



LABORATORIO 1 – PARADIGMA FUNCIONAL

Paradigmas de programación

Ignacio Andrés Tapia Donaire 21.142.512-1

Sección:

13204-0-A-1

Profesor:

Edmundo Leiva Lobos

Fecha Entrega:

05 de abril del 2025

Contenido

Introducción:	3
Descripción del problema:	3
Paradigma:	3
Análisis del problema:	4
Diseño de solución:	5
Aspectos de implementación:	6
Instrucciones de uso:	6
Resultados y autoevaluación:	6
Conclusiones:	7
Referencias:	7
Anexos:	7

Introducción:

El DIINF de la Usach ha solicitado la construcción de un juego llamado **CAPITALIA**, juego de similares características a **Monopoly** donde su particularidad radica en el uso de impuestos y el cambio de reglas que pueden modificar el cómo se juega a este título para hacer de este mismo una experiencia más interesante y única. El encargado de llevar a cabo esta idea es el presente en el informe, donde se debe de utilizar el paradigma especificado para construir el juego en su totalidad.

Descripción del problema:

El problema principal radica en la creación y actualización del juego, donde la primera instancia cuesta debido a la naturaleza de cambiar de paradigma y encontrarnos en un entorno poco familiar y completamente nuevo, y el segundo ya que no solo nos enfrentamos a algo completamente nuevo, sino que además su manejo es distinto y por tanto el hacer algo tan simple como “cambiar” las cosas no es un mero cambio de variable, sino crear instancias nuevas con lo ya creado y junto al cambio, esto debido principalmente a la naturaleza propia del **paradigma funcional** el cual será discutido en el punto siguiente.

Paradigma:

El tipo de paradigma utilizado para este laboratorio particular es el **funcional**, el cual radica en el uso de funciones y listas para poder manejar todo tipo de variable que se pueda llegar a utilizar. La parte más importante por tomar en cuenta de este paradigma es si es el hecho de la **no mutabilidad**, ósea, no se puede llegar y cambiar una variable como se antoje, sino que se debe hacer de manera más meticulosa. Para ello disponemos de herramientas tanto nuevas como ya conocidas, para el caso particular la aplicación principal consta en lo siguiente:

- **Recursión:** De manera simple, es llamar a la función dentro de si misma para poder resolver el problema, dentro de esta tenemos un par de casos:

- **Recursión Natural:** Se llama a la función al final de esta, creando así la recursividad, **no hace operaciones de ningún tipo, sino que llama a las variables ocupadas dentro de si misma para volverlas a ocupar.**
- **Recursión de Cola:** Llamamos la función al final de esta, **en este caso hacemos también la operación necesaria para resolver el problema.**

Es importante denotar que ambas requieren de un **caso base** para poder finalizar, en caso contrario podríamos encontrarnos con un loop infinito.

- **Funciones de orden superior:** Funciones las cuales reciben a otras de parámetros y devuelven una función, extremadamente útiles y necesarias para la resolución general de problemas en el paradigma funcional.

Análisis del problema:

Para resolver el problema que tenemos en frente debemos de analizar detenidamente los puntos necesarios para poder hacerlo, dicho esto necesitamos:

- Constructores para el **juego, propiedades, jugadores, cartas y tablero.**
- Selectores para poder seleccionar y manejar las distintas “variables” que puedan tener cada constructor, **no necesariamente cada dato del constructor requiere de un selector ya que puede no ser utilizado.**
- Modificadores para poder, valga la redundancia, “modificar” los constructores a medida que avanza el juego, se debe denotar que el termino modificar no se refiere a literalmente cambiar la estructura, sino el **cambiar los valores acordes a lo que sucede dentro del juego, creando una copia del original para no mutarla/modificarla** ya que esto seria en contra del paradigma **funcional** al que estamos ligados.
- Otros TDA aparte para poder hacer manejo de variables que no necesariamente se encuentran en el constructor, sino mas bien ayudan al manejo de este o funcionamiento en su totalidad.

Con estos 4 puntos en mente, debemos enfocarnos en el tipo de juego que se va a crear, este es una especie de **Monopoly** por lo cual es importante considerar cosas como **la cárcel, dados dobles, pagos y cobros, construcciones, hipotecas, bancarrota y turnos**, sin contar que para el caso particular también debemos tener en cuenta **impuestos**, por lo cual los selectores y modificadores juegan un papel extremadamente importante a la hora de como se va a resolver el laboratorio.

Diseño de solución:

En primera instancia y siguiendo lo anteriormente visto, primero debemos construir cada TDA para poder armar el juego, por lo cual se crean los **TDA jugador, TDA propiedad, TDA carta, TDA tablero y TDA juego. (Anexo A)**

Hecho esto, se crean selectores para cada variable dentro de los constructores anteriormente **sola y exclusivamente cuando sean necesarios** ya que la idea no es tener código muerto, con esto podemos ir creando los selectores para tener:

- Dinero del banco.
- Turnos del juego.
- Jugadores totales.
- Cartas en juego.
- Propiedades en juego.
- Casillas especiales.
- Tamaño del tablero.
- Cantidad máxima de jugadores, casas, hoteles y dados.
- Impuesto total.

Dado que se tienen los selectores, se debe empezar con los modificadores, estos juegan el rol clave de ir actualizando los valores de cada TDA, aquello se refiere a ver si alguna propiedad tiene dueño, si el jugador avanza, si está en la cárcel, si debe de pagar o va a comprar, si lo que va a comprar va a ser una propiedad o una casa, inclusive construir un hotel, etc.

Naturalmente cada modificador debe de tener en cuenta las reglas establecidas por la propia lógica del juego, como bien se menciono antes se debe verificar que X propiedad no tenga dueño, que si X jugador se puede mover o si está en la cárcel, existe la posibilidad de que pase por la salida y por tanto se le debe dar el dinero correspondiente y en el caso particular de **Capitalia** cobrar el impuesto asociado a ello, ver la casilla en la que el jugador se encuentra parado, verificar que los dados sean o no dobles, etc. Todos los puntos antes mencionados hacen de los modificadores extremadamente importantes en el laboratorio particular y son básicamente la columna vertebral del proyecto.

Se denota la existencia de los TDA “otros”, normalmente encargados de cosas fuera de los 5 TDA principales como el lanzamiento de los dados, por ejemplo.

Aspectos de implementación:

Estructura del Proyecto: Cada TDA se encuentra en su archivo respectivo, ósea, se encuentran separados uno del otro así permitiendo un manejo mas organizado y coherente para el desarrollo del proyecto.

Bibliotecas empleadas: Se hace uso de racquet como parte del propio IDE utilizado para el desarrollo del proyecto, en particular se ocupan solo funciones primitivas encontradas en lenguaje estándar Scheme (R6RS).

Razones de uso: Dada la naturalidad el proyecto y el entorno académico se utiliza solamente lo entregado por la misma coordinación del ramo para no acomplejar ni ir en contra de los deseos propios de esta última.

Compilador/Interprete utilizado: Se hace uso de Racket, lenguaje de programación que posee su propio IDE el cual es utilizado ya que es una variante del lenguaje necesario a ocupar Scheme.

Instrucciones de uso:

Para utilizar el programa en si se deben seguir los siguientes pasos:

- Abrir la carpeta contenedora de nombre “Lab1_21142512_Ignacio_TapiaDonaire” y verificar existencia de archivos (Anexo B)
- Abrir el archivo “script_base_21142512_Ignacio_TapiaDonaire.rkt”, el cual debe de abrir el IDE “Dr.Racket” (Anexo C)
- Hacer click en el icono “Run” encontrado en la esquina superior derecha, al lado izquierdo del icono “Stop”
- Verificar mediante la propia consola que el programa ejecute de manera exitosa

Estos mismos pasos deben de ser ejecutados para el script_2 y script_3 encontrados en la misma carpeta.

Resultados y autoevaluación:

La autoevaluación se encuentra dentro de la misma carpeta de “Lab1_21142512_Ignacio_TapiaDonaire”, se debe destacar el RF21 se encuentra con 0 ya que no se fue capaz de concretar, sin embargo, todos los demás puntos se encuentran hechos y los scripts pueden ejecutar.

Con ello se puede decir que los resultados finales no son satisfactorios para la resolución del proyecto ya que no se concreto el ultimo RF necesario para la ejecución completa del juego en cuestión.

Conclusiones:

El paradigma funcional supone un primer y gran desafío en la carrera, no solo cambia la perspectiva completa de como se debe analizar sino de también como se debe aproximar uno a la hora de solucionar problemas, el cambiar de mentalidad y ser capaz de adaptarte al mismo es sin lugar a duda un desafío que no se estimaba a ser tan grande.

A futuro se debe tener en cuenta que cada paradigma es un mundo distinto y por tanto deben de ser tratados como tal a pesar de encontrar similitudes las cuales favorecen el aprender dicho paradigma, para el caso del paradigma funcional es notable lo limpio y a la vez lo engorroso que puede llegar a ser si no se trata con cuidado, existen varias comodidades como el no preocuparse por variables y por lo mismo también supone un desafío.

Referencias:

Anexos:

```
Descripción: Constructor para crear las propiedades dentro de CAPITALIA
id (int) nombre (string) precio (int) renta (int) color (string) dueño (string)
juego (lista)
recursion: no aplica

; Descripción: Constructor para las cartas
; DOM: id (int) tipo (string) descripción (string)
; REC: carta (lista)
; Tipo recursión: No aplica







































define (propiedad id nombre precio renta dueño casas esHotel estaHipotecada
t id nombre precio renta dueño casas esHotel estaHipotecada)) (define (carta id tipo descripción acción)
(list id tipo descripción acción))

Descripción: Constructor de los jugadores en CAPITALIA
DOM: id (int) nombre (string) personaje (string) dinero (int) propiedades (list) posicion
REC: jugador (lista)
Tipo recursion: no aplica

define (jugador id nombre dinero propiedades posicion estaEnCarcel totalCartasSalirCarcel)
(list id nombre dinero propiedades posicion estaEnCarcel totalCartasSalirCarcel))

Descripción: Constructor que inicializa el juego CAPITALIA
DOM: player (lista) board (board) dineroBanco (int) numeroDados (int) turnoActual (int) tasaImpuesto (int)
EC: juego (lista)
ipo recursion: No aplica

fine (juego jugadores board dineroBanco numeroDados turnoActual tasaImpuesto maximoCasas maximoHoteles )
list jugadores board dineroBanco numeroDados turnoActual tasaImpuesto maximoCasas maximoHoteles))
```

 compiled		28-04-2025 20:25	Carpeta de archivos	
 Autoevaluación_21142512_Ignacio_Tapia...		05-05-2025 3:27	Documento de te...	1 KB
 carta_211425121_Ignacio_TapiaDonaire		28-04-2025 20:25	Racket Document	2 KB
 juego_211425121_Ignacio_TapiaDonaire.b...		28-04-2025 20:25	Archivo BAK	7 KB
 juego_211425121_Ignacio_TapiaDonaire		05-05-2025 3:27	Racket Document	9 KB
 jugador_211425121_Ignacio_TapiaDonaire...		28-04-2025 20:25	Archivo BAK	7 KB
 jugador_211425121_Ignacio_TapiaDonaire		05-05-2025 2:11	Racket Document	9 KB
 main_211425121_Ignacio_TapiaDonaire.bak		28-04-2025 20:25	Archivo BAK	1 KB
 main_211425121_Ignacio_TapiaDonaire		05-05-2025 3:27	Racket Document	3 KB
 propiedad_211425121_Ignacio_TapiaDona...		28-04-2025 20:25	Archivo BAK	7 KB
 propiedad_211425121_Ignacio_TapiaDona...		04-05-2025 18:47	Racket Document	9 KB
 Repositorio_21142512_Ignacio_TapiaDona...		05-05-2025 2:17	Documento de te...	1 KB
 script_2_211425121_Ignacio_TapiaDonaire...		05-05-2025 0:00	Archivo BAK	1 KB
 script_2_211425121_Ignacio_TapiaDonaire		05-05-2025 0:41	Racket Document	5 KB
 script_3_211425121_Ignacio_TapiaDonaire...		05-05-2025 0:13	Archivo BAK	1 KB
 script_3_211425121_Ignacio_TapiaDonaire		05-05-2025 0:43	Racket Document	5 KB
 script_base_211425121_Ignacio_TapiaDon...		28-04-2025 20:25	Archivo BAK	4 KB
 script_base_211425121_Ignacio_TapiaDon...		05-05-2025 2:00	Racket Document	11 KB
 tablero_211425121_Ignacio_TapiaDonaire		28-04-2025 20:25	Racket Document	3 KB

Anexo B



Anexo C