Instituto Tecnológico de Costa Rica
Ingeniería en Computadores

Lenguajes, compiladores e intérpretes

Tarea 1

Pablo Naranjo Monge - 2020084495
Jan Marschatz Aguilar - 2020026368
Axel Cordero Martínez - 2019052017

Rubik Simulator

Bitacora

Fecha	Actividad	Participantes
27/2/22	Nos reunimos en llamada y creamos los documentos iniciales	Axel – Jan - Pablo
1/3/22	Iniciamos la programación creando archivos RS y creando funciones	Axel - Jan
5/3/22	Se añada documento con utilidades para el algoritmo de listas	Pablo - Jan
6/3/22	Se añadieron más funciones varias y validaciones en el archivo de Utilidades.rkt	Jan
6/3/22	Se agrega archivo de gui con funciones para generar cubo	Pablo
6/3/22	Se agregan condiciones de verificación en archivos RS.rkt y en Utilidades.rkt	Axel
7/3/22	Se añade fondo, función de click y función de pintar en el GUI.rkt	Pablo
7/3/22	Se añade función de verificar condiciones sobre columnas y filas en RS.rkt	Axel
7/3/22	Se añade en condiciones para rotación de filas y columnas	Jan
7/3/22	Se añade función de rotar caras en el GUI.rkt	Pablo
8/3/22	Se crea función de cambiar filas en Utilidades.rkt	Jan – Pablo - Axel
8/3/22	Se crea función de update que se conecta con la parte grafica	Jan – Pablo - Axel
8/3/22	Función de rotación de filas-columnas-caras	Jan – Pablo - Axel
9/3/22	Documentación Interna	Jan – Pablo - Axel
9/3/22	Creación de manual y documentación técnico	Jan – Pablo - Axel

1.1. Descripción de las funciones implementadas.

Función RS:

La función RS va a ser la primera función que llamaría el programa lo cual verifica condiciones para ver si las entradas que da el jugador están correctos y que puedan proceder a la parte grafica.
Las verificaciones se hacen en subfunciones que tienen que dar falso o verdadero los cuales son
(lenFilCol)
(verificarMovs)
(largo Matriz)
Estos se explicarán en algoritmos desarrollados.

Funcion elemento:

La función elemento extrae el n-esimo elemento de una lista, el elemento que extrae puede ser una lista, un numero entero, un string entre otros. Primero verifica si la lista ingresada está vacía, en caso de que lo esté, retorna falso ya que no se puede buscar un elemento dentro de una lista vacía. En caso de que no esté vacía se comprueban dos condiciones más, la primera verifica si el index coincide con 1, en caso de ser así significa que el elemento buscado es el primero de la lista por ende se retorna un (car lista) el cual regresa el primer elemento de una lista, en caso de que esta condición no se cumpla se envía a la última, se llama recursivamente la función elemento restándole uno al index y aplicándole un “cdr” a la lista, que sería toda la lista menos el primer elemento.

Función Update:

La función update revisa primero que el tamaño del cubo este dentro del rango, segundo que la variable Cubo sea una lista y que no esté vacía, tercero que Movs sea una lista y que no esté vacía.

Función fila-col?:

Lee la primera instrucción que viene en la variable Movs(lista de movimientos) y la separa por fila o columna y dirección, luego verifica si coincide con alguna de las condiciones y si coincide, envía los parámetros de fila y columna a la siguiente función.

Función fil-col-central?:

La función determina si la fila/columna que se quiere mover se ubica en el "centro" o es una "esquina". Cuando hablamos de centro nos referimos a que solo cuatro de las seis caras sufren un cambio y esquinas cuando cinco de las seis caras sufren un cambio. Esta validación se hace con el número que viene en la primera instrucción dentro de la lista Movs.

Función Esquinas:

La función verifica la "esquina" y la dirección que se desea cambiar dentro del cubo. Primero la función compara el número de “esquina” a ver si es la primera o la última, luego compara la variable “dir” para ver en qué dirección se está moviendo la fila/columna. Una vez que entra en alguna de las condiciones se llama a la función actualizar-fila la cual realiza el cambio. Nos referimos a esquinas cuando cinco de las seis caras sufren un cambio.

1.2. Descripción de las estructuras de datos desarrolladas.

Listas: esta se utiliza para el manejo de los movimientos que se desean realizar en el cubo, en la forma ‘(F1D F2I C2A C1A). De esta en la primera ejecución se toma el primer elemento para ser analizado, luego el segundo y así sucesivamente.

Matrices: el cubo fue representado como matriz, esta posee en su primer nivel, una lista de las caras del cubo, luego cada una de estas tiene una lista con las filas y sus respectivos colores. De forma que por ejemplo, un cubo se vería representado de la siguiente manera.

```
'( ( (A A)
    (A A))
     ( (B B)
      (B B))
     ( (V V)
      (V V))
     ( (Y Y)
      (Y Y))
     ( (R R)
      (R R) )
     ( (N N)
      (N N)  ))
```

1.3. Descripción detallada de los algoritmos desarrollados.

lenFilCol: Función que verifica lo largo de las filas, las columnas y si el cubo tiene 6 caras.

Esta función se implementa mediante el algoritmo de **recursión de cola** y tiene que dar de salida un falso o un verdadero.

verificarMovs: Función que verifica que los movimientos tengan formato correcto (ej. (F1D)).

Esta función se implementa mediante el algoritmo de **recursión de pila** y tiene que dar de salida un falso o un verdadero.

Sus: va recorriendo toda la lista que se provee, hasta llegar a la posición del elemento que se desea sustituir, en este caso, tomará el nuevo elemento y aplicará un cons con el cdr de la lista y suma 1 al contador, por otro lado, cuando todavía lo va recorriendo, hace un cons del car de la lista con una llamada recursiva de esta misma función enviando como lista el cdr de la lista y aumentando el contador de posición recorrida.

Cambiar-fila: esta emplea la función sus, para sustituir un elemento dado como newfila en la fila número y número de cara que se desee realizar el cambio en el cubo, esta retornará el cubo con el cambio, para ello en el primer llamado de sus se realiza un cambio total de una cara, para luego tomar esa cara y cambiarle una fila, luego la función sus reconstruirá esta fila con el resto de la cara y luego con el resto del cubo.

actualizar-fila: Hace el cambio de fila ya sea hacia la izquierda o hacia la derecha

La función de actualizar-fila, primero revisa si el parámetro “dir” corresponde a izquierda o derecha y luego se llama a la función cambiar-fila(la cual realiza el cambio como tal) recursivamente para cambiar todas las caras que se ven afectadas por este cambio. La función actualizar fila siempre realiza una rotación hacia la izquierda, entonces para rotar derecha se ejecuta la rotación hacia la izquierda tres veces, esto se debe a que, en un cubo cuando uno hace una rotación hacia la derecha es lo mismo que rotarlo tres veces hacia la izquierda.

rotarDer: Realiza una rotación hacia la derecha del cubo, para lo que se realizan 3 movimientos hacia la izquierda, implementando el llamado la función actualizar fila.

Esta función se encarga de rota una de las “esquinas” indicadas, se llama tres veces a la función de actualizar-fila(la cual mueve una fila hacia la izquierda), esto se debe a que, en un cubo, rotar una vez a la derecha es lo mismo que rotar 3 veces a la izquierda y viceversa.

1.4. Problemas sin solución: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

Uno de los problemas que no se logró dar una solución es que el orden de las columnas quedó invertido, esto no supone problemas al momento de manejar la lista, pero inicialmente se deseaba que la columna 1 se situara a la izquierda y quedó a la derecha.

1.5. Plan de Actividades realizadas por estudiante: Este es un planeamiento de las actividades que se realizarán para completar la tarea, este debe incluir descripción de la tarea, tiempo estimado de completitud, fecha de entrega y responsable a cargo.

Roles Asignados	Participante
Creación de condiciones de verificación y movimientos del cubo	Axel
Creación de Utilidades y movimientos del cubo	Jan
Creación de GUI y movimientos del cubo	Pablo

Fecha	Actividades	Participantes
Semana1 – 2 dias	Hacer reunión e investigar sobre la parte lógica y grafica de Racket.	Axel – Pablo - Jan
Semana1 – 2 dias	Empezar a crear funciones de verificar y el GUI	Axel - Pablo
Semana 1 – 3 dias	Crear un archivo de función útiles que nos facilitan la tarea	Pablo - Jan
Semana 2 - 2 dias	Crear Gui que pinte líneas, cubos y caras de un cubo rubik	Pablo
Semana 2 – 2 dias	Condiciones y movimientos de cubo Rubik completado con GUI	Axel – Pablo - Jan
Semana 2 – 3 dias	Arreglar errores y mejorar condiciones	Axel – Pablo - Jan

1.6. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

- Rotación derecha de la matriz: Inicialmente se deseaba realizar una función específica para la rotación derecha, el desarrollo de esta se complicó y se optó por hacer una función para que realice 3 rotaciones izquierda.

- Manejo de las filas, durante el desarrollo, las filas fueron un problema, ya que suponen sustituir muchos elementos de forma individual, entonces la solución más optima fue realizar una función que rote el cubo hacia la izquierda completamente y tratar las columnas como filas, reutilizando todos estos métodos.

1.7. Conclusiones y Recomendaciones del proyecto.

Como principales conclusiones, se desprenden que se logró aprender el manejo de listas en Scheme de manera exitosa, ya que el manejo de movimientos está basado en funciones de *car* y *cdr*, también el uso de paradigma funcional y recursión. Por otro lado, se obtuvo exitosamente un programa que simula un cubo Rubik y logra realizar los movimientos, además, de una interfaz gráfica que muestra en cubo desde diferentes perspectivas.

Se recomienda el uso de programas externos para visualizar un cubo en 3 dimensiones si no se tiene uno físico, también el planear de forma lógica los movimientos antes de implementarlos, dado que de esta forma se pueden ver situaciones como a las que se llegaron en este trabajo, como el que era más fácil realizar una función que rotara el cubo completo hacia la izquierda, para poder tratar las columnas como filas y así no tener que invertir tiempo en funciones para mover las columnas.

1.8. Bibliografía consultada en todo el proyecto.

Racket-Lang. (s.f). *Ghaphics: Legacy Library.* https://docs.racket-lang.org/graphics/index.html

Helo, J. (2005). *Introducción a la programación con Scheme.* Editorial Tecnológico de Costa Rica.