Informe Laboratorio Nº1 Paradigma Funcional

Nombre: Ian Rioseco Castro

Rut: 21.249.591-2

Profesor: Edmundo Leiva

2.Indice

Contenido

..... 1
Informe Laboratorio N º1 Paradigma Funcional 1
2. Índice 2
3. Introducción 3
3,1. Descripción del problema: 3
3,2. Descripción del paradigma y relación con el laboratorio 3
4. Análisis del problema 4
4,1. TDAs obligatorios: 4
5. Diseño de la solución 5
5,1. Idea principal: 5
5,2. Funciones Auxiliares 5
6. Aspectos de implementación 6
7. Instrucciones de uso 6
8. Resultados y Auto Evaluación 7
9. Conclusión 7
10. Anexos 8
11. Bibliografía 11

3. Introducción

3,1. Descripción del problema:

En este laboratorio, desarrollaremos el juego **Conecta 4** utilizando el lenguaje de programación **Scheme** en **DrRacket**, aplicando el paradigma funcional. Conecta 4 es un juego para dos jugadores que se juega en un tablero vertical de 6 filas y 7 columnas. Cada jugador tiene un color asociado, "rojo" o "amarillo", y dispone de 21 fichas para una partida normal o 10 fichas en una versión rápida del juego. Los jugadores alternan turnos para dejar caer sus fichas en las columnas del tablero, buscando formar una línea de cuatro fichas consecutivas del mismo color en sentido vertical, horizontal o diagonal. Si el tablero se llena o ambos jugadores se quedan sin fichas sin formar una línea de cuatro, el juego termina en empate. Los jugadores también pueden bloquearse mutuamente para evitar que el oponente complete una línea de cuatro.

El objetivo es construir un juego de Conecta 4 que permita a los jugadores realizar todas las acciones posibles en el juego físico, tales como colocar fichas, verificar victorias o empates, y gestionar turnos y estadísticas.

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

Scheme es un lenguaje de programación funcional, lo que significa que no se modifica el estado de variables, sino que se generan nuevos datos a partir de funciones. En lugar de manipular directamente el estado de variables, se construyen nuevas listas y estructuras de datos, empleando frecuentemente recursión natural o de cola. Este paradigma se ajusta a la creación de un Conecta 4 ya que el tablero puede representarse como una serie de listas anidadas. Con la potencia de Scheme en la manipulación de listas y la creación de funciones recursivas, se facilita la construcción de este juego de manera funcional, manteniendo claridad y estructura en el código.

4. 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. Diseño de la solución

5,1. Idea principal:

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 Player**: Representa el perfil del jugador. Almacena información como ID, nombre, color de la ficha, número de victorias, derrotas, empates y piezas restantes. Contiene selectores para cada dato y modificadores para actualizar estadísticas y fichas restantes de manera eficiente, permitiendo que otras funciones interactúen fácilmente con los datos del jugador.
2. **TDA Piece**: Gestiona la creación y manipulación de las fichas que usarán los jugadores en el tablero, conteniendo el color de cada ficha para mantener una clara diferenciación entre las fichas de ambos jugadores.
3. **TDA Board**: Genera el tablero 7x6, organizando las columnas y filas necesarias. Incluye funciones para verificar el estado del tablero, gestionar jugadas, colocar fichas en la posición más baja de cada columna y evaluar condiciones de victoria. Estas verificaciones se hacen en las direcciones vertical, horizontal y diagonal, empleando recursión natural para identificar cuatro fichas consecutivas del mismo color y así determinar el ganador.
4. **TDA Game**: Controla el estado general del juego. Almacena los perfiles de ambos jugadores, el tablero actual, el turno en curso y un historial de la partida. Ofrece funciones para actualizar y almacenar el historial, verificar empates, actualizar estadísticas de jugadores, determinar el jugador actual y finalizar el juego. También cuenta con una función para manejar los turnos de los jugadores la cual, valida el turno, actualiza el tablero e historial y verifica condiciones de victoria o empate.

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

5,2. Funciones externas:

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.

6.Aspectos de implementación

El proyecto consta de ocho archivos esenciales: uno para cada TDA, tres para scripts de prueba y uno para el archivo principal, Main, que contiene la función externa. Los scripts de prueba cubren distintos casos posibles del juego, lo que garantiza el correcto funcionamiento de las funciones principales. La implementación fue realizada en DrRacket (versión 8.12), y se evitó el uso de funciones que manipulan directamente el estado, como set!, para respetar el paradigma funcional. Todos los archivos .rkt deben estar abiertos y ubicados en la misma carpeta para asegurar una correcta ejecución de las pruebas y funciones principales.

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 editar 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, 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

En el proyecto se lograron cumplir 18 de los 20 requisitos funcionales. Sin embargo, no se implementó la inteligencia artificial (IA) para jugar contra la computadora, debido a limitaciones de tiempo y la falta de conocimientos en la herramienta Gemini. La creación de las funciones para verificar victorias utilizando recursión fue particularmente desafiante, debido a las diversas condiciones a considerar, lo que impactó su fiabilidad en todos los casos posibles. Adicionalmente, la función de movimiento para los turnos presentó problemas debido a la cantidad de condiciones que debe evaluar.

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

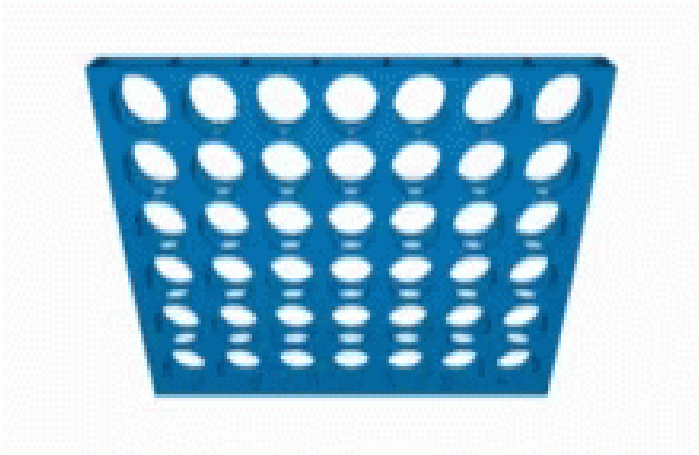


Figura 2

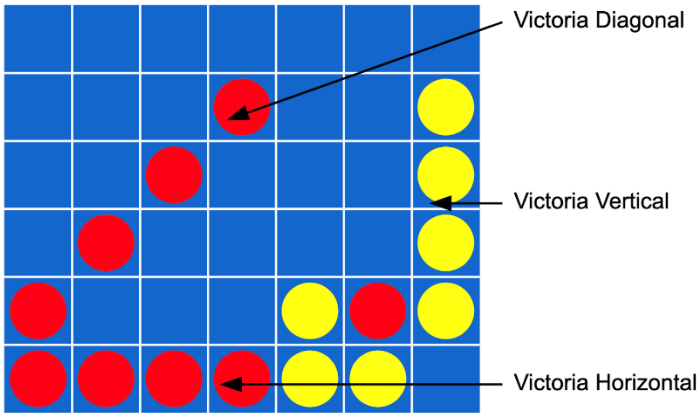
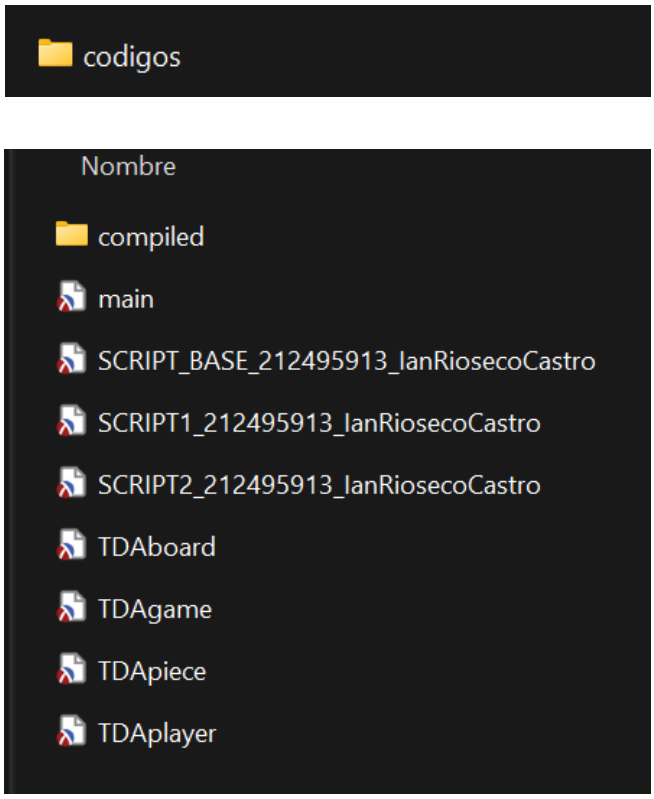


Figura 3

Figura 4



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-lang.org/reference/let.html#%28form._%28%28lib._racket%2Fprivate%2Fletstx-scheme..rkt%29._let%29%29