

**Área Académica de Ingeniería en Computadores**

**Lenguajes, Compiladores e intérpretes (CE3104)**

**Tarea# 1: RubikSimulator**

**Realizado por:**

**Brayan León Urbina 2018234632**

**Yerald Salazar Elizondo 2018099115**

**Profesor:**

**MARCO RIVERA MENESES**

**Fecha: 20/05/2020**

**Descripción del problema**

Se desea crear un simulador de cubo Rubik, el cual debe ser de tamaño nxn elegible, darle una combinación inicial de colores y recibir una lista de movimientos el cual se le debe aplicar al cubo, este cubo debe tener animaciones de movimiento e ir cambiando a tiempo real mientras se le van aplicando las órdenes dadas hasta llegar al resultado final.

La lógica del programa y la interfaz para el usuario deben ser construidos en el lenguaje de programación Racket, cuya estructura principal de datos son las listas, y la lógica debe ser escrita utilizando el paradigma funcional, el cual se basa en el uso de funciones para la resolución de los problemas.

**Objetivo General**

Desarrollar una aplicación que permita reafirmar el conocimiento del paradigma de programación funcional.

**Objetivos Específicos**

Crear una aplicación que resuelva el problema utilizando Racket.

Aplicar los conceptos de programación funcional.

Crear y manipular listas como estructuras de datos.

* 1. **Descripción de las funciones implementadas.**

A continuación se detallaran todas las funciones utilizadas para el manejo de datos del programa dividas por secciones de acuerdo a su función

**Manejo de listas**

Funciones básicas para el manejo de listas que servirán como base para el manejo de matrices

(get x lista)

Busca el elemento en la posición x en lista.

(eliminar\_indice x lista)

Elimina el elemento en la posición x de lista.

(alfinal dato lista)

Ingresa el elemento en dato en la última posición de lista.

(poner\_adelante dato x lista)

Ingresa el elemento en dato adelante del elemento que se encuentra en la posición x de lista,

(poner\_atras dato x lista)

Ingresa el elemento en dato atrás del elemento que se encuentra en la posición x de lista.

(editar dato pos lista)

Cambia el valor del elemento en la posición pos de lista por dato

(define (invertir lista)

invierte una lista dada

**Manejo de Matriz**

Funciones para el manejo de matrices, usadas y creadas principalmente para el algoritmo de rotar la cara un cubo.

(rotar\_adelante i f matriz) y (rotar\_atras i f matriz)

rotar\_adelante rota los elementos en sentido horario 90° y rotar\_atras en sentido antihorario de una matriz de i a f formando un cuadrilátero cuyas esquinas serian (f,i),(i,i), (i,f) y (f,f)

(rotar\_arriba\_a i f p matriz) (rotar\_derecha\_a f p matriz) (rotar\_abajo\_a i f p matriz) (rotar\_izquierda\_a i f p matriz)

(rotar\_arriba\_at i f p matriz) (rotar\_izquierda\_at f p matriz) (rotar\_abajo\_at i f p matriz) (rotar\_derecha\_t i f p matriz)

Esta serie de funciones mueven las partes de filas y columnas para rotar\_adelante y rotar\_atras de i a f, deben usarse en ese orden para un correcto funcionamiento, p es un contador, rotar\_arriba\_a simula la rotación de la fila de arriba hacia la columna de la derecha, rotar\_derecha\_a simula la rotación de la columna de la derecha hacia la fila de abajo, rotar\_abajo\_a simula la rotación de la fila de abajo hacia la columna de la izquierda y rotar\_izqierda\_a finalmente simula la rotación de la columna de la izquierda hacia la fila de arriba, la serie de funciones de rotar\_lado\_at realizan lo mismo pero en sentido contrario(arriba a izquierda y así sucesivamente).

(mover\_ele\_matriz\_adelante fila columna columnafin filafin matriz) (mover\_ele\_matriz\_adelante fila columna columnafin filafin matriz)

Mueven un elemento que se encuentra en columna, fila hacia adelante o atrás del elemento en matriz que se encuentra en la posición columnafin,filafin. Tomar en cuenta que primero se elimina el elemento movido y después se coloca en la nueva posición para efectos de los valores de los parámetros

(get\_columna x matriz)

Obtiene la columna en la posición x de matriz.

(editar\_columna datos x matriz)

Edita la columna en la posición x con los valores de datos (datos debe contener la misma cantidad de elementos que el largo de las columnas de la matriz) de matriz.

(rotar\_cara\_atras i f matriz) (rotar\_cara\_adelante i f matriz)

rotar\_adelante rota los elementos en sentido horario 90° y rotar\_atras en sentido antihorario de una matriz de i a f formando un cuadrilátero cuyas esquinas serian (f,i),(i,i), (i,f) y (f,f) y luego de i+1 a f+1 y así sucesivamente hasta llegar a f=i o i>f.

**Manejo del cubo**

(rotar\_f x dir cubo n) (rotar\_derecha\_cubo x cubo) (rotar\_izquierda\_cubo x cubo)

Simulan la rotación de una fila de un cubo en la posición x con dirección dir(“D” derecha y “I” izquierda) de un cubo de 6 caras( entre las caras en posición 1,5,2,4) y de tamaño nxn cada cara, rotar\_f manda a llamar a las ultimas 2 y a la de rotar cara si x es 1 o n.

(rotar\_c x dir cubo n) (rotar\_arriba\_cubo x cubo) (rotar\_abajo\_cubo x cubo)

Simulan la rotación de una columna de un cubo en la posición x con dirección dir(“A” arriba y “B” abajo) de un cubo de 6 caras( entre las caras en posición 1,3,6,4) y de tamaño nxn cada cara, rotar\_c manda a llamar a las ultimas 2 y a la de rotar cara si x es 1 o n.

(RS x cubo movs)

Función principal de la tarea programada, pide las matrices de un cubo nxn donde x es n y una lista con los movimientos de la forma “mnd” donde m tipo de movimiento puede ser “F” fila o “C” columna, n es la posición de la fila o columna es un valor númerico, y d es la dirección que puede ser “D” derecha e “I” izquierda para “F” y “A” arriba y “B” abajo para “C”

(realizar\_movimientos x cubo movs)

Función que realiza los movimientos de movs recursivamente en las matrices cubo de tamaño nxn donde n es x.

(realizar\_mov x cubo mov)

Función que manda a realizar el movimiento mov en las matrices cubo de tamaño nxn donde n es x.

**Comprobaciones de datos**

Funciones para comprobar que los datos ingresados sean correctos

(es\_int x)

Verifica que el es string x sea un número.

(es\_len\_fila x fila)

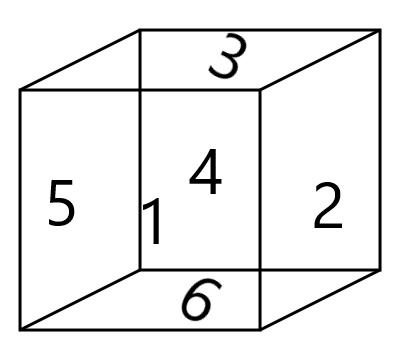
verifica que la fila sea de tamaño x.

(es\_len\_columna x matriz)

verifica que la columna de matriz sea de tamaño x.

**1.2. Descripción de las estructuras de datos desarrolladas.**

Para este proyecto se necesitaba una estructura de dato que simulara las caras de un cubo por lo cual se utilizó una lista 6 matrices como estructura donde cada cara del cubo era una lista de listas ósea una matriz y finalmente todas las matrices estarían en una lista donde su orden en la lista diría la cara del cubo que es quedando en el siguiente orden:



En forma 2d quedaría de la siguiente forma

(3)

(5)(1)(2)(4)

(6)

Y para efectos de racquet seria de la siguiente forma para un 2x2:

( ((1 1) (1 1)) ((2 2) (2 2)) ((3 3) (3 3)) ((4 4) (4 4)) ((5 5) (5 5)) ((6 6) (6 6))

Y con esta estructura se desarrolló la lógica del movimiento y para efectos de los colores de la interfaz se debe utilizar “b” blanco, “r” rojo, “y” amarillo, “a” azul “n” naranja y “v” verde.

**1.3. Descripción detallada de los algoritmos desarrollados.**

**Algoritmo para rotación de una matriz**

Dado que el mover una fila o una columna 1 o n de un cubo implica la rotación de una cara del cubo, y la cara del cubo es una matriz, se necesita plantear un algoritmo que simule esta rotación en la matriz por ello se diseño un sistema en el que se intercambian los datos desde un punto i(i,i) inicial hasta un j(j,j) final, pasando las filas (los elementos i,i a i,j y los elementos de j,i a j,j ) a ser columnas y las columnas (los elementos de i,i a j,i y de i,j a j,j) a ser filas, esto se repite desde los puntos i=0 a j=n bajando 1 en j y subiendo 1 en i ósea i=1 y j =n-1 hasta que i rebase o iguale a j donde se dará por hecho que se ha rotado cada parte de la matriz para una rotación completa

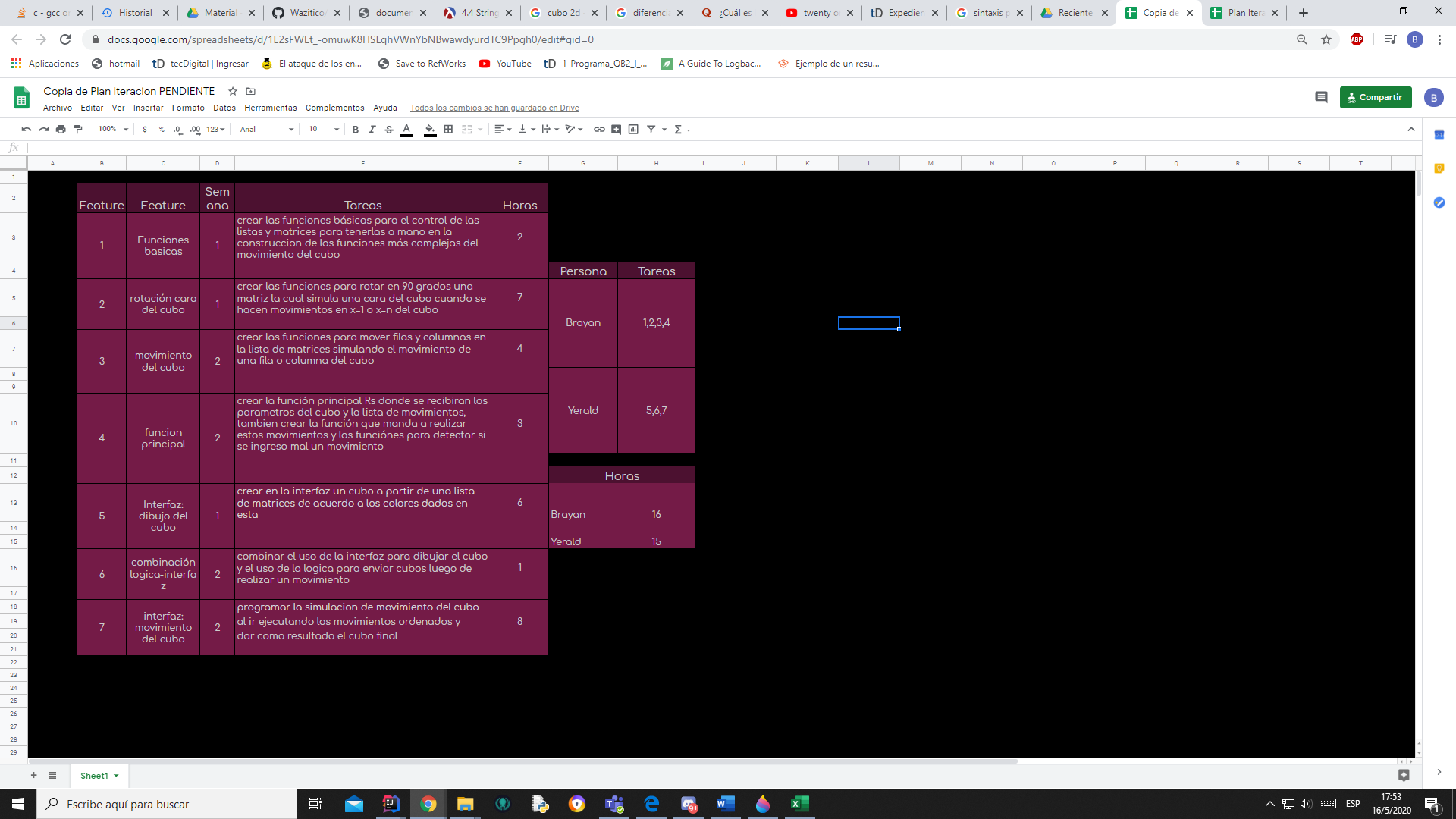
**Algoritmo de simulación de rotar filas y columnas de un cubo**

Para esto se debe tener claro la estructura que se va a usar como cubo, el orden de las caras, y las conexiones ficticias entre las caras, con esto el algoritmo es bastante sencillo dado que mover una fila del cubo implica intercambiar las filas de las matrices que están en la misma hilera ya sea hacia la derecha o la izquierda, y las columnas de las matrices que están en la misma línea vertical cuando se trata de mover una columna del cubo.

**1.4. Problemas sin solución.**

Aun no existen problemas encontrados, por tanto, no hay problemas sin solución para este momento.

**1.5. Plan de Actividades realizadas por estudiante**



**1.6. Problemas encontrados**

No se ha detectado problemas en las ejecuciones de prueba hasta el momento.

**1.7. Conclusiones y Recomendaciones del proyecto.**

**Conclusiones**

* Es importante plantearse y crear funciones bases para la utilización en funciones más complejas y estas misma en otras más complejas ya que puede volver funciones muy complicadas, construibles y relativamente más sencillas de realizar.
* Se puede ver como el manejo de listas, las cuales son muy usadas en racket pueden llegar a modelar problemas tan complejos como la simulación de un cubo y las ordenes de movimiento de este con simples listas y una buena estructura.
* Es importante ver como un buen modelo de estructura puede llegar a simplificar y determinar la solución de un problema en específico, en este caso el de un cubo.
* Una buena organización y elección de ambos seguir una misma estructura desde un inicio puede ayudar a que no ocurran errores a futuro o haya incongruencias entre cada una de las partes realizadas.
* La importancia de que en un lenguaje de paradigma funcional hacer muy buen uso de las funciones creadas, que están sean claras, concisas y dividan bien el problema para mantener un buen orden.

**Recomendaciones**

* Aprender muy bien el uso de recursividad para los lenguajes de paradigma funcional ya que estos serán sus mayores aliados a la hora de programar con estos lenguajes
* Aprender muy bien la estructura de racket ya que esta puede ser un poco diferente a las que uno esta acostumbrado como java o c++, dado que si no encontrar errores incluso de sintaxis se puede volver complicado si no se tiene una buena comprensión de este
* Llevar un buen orden de un conjunto de funciones comentar y separar si es necesario dado que si no, puede llegar a ser muy complicado encontrar funciones, o llevar un buen seguimiento de estas al no existir clases con que separarlas.

**1.8Bibliografia**

• Guzman, J. E. (2006). Introducción a la programación con Scheme. Cartago: Editorial Tecnológica de Costa Rica.

Docs.racket-lang.org. (2019). Strings. [online] disponible en: <https://docs.racket-lang.org/reference/strings.html>

**2. bitácora:**

8 de marzo. Fui pensando la lógica para la simulación del cubo del cubo cuales van a ser las funciones que voy a necesitar, que funciones básicas necesito para llevar a cabo las grandes.

9 de marzo. Le explico un poco a Yerald el cual es mi idea respecto a la lógica, que es lo que podría enviarle para la interfaz del cubo y como más o menos debería ser la lógica de la interfaz respecto a las animaciones de movimiento y empiezo a crear las funciones básicas para la lógica que sería sobre el control de listas.

11 de marzo. Termino de crear las funciones básicas y empiezo a utilizarlas para crear funciones para el control de matrices, con las más importantes que es mover elemento de una matriz a otra posición

12 de marzo. Empiezo las funciones que simulan las rotaciones del cubo de una cara, lo cual seria rotar una matriz en una dirección empiezo con líneas relativas en forma de un cuadrado de la matriz.

13 de marzo. Terminó de crear las funciones para rotar un cuadrado de la cara y la uno en otra función recursiva que haga girar toda la cara.

15 de marzo. Hablo con Yerald de que ambos estuvimos algo enfermos y en mi caso no pude trabajar ayer, se define el orden de las matrices para poder empezar a trabajar en la lógica de mover las filas y columnas, pruebo las funciones y encuentro un error en una de las funciones base o básicas que creaba -1 al rotar la cara por un orden de las variables empiezo y termino de crear las funciones para mover filas y columnas, creo un par de funciones para obtener filas y columnas de una matriz y con ello mover filas y columnas se vuelve muy fácil.

16 de marzo, creo la función principal la del cubo con sus métodos para ver el tipo de movimiento e ir haciendo cada uno de los movimientos para ello también investigue sobre el manejo de strings, pruebo que todo funcione correctamente y funciona bien y documento internamente cada función para finalizar.