
**UNIVERSIDAD DE SANTIAGO DE CHILE**
**FACULTAD DE INGENIERÍA**
**DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA**

# LENGUAJE Y MÉTODOS DE PROGRAMACIÓN

# PUZZLE CANDY CRASH EN PROGRAMACION FUNCIONAL

Profesor: Roberto González
Alumno: Roberto Orellana

Fecha de Entrega: 15/05/2017

Santiago de Chile

03 - 07 – 2017

# TABLA DE CONTENIDOS

# TABLA DE CONTENIDOS

# CAPÍTULO 1. INTRODUCCION

El siguiente informe trata de explicar cómo llegar a la solución del laboratorio N°2 de Lenguaje y métodos de Programación.

## 1.1 ANTECEDENTES Y MOTIVACIÓN

El laboratorio consta de crear un puzzle tipo Candy Crush en cuatro lenguajes distintos de programación, siendo este el segundo en Lenguaje de Programación Funcional (Scheme) en el entorno de desarrollo Dr.Racket. La motivación es el deseo del aprendizaje continuo de un nuevo lenguaje y método de programación

## 1.2 DESCRIPCIÓN DEL PROBLEMA

El Problema consiste en crear un juego tipo CandyCrush con el lenguaje de programación Funcional (scheme) el cual consiste en mover y eliminar una cantidad de dulces que contiene un tablero, con un cierto límite de movimientos, la forma de eliminar dichos dulces consiste en unir tres o más dulces del mismo tipo, al eliminar la mayor cantidad de dulces consigues puntos que te permiten subir de nivel.

## 1.3 SOLUCIÓN PROPUESTA

La solución propuesta a este problema radica inicialmente en el análisis de los requerimientos funcionales y no funcionales contenidos en el enunciado del Laboratorio, y de esta forma llegar a una posible solución comenzando por la implementación apropiada para los problemas a través del TDA, luego la creación del tablero que es la base de todas las funciones, se utiliza como representación lista de listas que contienen elementos al azar, e ir acoplando todas las otras funcionalidades que contiene el juego.

## 1.4 OBJETIVOS Y ALCANCES DEL PROYECTO

### 1.4.1 Objetivo General

El Objetivo general del proyecto consta en lograr el correcto funcionamiento del programa a través del lenguaje de programación Scheme, utilizando lo aprendido en clases y laboratorio sobre paradigma de programación Funcional.
Conocer el correcto uso de cada código utilizado en cada función, reconocer el tipo de recursión, reconocer las 6 capas de la estructura del TDA (Representación, Constructores, Funciones de Pertenencia, Selectores, Modificadores y Otras Funciones).

### 1.4.2 Objetivos Específicos

Lograr que el usuario (con conocimientos básico en programación de Scheme (DrRacket)) pueda crear tableros del tipo recursión lineal (con estado pendientes), y recursión de cola (sin estado pendientes) a partir de una semilla, comprobar que el tablero que se está utilizando contiene los requisitos mínimos que este debe tener, que el usuario pueda hacer movimientos e intercambiar Candys, comprobar y eliminar aquellos que contengan 3 o más Candys iguales adyacentes en una línea y que puedan imprimir el tablero por pantalla.

### 1.4.3 Alcances

El alcance o “Scope” del proyecto se enmarca en la funcionalidad para crear tableros, verificar su contenido y jugar un juego que pueda ser ejecutado por personas con conocimientos básico sobre Scheme.
Obtener un correcto funcionamiento del programa gracias a los objetivos y problemas que se plantearon anteriormente.

## 1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

Las herramientas y técnicas utilizadas para la solución de final del problema constan de lo aprendido en clases (TDA, recursiones, etc), investigación propia del alumno y las metodologías enseñadas por el profesor, analizando el problema y teniendo una abstracción de cómo resolverlo.
El Software utilizado en mi caso fue la de DrRacket que gracias a su fácil compilación y ejecución hace que sea una interfaz más amigable para el usuario.

# CAPITULO 2. FUNDAMENTOS TEORICOS

## 2.1 Método

El Método Utilizado para representar o modelar la creación del programa fue el **TDA** (tipo de dato Abstracto) El juego CandyCrush consiste en intercambiar piezas (dulces) adyacentes en cualquier sentido para crear una fila de 3 o más dulces iguales.

1 El tablero solo puede contener dulces válidos.
2 debe crear siempre el mismo tablero a través de la misma semilla
3 Las dimensiones del tablero no tienen mayor restricción, siempre que no sean menores que 6, la cantidad de dulces no sea menor a 5 o mayor que 7

- **Representación**: es la base de todo, sabiendo bien lo que se va a representar y cuáles son las características y limitaciones del lenguaje.

- **Constructores:** en una función que retorna el tipo de dato abstracto, esta función debe comprobar que se utilizan datos de entrada adecuados y entrega resultados que sean interpretadas por las demás funciones.

- **Funciones de Pertenencia:** son creadas para verificar si un parámetro corresponde a un tipo de dato específico.

- **Selectores:** corresponde a las funciones que nos permitirán acceder a cualquiera de los datos representados, por medio de su retorno.

- **Modificadores:** son las funciones que tienen la capacidad de retornar el TDA o parte de el con información modificada.

- **Otras Funciones:** Estas funciones hacen uso de las funciones y estructuras definidas anteriormente.

## 2.2 Definiciones

- **Paradigma de programación Funcional:** La programación Funcional es un paradigma basado en el uso de funciones matemáticas, la programación funcional tiene sus raíces en el cálculo lambda, es un lenguaje donde las variables no tienen estado, no hay cambios en éstas a lo largo del tiempo y son inmutables. Además los programas se estructuran componiendo expresiones que se evalúan como funciones.

- **Funciones de orden superior:** Funciones de orden superior son funciones que pueden tomar otras funciones como argumentos o devolverlos como resultados.

- **Recursividad:** Las funciones recursivas se invocan a sí mismas, permitiendo que una operación se realice una y otra vez hasta alcanzar el caso base.

- **Función:** una función es un grupo de instrucciones con un objetivo en particular y que se ejecuta al ser llamada desde otra función o procedimiento. Una función puede llamarse múltiples veces e incluso llamarse a sí misma (función recurrente).

- **Matriz:** corresponde a un conjunto ordenados de datos, agrupados en filas y columnas. Esta estructura tiene la capacidad de almacenar elementos numéricos como también alfanuméricos

# CAPITULO 3. DESARROLLO DE LA SOLUCION

## 3.1 Análisis

En el análisis la deducción principal fue la implementación el TDA y luego la creación del tablero y poder trabajar sobre él, que es donde van todas las otras funciones del juego.

**Representación:** Para el proyecto se utiliza como representación lista (Fila) de listas (columnas) que contienen elementos al azar (números). Se prefiere esta representación por su fácil entendimiento, uso y análisis de filas.
‘( ‘(….) ’(…. ). . . . ‘(….) )
En la representación, una lista encapsula todo el tablero, mientras que en su interior cada lista representa filas del tablero. Los elementos al interior de las listas (filas) internas representan las celdas con numeros del tablero.

**Constructores:** para crear el tablero necesitamos crear una lista de listas que representan las filas y dentro de las listas (filas) los Candy aleatorios que representaran a las columnas. Se define la función CreateBoardRL función crear tablero con recursión Lineal, con estados pendientes.
Función CreateBoardRC función crear tablero con recursión de cola, sin estados pendiente. Se define la función "random-candy" que recibe una lista y devuelve un contenido de la lista al azar.

**Funciones de Pertenencia:** tenemos la comprobación de los valores ingresados en el tablero, además tenemos las funciones del checkboard que comprueba si el tablero cumple con los parámetros para ser considerado como un tablero valido.

**Selectores:** me permite obtener los valores del tablero (fila, columna, Candy) para posteriormente hacer los movimientos (Candy origen, Candy destino), y obtener la primera fila de Candys iguales. “obtenerPorigen” función obtiene Origen Candy, “obtenerPdestino” función obtiene destino Candy, “obtenerCandy” función obtiene Candy

**Modificadores:** me permite modificar los valores de dos puntos dentro de la tabla ademas de la eliminación y creación de nuevos Candy.
“remplazaCandy” remplaza el dulce de una lista en una ubicación por un elemento (otro dulce), “tableromod” modifica el tablero con las dos posiciones intercambiadas

**Otras Funciones:** contiene todas las otras funciones que me permiten hacer uso del tablero, como imprimir tablero, juegar con el tablero, etc.

### 3.2 Diseño

Podríamos decir que el juego no tiene diseño o interfaz gráfica ya que funciona en base a llamadas de funciones, a través de interfaz de DRracke se puede hacer el llamado a las distintas funcionalidades del programa, CreateBoardRL, CreateBoardRC, CheckBoard, Play, CheckCandy y board -> string, al ingresar cada una de estas funciones se debe ingresar con los parámetros básicos mencionados en el siguiente ejemplo y tener conocimiento básico sobre programación de Scheme, por ejemplo:

**CreateBoardRL**

```
(define (createboardRL N M candy goals seed)
```

Esta función crea un tablero con Recursión Lineal, la función es llamada createboardRL, y esta función contiene 5 parámetros, N que contiene el tamaño de las Filas, M contiene el tamaño de las Columnas, Candy contiene la cantidad de tipos de dulces que desea dentro del tablero, el rango no puede ser menor a 3 tampoco mayor que 5, goals es la cantidad de candys que debes destruir para superar el nivel, seed es la semilla que te permite crear el mismo tablero mientras la semilla sea la misma, si cambias el número de la semilla el tablero se generara con nuevos candys. Por ejemplo:

**N:** corresponde al tamaño de la fila.
**M:** corresponde al tamaño de la columna.
**Candy:** corresponde a la cantidad de tipos de Candy en el tablero, desde el 5 hasta 7.
**Goals:** corresponde a la cantidad de candys a reventar para superar el nivel,
**Seed:** corresponde a la semilla para crear el tablero, si cambia la semilla cambia el tablero.

**CreateBoardRC**

```
(define (createboardRC N M candy goals  seed)
```

Esta función crea un tablero con Recursión de Cola, la función es llamada createboardRC, y esta función contiene 5 parámetros, N que contiene el tamaño de las Filas, M contiene el tamaño de las Columnas, Candy contiene la cantidad de tipos de dulces que desea dentro del tablero, el rango no puede ser menor a 3 tampoco mayor que 5, goals es la cantidad de candys que debes destruir para superar el nivel, seed es la semilla que te permite crear el mismo tablero mientras la semilla sea la misma, si cambias el número de la semilla el tablero se generara con nuevos candys. Por ejemplo:

**N:** corresponde al tamaño de la fila.
**M:** corresponde al tamaño de la columna.
**Candy:** corresponde a la cantidad de tipos de Candy en el tablero, desde el 5 hasta 7.
**Goals:** corresponde a la cantidad de candys a reventar para superar el nivel,
**Seed:** corresponde a la semilla para crear el tablero, si cambia la semilla cambia el tablero.

## CheckBoard

```
(define (checkBoard board)
```

Esta función recibe un tablero (board) y permite verificar si este cumple con los criterios para ser considerado un tablero valido, si el tablero es válido entrega `#t`, y si no es un tablero valido entrega `#f` la función es llamada checkBoard, y esta función contiene 1 tablero (board), como en el ejemplo anterior board tiene 5 parámetros,

## Play

```
(define (play board pOri pDest seed)
```

Esta función Realiza las jugadas en el tablero, la función es llamada play y recibe como parámetros un tablero (board), posición de origen de Candy (pOri), posición de destino del Candy (pDest) y una semilla (seed) para generar nuevos dulces en el caso de que se encuentre con 3 o más dulces iguales en la misma línea.

**pOri** contiene una lista con dos parámetros para el Candy origen, fila y columna. (list 3 3)
**pDest** contiene una lista con dos parámetros para el Candy destino, fila y columna. (list 4 3)
**seed** contiene la semilla en caso de que se genere una fila de 3 o más Candys iguales (1)

## CheckCandie

```
(define (checkCandies board)
```

Esta función recibe un tablero (board) y permite verificar si 3 o más Candy están unidos en una misma fila, la función es llamada checkCandies, y esta función contiene 1 tablero como en los ejemplos anteriores.

## Board→String

```
(define (board->string board)
```

Esta función recibe un matriz (board) e imprime el tablero por pantalla para una mejor visualización para el usuario, la función es llamada board→string, y esta función contiene 1 tablero como en los ejemplos anteriores.

# CAPITULO 4. RESULTADOS

Se da cumplimiento a los objetivos del proyecto, creando las funciones requeridas; una función que realiza una jugada, y funciones que validan el resultado del tablero.
Se cumple también con aceptar diferentes tamaños de tablero, y a su vez, rellenarlos con dulces aleatorios según la semilla ingresada

# CAPITULO 5. CONCLUSION

El uso del TDA se hace fundamental a la hora de representar o modelar la creación de un programa, ya sea en programación funcional como el que acabamos de hace, o cualquier otro lenguaje y método de programación.
Gracias a lo aprendido en clases y en este laboratorio logre adquirir nuevos conocimientos sobre este nuevo lenguaje de programación funcional para mí. Adquirí gran conocimiento sobre Funciones, funciones de primera clase y de orden superior, Recursividad y su forma de operar en el modo lineal (con estados pendientes) y de cola (sin estados pendientes), edemas de ser un lenguaje bastante robusto basado solo en funciones, su sintaxis es bastante clara y entendible.

# CAPITULO 6. REFERENCIAS


https://www.udesantiagovirtual.cl/
https://www.google.cl/
https://www.youtube.com/
https://es.wikipedia.org/
Larson, Jim, *An Introduction to Lambda Calculus and Scheme.*