```
string vectorTablero[9] = {};
```
guarda las posiciones de los jugadores en un vector

```
#include <ctime>
```
Delay de la animación del robot

```
#include <cstring>
```
Cambia una variable tipo char a string

```
void eleccion()
```
la función sirve para elegir el modo de juego, dos jugadores o la CPU ; tomando la variable modo de juego

```
void primerMenu()
```
menú principal del juego, que con un ciclo for escoge el modo a jugar dentro del menú tomando una variable booleana llamada booldecision en true y usar un delay para la animación del menú

```
void imprimirTablero()
```
imprime la matriz del triqui principal para usarse en todo el programa siendo una 3x3 ciclos for anidado y con las respectivas separaciones

```
void ceros()
```
la función vuelve a poner a matriz y los vectores vacíos para volver a jugar, algo así como reiniciar la matriz para jugar de nuevo, luego se importa la función para elegir 0 y volver a jugar en todos los modos de juego

```
 contVector = 0;
```
se usa para saber los turnos que se llevan en la partida

```
    int posVector = 0;
```
variable local que sirve para posicionar el vector de 1 en 1

```
 ganador();
        //Usamos este if y esta etiqueta por si el jugador 1 dio con una
combinacion ganadora no le quede el turno al jugador 2
        //ya que podria ocasionar un  bucle o una falla en la victoria
        if (ganador() == 1 || ganador() == 2)
        {

            goto etc;
        }
```
Si ganador==2 quiere decir es un empate y si ganador==1 quiere decir que un jugador ganó

```
 if ((tablero[i][j] == "X" && tablero[i][j + 1] == "X" && tablero[i][j + 2]
== "X") || (tablero[i][j] == "O" && tablero[i][j + 1] == "O" && tablero[i][j
+ 2] == "O"))
            {
                return 1;
            }
```

En la función ganador indica las verticales

```
if ((tablero[j][i] == "X" && tablero[j + 1][i] == "X" && tablero[j + 2][i]
== "X") || (tablero[j][i] == "O" && tablero[j + 1][i] == "O" && tablero[j +
2][i] == "O"))
            {

                return 1;
            }
```

Indica las horizontales

```
if ((vectorTablero[0] == "X" && vectorTablero[4] == "X" && vectorTablero[8]
== "X") || (vectorTablero[0] == "O" && vectorTablero[4] == "O" &&
vectorTablero[8] == "O"))
            {
                return 1;
            }
```

Indica la diagonal primaria

```
if ((vectorTablero[6] == "X" && vectorTablero[4] == "X" && vectorTablero[2]
== "X") || (vectorTablero[6] == "O" && vectorTablero[4] == "O" &&
vectorTablero[2] == "O"))
            {
                return 1;
            }
```

Indica la diagonal secundaria

```
    fstream his("historial.txt", fstream::in | fstream::app);
```

esta línea de código se encarga de no sobre escribir el archivo txt escribiéndolo abajo

```
   his << put_time(&tm, "%d/%m/%Y -- %H:%M:%S") << endl;
    his << "\n";
    his.close();
```

líneas de código encargadas de mostrar la fecha y hora del historial y por último se cierra el archivo txt.

```
void limpiarHistorial() {
    ofstream historia("historial.txt");
}
```

Esta función se encarga de limpiar el historial y reiniciarlo en 0.