + to\_string(c) + ")" + "\n";    tab.aumentaJ();  archivo.escribir(texto);    }    archivo1.guardart(tab, "JvsC.txt");    system("cls");      **if**(tab.tablero\_lleno()){  cout<<"\t\t\tHa sido un empate"<<endl;  cout<<"\t\t\t";system("pause");  **break**;  }    }**while**(!tab.Ganador());    l= tab.tablero\_lleno();  ganador= tab.Ganador();    **if**(ganador){  cout<<"\t\t\tEl jugador "<<ganador<<" ha ganado"<<endl;  cout<<"\t\t\t";system("pause");  **break**;  }    **if**(l!= '-'){  **break**;  }  }  }  };  */\*----------------------------------------------------------------------------------------------*  *CLASE COMPUTADOR VS COMPUTADOR*  *-------------------------------------------------------------------------------------------------\*/*  **class** ComputadorvsComputador: **public** Tablero{  private:  Tablero tab;    public:    **void** cargarCvsC(){  **int** conf;    cout<<"Desea cargar una partida anterior Si(1)/No(2)?"<<endl;  cin>>conf;    ArchivoTx archivo1 ("CvsC.txt");    **switch**(conf){  **case** 1:    archivo1.cargart(tab, "CvsC.txt");  system("cls");  **break**;    **case** 2:    **break**;  }    }    **void** iniciarCvsC(){  **int** f, c;  **char** ganador, l, x, o;    ArchivoTx archivo("Jugadas.txt");  ArchivoTx archivo1 ("CvsC.txt");    cargarCvsC();    **while**(true){    **do**{  tab.mostrar();    cout<<"\t\t\tJugador "<<tab.getturno()<<", es su turno"<<endl;  **auto** t1= std::chrono::high\_resolution\_clock::now();    **if**(tab.getturno()=='X'){    cout<<"\t\t\tLa computadora 1 esta pensando.."<<endl;  cout<<"\t\t\t";system("pause");    **do**{    f= rand()%3;  c= rand()%3;    }**while**(!tab.casilla\_vacia(f, c));    tab.hacer\_jugada(f, c);    **auto** t2= std::chrono::high\_resolution\_clock::now();  *// Medir el tiempo que tarda el jugador 1 en hacer la jugada*  **double** time1= std::chrono::duration\_cast<std::chrono::duration<**double**>>(t2-t1).count();  **double** time2= 0.0;    **int** nJ= 0;  nJ= tab.gerturnoJ();      string texto= "Mov " + to\_string(tab.gerturnoJ()) + " t1: " + to\_string(time1) + " 1 COMPUTADORA " + "X (" + to\_string(f) + ", " + to\_string(c) + ")" + "\n";    tab.aumentaJ();    archivo.escribir(texto);    }**else**{    cout<<"\t\t\tLa computadora 2 esta pensando.."<<endl;  **auto** t1= std::chrono::high\_resolution\_clock::now();  cout<<"\t\t\t";system("pause");  **do**{  f= rand()%3;  c= rand()%3;    }**while**(!tab.casilla\_vacia(f, c));    tab.hacer\_jugada(f, c);    **auto** t2= std::chrono::high\_resolution\_clock::now();  **double** time1= 0.0;  *// Medir el tiempo que tarda el jugador 2 en hacer la jugada*  **double** time2= std::chrono::duration\_cast<std::chrono::duration<**double**>>(t2-t1).count();    **int** nJ= 0;  nJ= tab.gerturnoJ();    string texto= "Mov " + to\_string(nJ) + " t2: " + to\_string(time2) + " 2 COMPUTADORA " + "O (" + to\_string(f) + ", " + to\_string(c) + ")" + "\n";    tab.aumentaJ();  archivo.escribir(texto);    }    archivo1.guardart(tab, "CvsC.txt");  system("cls");    **if**(tab.tablero\_lleno()){  cout<<"\t\t\tHa sido un empate"<<endl;  cout<<"\t\t\t";system("pause");  **break**;  }    }**while**(!tab.Ganador());    l= tab.tablero\_lleno();  ganador= tab.Ganador();    **if**(ganador){  cout<<"\t\t\tEl jugador "<<ganador<<" ha ganado"<<endl;  cout<<"\t\t\t";system("pause");  **break**;  }    **if**(l!= '-'){  **break**;  }  }  }  };  */\*----------------------------------------------------------------------------------------------*  *MAIN*  *-------------------------------------------------------------------------------------------------\*/*  **int** main(){  Menu m;  Tablero tab;  JugadorvsJugador t;  JugadorvsComputador b;  ComputadorvsComputador c;      **do**{  m.titulo\_juego();  m.mostrar\_opciones();    system("cls");    **switch**(m.obtener\_opcion()){    **case** 1:  t.iniciarJvsJ();  system("cls");  **break**;  **case** 2:  b.iniciarJvsC();  system("cls");  **break**;  **case** 3:  c.iniciarCvsC();  system("cls");  **break**;  **case** 4:  tab.tutorial();  system ("cls");  **break**;  **case** 5:  cout<<"Hasta luego! \n";  **break**;  default:  system("cls");  **break**;    }    }**while**(m.obtener\_opcion()!= 5);    **return** 0;  } |

**CASO DE PRUEBAS**

En el primer intento de insertar los archivos solamente se colocó en la clase JugadorvsJugador ya que allí se estaba probando, no guardaba todas las posiciones solamente una posición. Prestaba problemas de legibilidad.(Archivo intento1.txt)

En el segundo intento se logró que se entendiera pero faltaba únicamente el tiempo. Seguía con algunos problemas como no guardaba todas las posiciones al realizar. (Archivo intento2.txt)

En proyecto final se resolvió todos los problemas. Guarda todas las posiciones, guarda todos los tiempos, turnos, es decir, los criterios solicitados.