

# Informe Laboratorio N º1 Paradigma Funcional

Nombre: Ian Rioseco Castro

Rut: 21.249.591-2

Profesor: Edmundo Leiva

## 2.Indice

Contenido

[............................................................................................................................................................. 1](#_bookmark0)

[Informe Laboratorio N º1 Paradigma Funcional 1](#_bookmark1)

1. [Índice 2](#_bookmark2)
2. [Introducción 3](#_bookmark3)

[3,1. Descripción del problema: 3](#_bookmark4)

[3,2. Descripción del paradigma y relación con el laboratorio 3](#_bookmark5)

1. [Análisis del problema 4](#_bookmark6)

[4,1. TDAs obligatorios: 4](#_bookmark7)

1. [Diseño de la solución 5](#_bookmark8)

[5,1. Idea principal: 5](#_bookmark9)

[5,2. Funciones Auxiliares 5](#_bookmark10)

1. [Aspectos de implementación 6](#_bookmark11)
2. [Instrucciones de uso 6](#_bookmark12)
3. [Resultados y Auto Evaluación 7](#_bookmark13)
4. [Conclusión 7](#_bookmark14)
5. [Anexos 8](#_bookmark15)
6. [Bibliografía 11](#_bookmark16)

3,1. Descripción del problema:

## Introducción

En este laboratorio, abordaremos el lenguaje de programación scheme el cual aplicaremos en el desarrollo de un juego llamado conecta 4 (figura 1), conecta 4 es un juego para 2 jugadores que se juega en un tablero vertical de 6 filas y 7 columnas. Cada jugador cuenta con 21 fichas el cual sería un juego normal y 10 fichas si es un juego rápido, cada jugador tiene un color asociado el cual puede ser “rojo” o “amarillo”. Los jugadores van por turno dejando caer sus fichas desde la parte superior del tablero en alguna de las 7 columnas de este. Las fichas se van posicionando en la parte baja del tablero, con el objetivo de ser el primero en formar una línea de 4 fichas del mismo color, ya se vertical, horizontal o diagonal (figura 2). Si esto no sucede ya sea por que llenaron el tablero o ambos se quedaron sin fichas se considera empate. Los jugadores se pueden ir bloqueando mutuamente para así no completar la línea de 4 fichas, estas no se pueden sobreponer a otra ficha ya posicionada.

El objetivo de este laboratorio es construir un conecta 4, permitiendo a los jugadores realizar las mismas operaciones que pueden realizar en el juego físico.

3,2. Descripción del paradigma y relación con el laboratorio:

Para este proyecto, se utilizará el lenguaje de programación scheme mediante DrRacket, este es un lenguaje funcional. El paradigma funcional se caracteriza por el uso de funciones, ya que no se realiza el uso tradicional de variables. En lugar de modificar datos ya existentes, se crean nuevos datos mediante el uso de funciones y para esto es muy esencial la recursión, las funciones que se crean se utilizan para manipular listas ya sea con recursión natural, de cola o simplemente sin recursión.

El enfoque de este paradigma basado en funciones y listas funciona en la creación de un conecta 4 ya que el tablero puede ser visto como una serie de listas de listas, además con la utilización de scheme el cual es un lenguaje funcional se tienen las herramientas necesarias para la construcción de este juego, empleando recursión y las funciones como base del proyecto.

## Análisis del problema

Para resolver la problemática de este proyecto, se requerirá la manipulación de listas de listas, creación de funciones, constante creación de nuevos datos basados en uno anterior y el uso de recursión natural o de cola, esto nos ayudara a poder simular cada una de las operaciones que se pueden realizar en el conecta 4 físico, y las problemáticas que se puedan dar mientras se juega.

La creación de TDAs nos ayudara a mantener un orden y claridad en el proyecto, cada uno de estos tiene una función constructora, selectora y modificadora, esto con el fin de construir una estructura fácil de manipular y realizar las interacciones entre funciones de TDAs distintos, con el fin de tener una buena organización de la solución al problema.

4,1. TDAs obligatorios:

Este laboratorio consta con 4 TDAs principales requeridos para el correcto funcionamiento del conecta 4, los cuales se usarán para la creación de las demás funciones necesarias para resolver el problema.

* TDA Player: constructor que consta de, ID del jugador, nombre, color de la ficha, numero de victorias, numero de derrotas, numero de empates, cantidad de piezas actuales. Se crea un perfil para cada jugador.
* TDA Piece: constructor el cual solo necesita el color que usaremos para crear una pieza.
* TDA Board: constructor, para crear el tablero necesitamos que sea 7 columnas X 6 filas, donde cada índice dentro de la fila es una columna.
* TDA Game: constructor que consta de, jugador 1, jugador 2, tablero, turno-actual, además y este se utilizara para guardar el historial de juego.

5,1. Idea principal:

## Diseño de la solución

Para abordar el desarrollo del juego de Conecta 4, se proponen cuatro Tipos de Datos Abstractos (TDAs) principales que organizan y gestionan los elementos del juego, implementando estructuras claras con selectores y modificadores.

1. **TDA Jugador**: Define el perfil del jugador mediante una lista que almacena datos como ID, nombre, color, victorias, derrotas, empates y piezas. Incluye selectores para cada dato y modificadores para actualizar victorias, derrotas, empates y piezas restantes, permitiendo una interacción eficiente con otras funciones del juego.
2. **TDA Pieza**: Contiene un constructor y un selector, enfocados en crear y gestionar las piezas utilizadas en el tablero.
3. **TDA Tablero**: Crea el tablero 7x6 mediante funciones para generar columnas y filas. Incluye funciones para verificar estado del tablero, verificar jugadas, colocar piezas en la posición más baja de una columna, y verificar victorias en direcciones vertical, horizontal y diagonal, empleando recursión natural para evaluar cuatro piezas consecutivas del mismo color.
4. **TDA Juego**: Gestiona el estado del juego incluyendo el perfil de los dos jugadores, el tablero, el turno actual y un historial de juego. Incorpora funciones para generar y almacenar el historial, verificar empates, actualizar estadísticas de los jugadores, determinar el jugador actual y finalizar el juego. También se define una función de movimiento que valida el turno, actualiza el tablero y el historial, y verifica la victoria o empate, usando una bandera para evitar doble actualización de estadísticas al finalizar la partida.

Cada TDA se documentará adecuadamente para facilitar su comprensión y manipulación en el código del juego

5,2. Funciones Auxiliares:

A continuación, describimos funciones originales del código y decisiones que no estaban explicitadas en el enunciado. Se creo una función que crea una lista con los colores rojo y amarillo donde la primera posición es el jugador 1 y la segundo el jugador 2, esta lista la usaremos en las funciones de victorias para verificar el color ganador ya que así sabremos cual es el jugador ganador.

## Aspectos de implementación

El proyecto exige 8 archivos obligatorios: uno para cada TDA, 3 para los scripts de prueba los cuales deben considerar todas las situaciones del juego conecta 4 y uno para el Main el cual contendrá la función externa creada. Todos estos archivos fueron implementados en “DrRacket” versión 8.12. Por otra parte, ¡se prohíbe el uso de funciones que implican manipulación directa de estado, como set! y similares, para evitar la simulación de variables tradicionales. Además, no se permite el uso de bibliotecas externas, solamente se permite el uso de la biblioteca natural de Racket.

## 7.Instrucciones de uso

Lo más importante es utilizar correctamente los dominios de las funciones, ya que estas poseen un dominio definido el cual debe respetarse para garantizar el correcto funcionamiento del código, si este no se respeta se caerá el Código. Usa únicamente el archivo de pruebas y todos los archivos con extensión “. rkt” los cuales se encuentran en la carpeta de códigos; no edites otros archivos. Tener los 8 archivos abiertos al momento de ejecutar algunos de los scripts (figura 3) y todos en la misma carpeta (figura 4), No utilices funciones que no hayan sido implementadas. Aunque algunas funciones son necesarias, otras podrían no estar desarrolladas, así que evita utilizarlas para evitar errores o problemas inesperados, por otro lado, leer obligatoriamente el archivo “léeme.txt”, ya que este contiene instrucciones importantes para la ejecución del programa. Para generar correctamente una partida de conecta4.

## 8.Resultados y Auto Evaluación

Se lograron 18 de 20 requerimientos funcionales ya que no se realizaron los requerimientos para la inteligencia artificial. Que no se lograran los requerimientos de la IA es netamente por tiempo y además de que no se aprendió correctamente el uso de Gemini. El uso de recurso natural en las funciones de verificación de victoria fue verdaderamente un desafió ya que se debían tomar en cuenta musas condiciones, por lo que estas funciones no funcionan el 100% de las veces que se usó, por otro lado, la función para realizar un movimiento por jugador debido a la cantidad de condiciones tampoco funciono el 100% de las veces que se usó (figura 5).

## 9.Conclusión

En este laboratorio se diseñó e implementó el juego Conecta 4 utilizando el lenguaje de programación Scheme en DrRacket, explorando el paradigma funcional. Este enfoque se basó en la creación y manipulación de listas de listas, el uso de recursión y la construcción de Tipos de Datos Abstractos (TDAs) que representan elementos clave del juego, como el tablero, las fichas y los jugadores. Los TDAs permiten una estructura modular y manejable, facilitando la implementación de funciones de verificación de victorias, manejo de turnos y actualización de estadísticas.

Aunque el proyecto alcanzó la mayoría de los requerimientos, la falta de una IA funcional y ciertas limitaciones en la función de verificación de victorias sugieren áreas de mejora. Este desarrollo no solo permitió replicar las reglas y mecánicas de Conecta 4, sino también ejercitar principios fundamentales del paradigma funcional y los beneficios de la organización modular en la programación estructurada. La experiencia adquirida en el manejo de recursión y estructuración de TDAs será valiosa para proyectos futuros en lenguajes de programación funcional.

## 10.Anexos

Figura 1

Un conjunto de letras blancas en fondo azul

Descripción generada automáticamente con confianza media

**Gráfico, Gráfico de burbujas

Descripción generada automáticamente**

Figura 2

 Figura 3

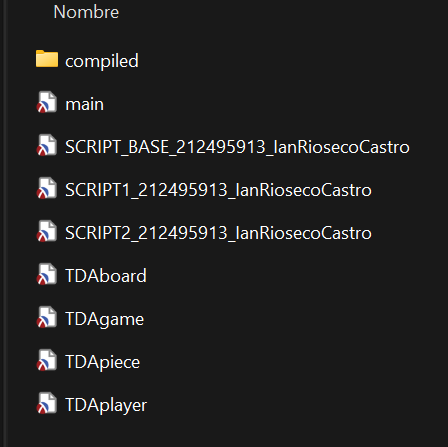


Figura 4



Figura 5

Texto

Descripción generada automáticamente

## 11.Bibliografía:

* 1. Matthew Flatt, M. (s. f.). 4.10 pairs and lists. The Racket Reference. Recuperado 9 de octubre de 2023, de <https://docs.racket-lang.org/reference/pairs.html>
  2. Matthew Flatt, M. (s. f.). 3.9 Local Binding: let. The Racket Reference. Recuperado 9 de octubre de 2023, de [https://docs.racket-](https://docs.racket-lang.org/reference/let.html#%28form._%28%28lib._racket%2Fprivate%2Fletstx-scheme..rkt%29._let%29%29)

[lang.org/reference/let.html#%28form.\_%28%28lib.\_racket%2Fprivate%2Fletstx-](https://docs.racket-lang.org/reference/let.html#%28form._%28%28lib._racket%2Fprivate%2Fletstx-scheme..rkt%29._let%29%29) [scheme..rkt%29.\_let%29%29](https://docs.racket-lang.org/reference/let.html#%28form._%28%28lib._racket%2Fprivate%2Fletstx-scheme..rkt%29._let%29%29)