# Desarrollo del juego Tic-Tac-Toe en C++

Farid Leonardo López Gómez y Diego Miranda
Universidad Sergio Arboleda
Bogotá, Colombia

## Resumen

Este artículo presenta el desarrollo del juego clásico *Tic-Tac-Toe* (tres en raya) implementado en el lenguaje C++. Se abordan los conceptos fundamentales aplicados, el diseño modular del código, las funciones principales, los retos de implementación y las conclusiones obtenidas. El proyecto permite la interacción entre dos jugadores, incluye validación de entradas y detección automática de victoria o empate.

## 1. Introducción

El objetivo principal del proyecto es aplicar de forma práctica los conceptos básicos de la programación estructurada mediante la creación del juego *Tic-Tac-Toe*. Este ejercicio fomenta la comprensión del uso de funciones, estructuras de control, arreglos bidimensionales y validación de datos.

El desarrollo se realizó en C++, permitiendo que dos jugadores se enfrenten alternando turnos y registrando sus movimientos en una matriz 3x3. El programa detecta automáticamente condiciones de victoria o empate, garantizando una experiencia fluida y completa.

## 2. Diseño de la solución

### 2.1. Estructura de datos

Se utilizó una matriz bidimensional de caracteres para representar el tablero. Cada posición inicia numerada del 1 al 9 y se reemplaza por el símbolo del jugador al realizar una jugada.

### 2.2. Funciones principales

- **estTabla():** Muestra el tablero actual.
- **ingCasilla():** Solicita la casilla al jugador.
- **verificarError():** Valida que la entrada sea correcta.
- **estadJuego():** Registra la jugada en el tablero.
- **verificarVictoria():** Evalúa si hay un ganador o empate.
- **cambioTurno():** Alterna entre jugadores.
- **reiniciarPartida():** Permite reiniciar la partida sin salir del programa.

## 2.3. Diagrama de flujo

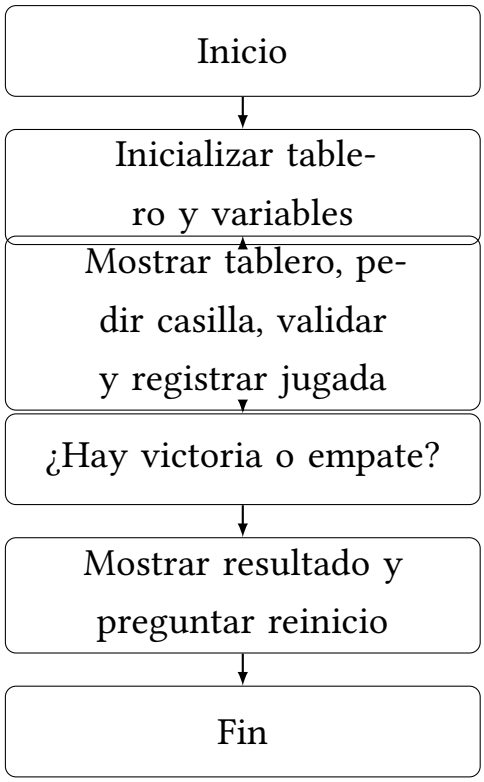


**Figura 1: Diagrama de flujo principal del programa Tic-Tac-Toe (ampliado)**

# 3. Implementación

## 3.1. Desafíos y soluciones

- Validación de entradas: implementada en `verificarError()`.
- Modularidad: funciones separadas por responsabilidades.
- Reinicio de partida: función que restaura las variables iniciales.
- Control de flujo: bucle principal hasta detectar fin del juego.

## 3.2. Decisiones de diseño

1. Uso de un ciclo principal controlado por condiciones de victoria.
2. Parámetros por referencia para modificar valores globales.
3. Separación de funciones para claridad y mantenimiento.

# 4. Conclusiones

El desarrollo del proyecto permitió reforzar el entendimiento de estructuras de datos, validaciones y modularidad en C++. El resultado fue un programa estable, legible y extensible. Como mejora futura, se sugiere incorporar una interfaz gráfica o modo IA para ampliar la funcionalidad.