# Objetivos Específicos:

1. **Alinear tres fichas**: Un jugador gana si logra colocar tres de sus fichas en fila, ya sea:
   1. Horizontalmente (en una de las tres filas).
   2. Verticalmente (en una de las tres columnas).
   3. Diagonalmente (en una de las dos diagonales).
2. **Bloquear al oponente**: Si el oponente está a punto de ganar, es importante que el jugador bloquee esa línea antes de que se complete.
3. **Empatar**: Si ambos jugadores juegan de manera óptima, el resultado puede ser un empate, donde no hay ganador después de que se llenen todas las posiciones del tablero.
4. **Condiciones de Terminado**: El juego termina cuando hay un ganador o cuando se han realizado 9 turnos sin que haya un ganador, resultando en un empate.

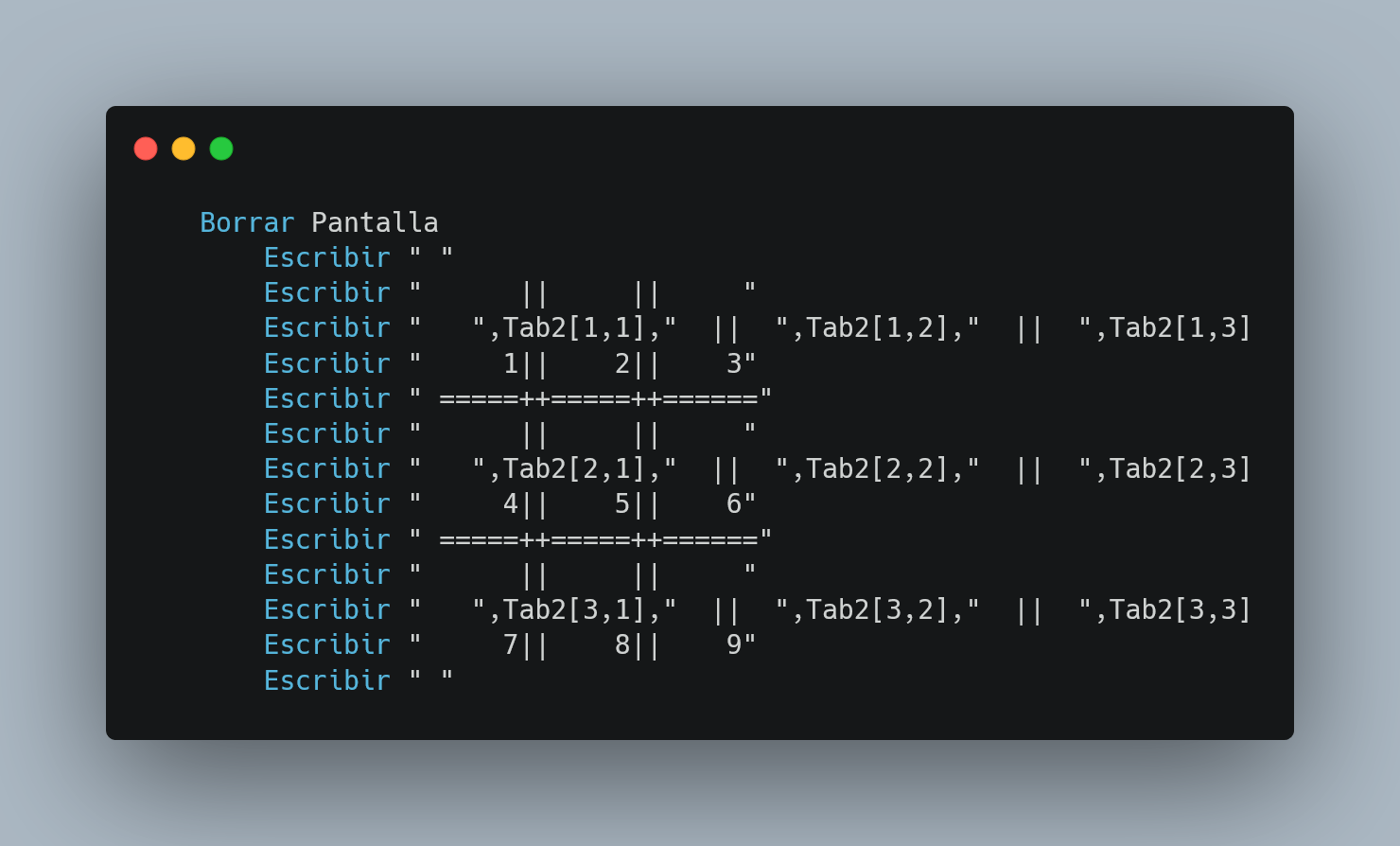
## Variables

1. **Tablero**:
   * Tab1[3][3]: Almacena los valores del juego (0: vacío, 1: 'X', 2: 'O').
   * Tab2[3][3]: Almacena la representación visual del tablero (como cadenas).
2. **Estado del juego**:
   * Terminado: Booleano que indica si el juego ha terminado.
   * HayGanador: Booleano que indica si hay un ganador.
   * CantTurnos: Contador de turnos jugados.
3. **Turno**:
   * Ficha: Carácter que representa la ficha del jugador actual ('X' o 'O').
   * Valor: Valor numérico asociado a la ficha (1 para 'X', 2 para 'O').
4. **Posición**:
   * Pos: Número de la posición donde el jugador desea colocar su ficha.
   * i, j: Índices para acceder a Tab1 y Tab2.

### 

# Interfaz de Usuario

La interfaz de usuario será una consola simple que mostrará el tablero y pedirá entradas al jugador. Aquí hay un esbozo de cómo se verá la interfaz:



**Pasos de Desarrollo**

1. **Planificación del Juego**
   * **Definición de Reglas**: Establecí las reglas básicas del juego, donde dos jugadores (un humano y una CPU) alternan turnos tratando de alinear tres fichas en un tablero de 3x3.
   * **Diseño de la Interfaz**: Opté por una interfaz de consola simple, donde el tablero se muestra visualmente y se solicita al jugador que ingrese su movimiento.
2. **Estructura de Datos**
   * Tablero: Decidí utilizar dos matrices:
   * Tab1: Almacena el estado del juego (0 para vacío, 1 para 'X', 2 para 'O').
   * Tab2: Almacena la representación visual del tablero.
   * Variables de Estado: Incluí variables para manejar el estado del juego, como terminado, hay\_ganador, y cant\_turnos.
3. **Implementación de Funciones**
   * Dibujo del Tablero: Creé una función para mostrar el tablero en la consola.
   * Validación de Movimientos: Implementé una función que verifica si la entrada del jugador es válida.
   * Verificación de Ganador: Desarrollé una función que comprueba si hay un ganador después de cada movimiento.
   * Turno de la CPU: Implementé una lógica para que la CPU elija una posición aleatoria válida.
4. **Bucle Principal del Juego**
   * Implementé un bucle que gestiona el flujo del juego, alternando entre los turnos del jugador humano y la CPU, y verificando el estado del juego tras cada movimiento.
5. **Pruebas y Ajustes**
   * Realicé pruebas manuales del juego para asegurarse de que las reglas funcionaran correctamente y que la lógica de verificación de ganadores fuera precisa. Ajusté el código según fuera necesario para solucionar errores o mejorar la jugabilidad.

#### **Decisiones Tomadas**

* **Elección de la Interfaz de Consola**: Opta por una interfaz de consola por su simplicidad y facilidad de implementación. Esto permitió centrarme en la lógica del juego sin complicaciones adicionales de diseño gráfico.
* **Uso de aleatoriedad para la CPU**: Decidí que la CPU elige posiciones aleatorias en lugar de implementar una lógica más compleja, lo que facilita la programación y mantiene el juego accesible para los jugadores.
* **Validación de Entradas**: Implementé una validación robusta para asegurar que las entradas del usuario fueran correctas, previniendo errores durante la ejecución del juego.

#### **Desafíos Enfrentados**

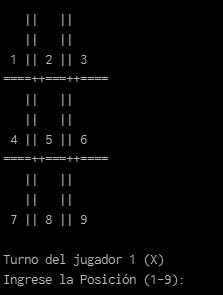
* **Gestión de Turnos**: Asegurar que el turno se alternará correctamente y que el estado del juego se actualizará de manera adecuada fue un desafío. Sin embargo, estableciendo una estructura clara en el bucle principal, logré mantener un flujo lógico.
* **Verificación de Ganadores**: Implementar la lógica para verificar las combinaciones ganadoras fue un poco complicado. Tuve que asegurarme de comprobar todas las filas, columnas y diagonales, lo que requirió una atención cuidadosa a los índices de las matrices.
* **Pruebas de Usabilidad**: Asegurar que la experiencia del usuario fuera fluida requería múltiples pruebas. Tuve que ajustar los mensajes de error y asegurarme de que la interfaz fuera intuitiva.

#### **Conclusión**

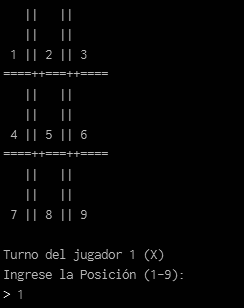
El desarrollo del juego "El gato" fue un proceso educativo que involucró planificación, implementación y prueba. Al final, logré crear un juego funcional que permite a los usuarios disfrutar de una partida clásica. Este proyecto no solo mejoró mis habilidades de programación, sino que también me enseñó la importancia de la planificación y la prueba en el desarrollo de software.

Paso a paso del Programa

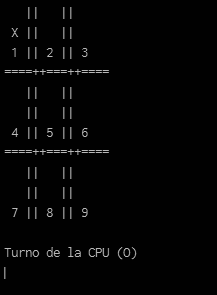
**Visualización del programa**



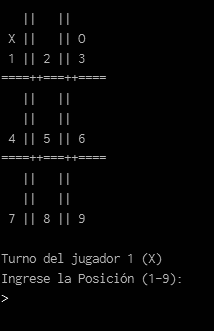
**Ingresar la posición que desee**



**Actualización de pantalla y turno de CPU**

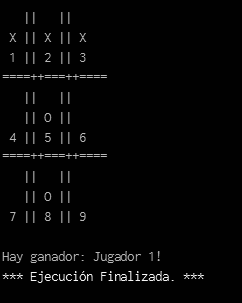


**Actualización de pantalla y turno de jugador 1**

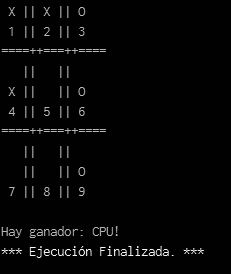


**Se repite hasta que un jugador gane o empate**

**caso 1 : Ganador jugador 1**



**Caso 2 : Ganador CPU**



**Caso 3 : Empate**

