Universidad Rafael Landívar
Facultad de Ingeniería.
Licenciatura en Informática y Sistemas.
Introducción a la programación (informática)
Catedrático: Ing. Rene Daniel Mejía Alvarado

# PROYECTO PRÁCTICO NO.02

# “CONECTA 4”

Estudiante: Oscar Daniel Xiquin Cumes
Carné: 1118423

Guatemala, 1 de marzo de 2023

# I. INTRODUCCIÓN

En este proyecto de programación que consiste en la creación de un programa que permita jugar Conecta Cuatro, un juego para dos personas en el cual se deben insertar fichas en una cuadrícula de 7 columnas y 6 filas con el objetivo de conectar cuatro fichas del mismo color de manera vertical, horizontal o diagonal. El programa debe permitir elegir entre jugar contra otra persona o contra la computadora, almacenar los nombres de los jugadores, validar que ningún jugador se llame "COMPUTADORA" en caso de jugar contra otra persona y mostrar la secuencia de fichas que le ayudaron a ganar el juego al jugador, además debe permitir iniciar una nueva partida o acceder a otras opciones disponibles en el programa, como la opción de mostrar los últimos 10 jugadores que ganaron una partida, con su tiempo y cantidad de turnos. Si se juega contra la computadora, ésta podrá insertar fichas aleatoriamente.

# II. ANALISIS

## 1. ENTRADAS

- Pregunta al usuario si jugarán dos personas o una persona contra la computadora.
- Pregunta y almacena en memoria el nombre de cada persona.
- El juego comienza en el turno del jugador 1, luego el jugador 2 (que podría ser la computadora) y así sucesivamente hasta que gane uno de los dos jugadores o hasta que ninguno de los dos jugadores tenga posibilidades de ganar.
- Si uno de los dos jugadores gana, el programa debe mostrar la secuencia de fichas que le ayudaron a ganar el juego.
- Al terminar el juego, el programa no debe cerrarse a menos que el usuario lo indique. En su lugar, deberá preguntarse al usuario si desea iniciar una nueva partida (contra la computadora u otra persona) o cualquiera de las opciones disponibles del programa.
- El programa debe tener una opción para mostrar los últimos 10 jugadores en ganar una partida.
- Si una partida termina en empate, la información de esa partida no debe almacenarse.

## 2. SALIDAS

- La bienvenida
- Imprimir en pantalla al ganador del juego.
- Imprimir en pantalla el tablero asi como las fichas que ingresa el usuario.
- Recomendaciones al usuario para que elija una opcion del menu la cual le parezca al usuario.
- Imprimir el tiempo, las partidas, turnos etc.

## 3. PROCESOS

1. Solicitar al usuario la modalidad de juego, ya sea dos personas o una persona contra la computadora, y almacenar esta información en memoria.

2. Si se escogió jugar contra la computadora, almacenar en memoria el nombre del primer jugador y establecer el nombre del segundo jugador como "COMPUTADORA". Si se escogió jugar con dos jugadores humanos, solicitar y almacenar en memoria los nombres de ambos jugadores, validando que ninguno de ellos se llame "COMPUTADORA"

3. Iniciar el juego en el turno del jugador 1, solicitando y validando la columna en la que se desea insertar una ficha. Continuar alternando los turnos de los jugadores hasta que uno de ellos gane o hasta que no haya posibilidad de ganar. Si uno de los jugadores gana, mostrar la secuencia de fichas que le ayudaron a ganar y preguntar al usuario si desea iniciar una nueva partida o salir del programa. Si no hay posibilidad de ganar, mostrar un mensaje indicando el empate y preguntar al usuario si desea iniciar una nueva partida o salir del programa.

## 4. RESTRICCIONES

**1.** Validar que el nombre de los jugadores no tenga caracteres especiales o números, ya que esto podría generar errores en la ejecución del programa.

**2.** Limitar el número máximo de fichas que se pueden insertar en cada columna, para evitar que el juego se vuelva interminable o que el programa se buguee.

**3.** Asegurarse de que el programa tenga una buena gestión de errores y excepciones, para que el usuario pueda entender qué salió mal y cómo solucionarlo en caso de que ocurra algún error inesperado.

# III. DISEÑO

**Diagrama de flujo 1**

*INICIO DEL JUEGO*

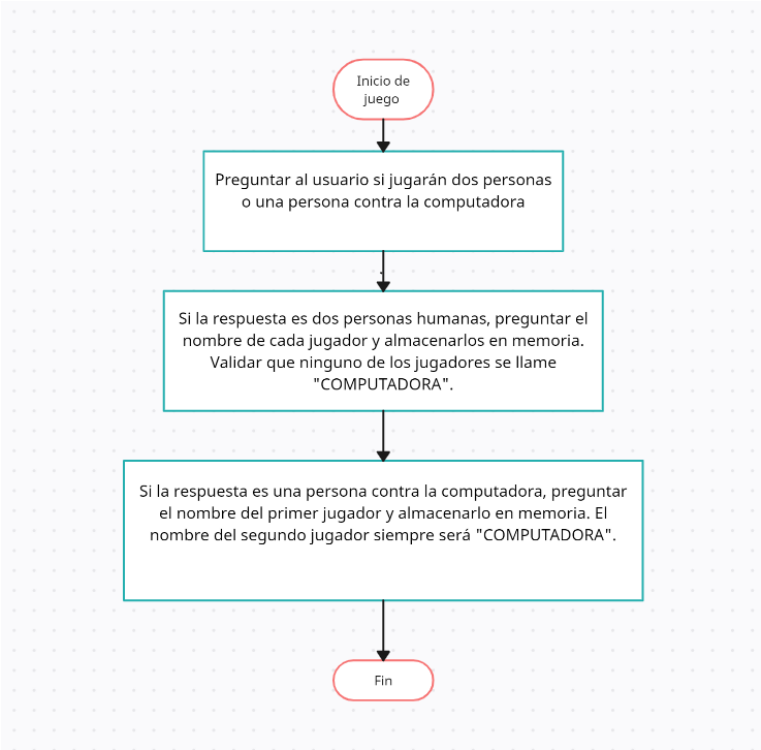


**Diagrama de flujo 2**

*ELABORACIRACION Y EJECUCIÓN DE JUEGO*

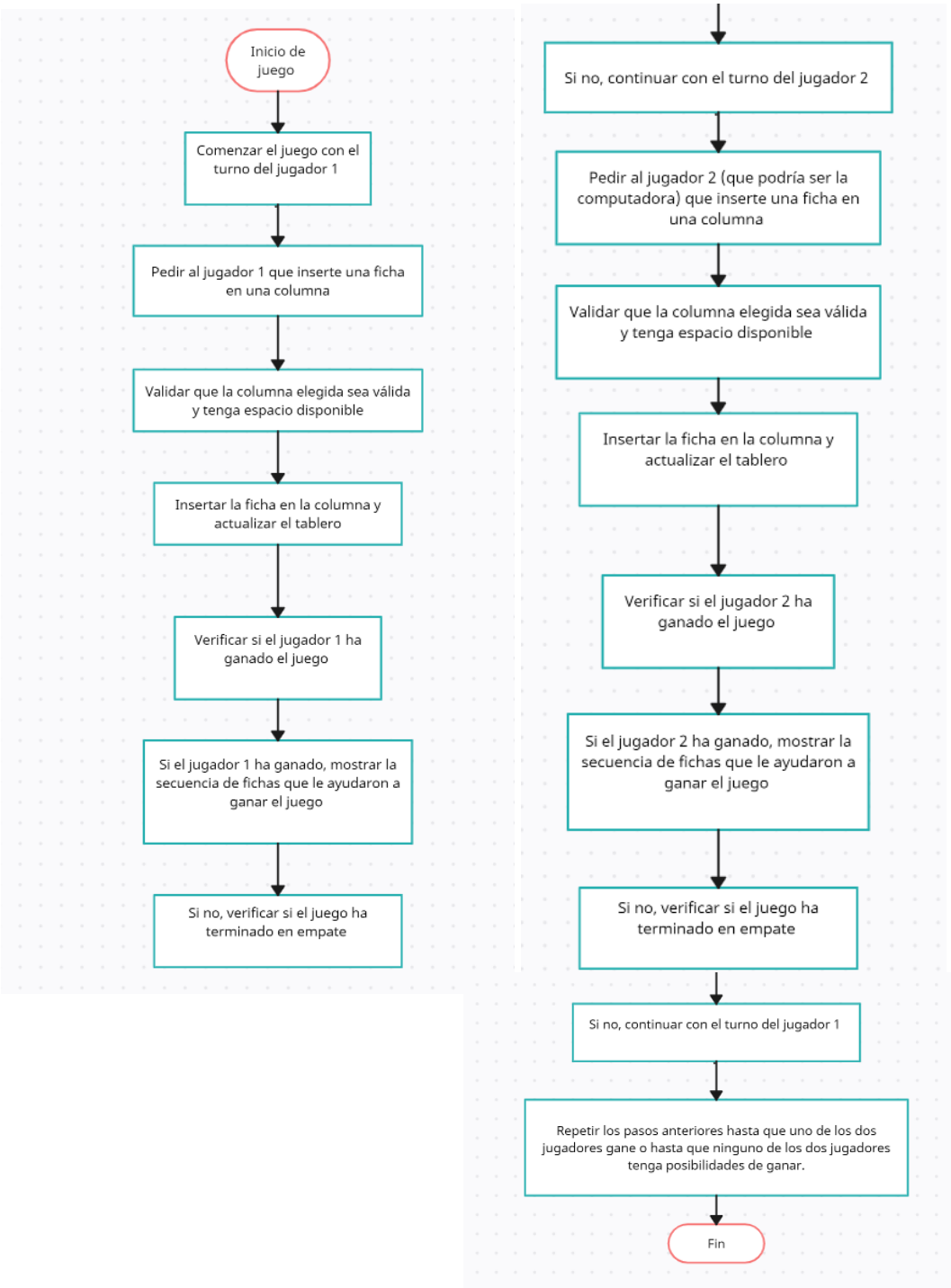
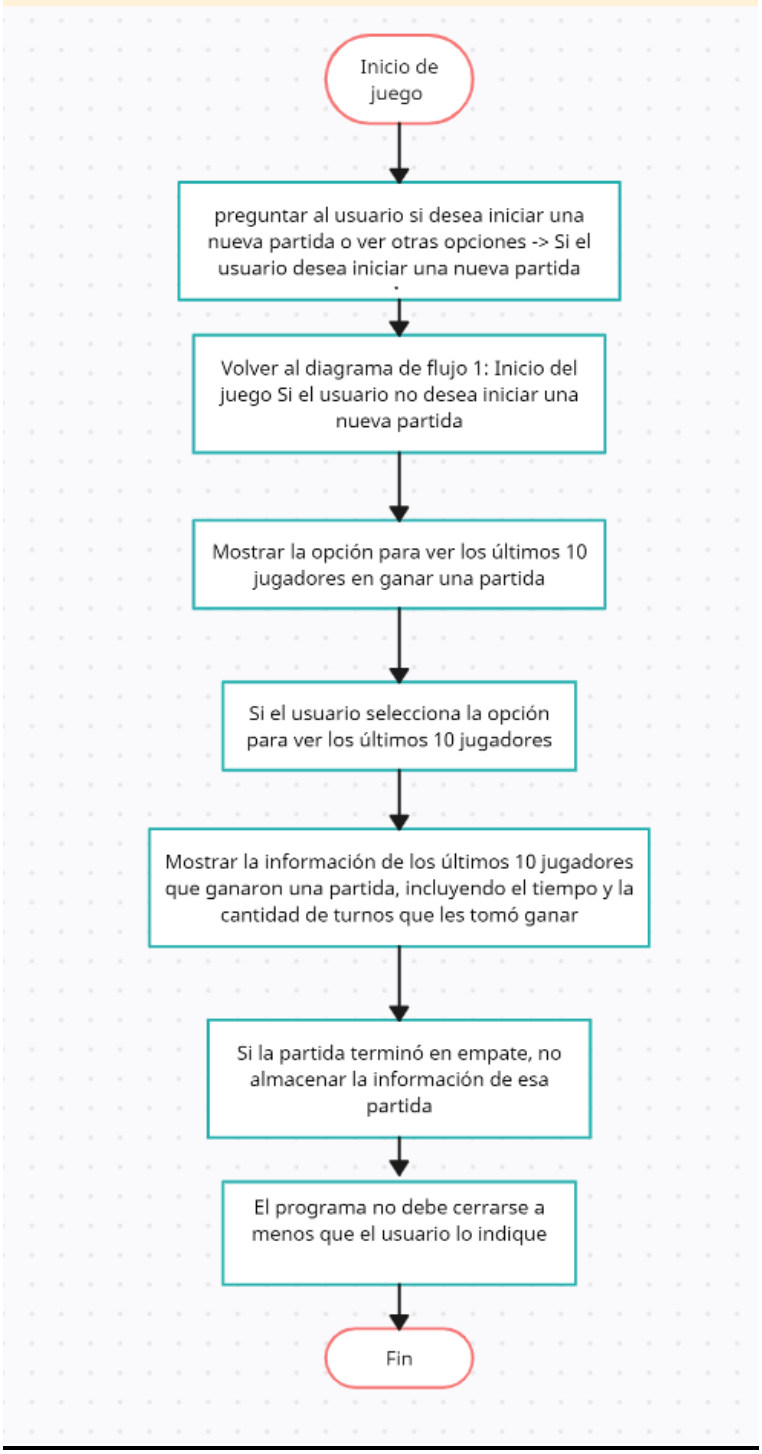


**Diagrama de flujo 3**

*FINALIZACIÓN DE JUEGO*



# IV. CONCLUSIONES

1. El programa tenga la capacidad de preguntar y almacenar en memoria los nombres de los jugadores, y que se validen los nombres para evitar confusiones en caso de que uno de los jugadores sea la computadora y así realizar un buen trabajo.

2. El programa debe asegurarse de que el juego se desarrolle correctamente, comenzando por el turno del jugador 1 y asegurándose de que ningún jugador se salte un turno. Además, el programa debe mostrar la secuencia de fichas que le ayudaron al ganador a obtener la victoria.

3. Es importante que el programa tenga opciones para que el usuario pueda continuar jugando o elegir otra opción, como ver los últimos 10 jugadores en ganar una partida. Además, el programa debe asegurarse de que la información de partidas que terminen en empate no se almacene.

# 4. RECOMENDACIONES

1. Usar una estructura de datos adecuada para representar el tablero del juego, por ejemplo, una matriz o una lista de listas. Esto hará que sea más fácil acceder a los elementos del tablero y actualizarlo después de cada turno.

2. Agrega comentarios y documentación para explicar qué hace cada parte del programa y cómo funciona. También puedes agregar documentación para el usuario final, explicando cómo jugar el juego y qué opciones están disponibles.

3. Valida las entradas del usuario, asegúrate de validar todas las entradas del usuario para prevenir errores y problemas en el juego. Por ejemplo, si el usuario ingresa un nombre inválido, muestra un mensaje de error y solicita que lo ingresen nuevamente.

4. Considera agregar gráficos para hacer el juego más atractivo visualmente, considera agregar gráficos y animaciones para hacer que el juego sea más interactivo y atractivo.

5. Usa una biblioteca de tiempo, para medir el tiempo y mostrarlo al usuario, esto hará que el código sea más sencillo y preciso al momento de medir el tiempo de juego.

6. Usa un mecanismo de almacenamiento de datos adecuado para almacenar la información de los últimos 10 jugadores en ganar una partida, considera utilizar una base de datos o algún otro mecanismo de almacenamiento de datos adecuado.

# 5. REFERENCIAS

- C#

El lenguaje por utilizar para la realización del código pedido por parte del supermercado PublicMart es el lenguaje C# el cual es un lenguaje de programación de propósito general, desarrollado por Microsoft como parte de la plataforma .NET. Es un lenguaje orientado a objetos y está diseñado para ser seguro, fácil de leer y escribir, y escalable para proyectos grandes. C# se utiliza comúnmente para el desarrollo de aplicaciones de escritorio, aplicaciones web, juegos, aplicaciones móviles y otros tipos de software.

- `using System;`

Hace uso de los namespaces que se encuentran dentro de la biblioteca de clases del .NET Framework llamada "System". Estos namespaces contienen una gran cantidad de clases, estructuras y métodos que son de utilidad en la programación en C#. Esto nos evita tener que escribir el nombre completo del namespace cada vez que se utiliza una clase o método que pertenece a él, lo que hace que el código sea más limpio y legible.

- `using System.Drawing;`

es una directiva de namespace en C# que permite acceder a clases y métodos de la biblioteca "System.Drawing", la cual proporciona funcionalidades para trabajar con gráficos, dibujar imágenes en la pantalla e imprimir en una impresora, y manipular imágenes en formatos como BMP, JPEG y PNG. Al agregar esta directiva al código fuente, se puede acceder a todas las clases y métodos de la biblioteca sin tener que escribir el nombre completo del espacio de nombres cada vez que se hace referencia a una clase o método dentro de él.

- `using System.Threading;`

Esta importa el espacio de nombres "System.Threading", que es parte de la biblioteca de clases de .NET en C#, espacio de nombres proporciona clases y métodos para trabajar con subprocesos y multitarea en aplicaciones C#.

- `using System.Diagnostics;`

"using System.Diagnostics;" es una directiva de namespace en C# que se utiliza para agregar el espacio de nombres "System.Diagnostics" al código fuente. Este espacio de nombres proporciona clases y métodos para trabajar con la depuración, el seguimiento y el monitoreo del rendimiento de las aplicaciones.

## 6. ANEXOS

NO APLICA.