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

Laboratorio 1
Paradigmas de programación
(Funcional - Scheme)
“Aplicación de edición de imágenes”

Nombre: Alan Berrios Estay
Profesor: Roberto Gonzales
Fecha de entrega: 26/09/2022

1) Introducción:

Este informe corresponde a la serie de tres laboratorios respectivos al curso de Paradigmas de programación, donde se tiene como tema una aplicación de edición de imágenes simplificada, tipo GIMP o Photoshop. En esta primera experiencia, se pide realizar un programa para realizar un tratamiento de imágenes simplificado. Este programa se realizaría en el lenguaje de programación Scheme, el cual es usado mediante la aplicación DrRacket, donde se implementarían conceptos del paradigma funcional mediante dicho lenguaje de programación.

2) Descripción del problema:

Se pide realizar un programa funcional en función al tema de una aplicación de imágenes simplificada. Este programa sería capaz de realizar diferentes funciones en torno a los pixeles de una imagen. Para implementar esto, es necesario tener en cuenta 2 componentes principales.

Pixeles: Estos representan los pixeles de una imagen, los cuales pueden variar entre 3 tipos, pixbit-d, pixrgb-d y pixhex-d. Son necesarios para el concepto de imagen y a su par son el centro de las funcionalidades del programa, ya que estos son los que serán modificados mediante las funciones.

Imagen: Es el concepto principal del programa, este se compone de pixeles de un solo tipo y es la entrada y salida de gran parte de las funciones.

Estos componentes deben ser implementados mediante Scheme, siguiendo el Paradigma Funcional.

3) Descripción del paradigma:

Como ya se ha dicho, en esta experiencia se trabajará con el Paradigma Funcional, el cual, tiene como base a las funciones en términos de abstracción y programación (programación funcional). Estas funciones se podrían explicar en tres elementos, la entrada o dominio de la función, el cual es el elemento a procesar, luego está el desarrollo de la función, el cual puede realizar diferentes procesos, y por último esta la salida o recorrido, que es el elemento de entrada ya procesado y devuelto por la función. Este paradigma trabaja en el término del ¿QUE?, el cual hace referencia a obtener el resultado y no en como obtenerlo. Aparte de este término, también se utiliza la notación del Calculo Lambda, la cual es una notación que permite expresar funciones, en este caso, basándose en la notación polaca. Como el Paradigma funcional solo trabaja con funciones, se idearon algunos conceptos aplicativos para poder ampliar las posibilidades de trabajo de este paradigma, como lo son las funciones anónimas, recursiones (y sus diferentes tipos), composición de funciones, currificacion y funciones de orden superior. En el caso del lenguaje Scheme, no es únicamente seguidor del Paradigma funcional, ya que también utiliza programación imperativa.

4) Análisis del problema:

Para el trabajo del problema, se realiza todo mediante el programa DrRacket, donde solo trabajaremos con la programación funcional e imperativa. Con esto, pasamos a analizar los componentes del problema. El fin del problema es poder representar una imagen y con ello, realizar diferentes funciones. En primer lugar, tendríamos el concepto principal que es el TDA Image, esta es una representación de una imagen visual, pero aplicada en código.

- Imagen: está conformada por su ancho y alto, y una cantidad n de pixeles, correspondientes al ancho y alto de la imagen (Número de pixeles en la coordenada X e Y respectivamente), y luego los pixeles de la imagen.

Siguiente a esto, tenemos los otros componentes importantes para la construcción de dicha imagen, que son los pixeles, como estos tienen 3 diferentes tipos, son 3 diferentes representaciones. TDA Pixbit-d, TDA Pixrgb-d y TDA Pixhex-d.

Teniendo estos 3 elementos básicos, podemos crear la representación de una Imagen, el cual es el elemento principal del programa, y con ello, se puede efectuar e incorporar funciones en torno a estas representaciones, de esta forma, pudiendo desarrollar el problema de este trabajo. Ahora se presentarían las diferentes funcionalidades principales que tendría este programa respecto a la imagen y sus pixeles:

- **bitmap?:** Función que verifica si una imagen tiene todos sus pixeles de tipo pixbit-d
- **pixmap?:** Función que verifica si una imagen tiene todos sus pixeles de tipo pixrgb-d
- **hexmap?:** Función que verifica si una imagen tiene todos sus pixeles de tipo pixhex-d
- **compressed?:** Función que verifica si una imagen esta comprimida
- **flipH:** Función que voltea horizontalmente una imagen
- **flipV:** Función que voltea verticalmente una imagen
- **crop:** Función que recorta una imagen a partir de un cuadrante
- **imgRGB->imgHex:** Transforma una imagen desde una representación RGB a una HEX.
- **rotate90:** Función que rota una imagen 90° hacia la derecha
- **compress:** Función que comprime una imagen eliminando el color más frecuente
- **edit:** Función que permite aplicar funciones especiales a una imagen
- **invertColorBit:** Función que permite invertir el color de una imagen bitmap
- **invertColorRGB:** Función que permite invertir el color de una imagen pixmap
- **adjustChannel:** Función que permite modificar un canal de una imagen pixmap
- **image->string:** Función que transforma una imagen a una representación string
- **depthLayers:** Función que separa una imagen por capas basándose en la profundidad de los pixeles
- **decompress:** Función que descomprime una imagen

Tener en cuenta que, para realizar estas funciones, es necesario estrictamente tener una imagen definida, la cual es parte del Dominio de cada una de estas funciones.
También, se pide que cada TDA (Image y pixeles) tengan un archivo por separado, donde se definirán las funciones anteriores en cada TDA, dependiendo de su correspondencia. También es posible crear funciones propias para facilitar el desarrollo de las principales.

5) Diseño de la solución:

Para el diseño de la solución, primero se diseño las representaciones de los TDA (Image y pixeles), ya que estos son lo principal del programa y respecto a estas representaciones se guiarán las funciones principales. (Respecto a las funciones propias, algunas de estas se mencionarán, se explicará en donde se usó y que hacen en la tabla anexada).

- **TDA Imagen:** lista: (ancho (Int) X alto (Int) X n-Pixeles). Se dice n ya que cada imagen puede tener un ancho y alto diferentes, y junto a ello, puede variar la cantidad de pixeles.

Tener en cuenta que todos los pixeles tienen coordenadas, y estas están representadas como (y, x) y cada ultimo valor de un pixel es la profundidad (Depth). Sus representaciones son:

- **TDA Pixbit-d:** lista con estos elementos: (y (Int) X x (Int) X bit (Int) X Depth (Int)). Con "bit" un valor 0 o 1, perteneciente al color.
- **TDA Pixrgb-d:** lista con estos elementos: (y (Int) X x (Int) X R (Int) X G (Int) X B (Int) X Depth (Int)). Con R, G y B como un valor entre 0 y 255.
- **TDA Pixhex-d:** lista con estos elementos: (y (Int) X x (Int) X string (Int) X Depth (Int)). Con string como el color del pixel en formato hexadecimal (Ejemplo color blanco: "#FFFFFF").

Con estas representaciones mencionadas, todas las funciones obligatorias se guiarán con estas respecto a sus procesos. Ahora se explicará brevemente la metodología de algunas de estas funciones:

- **Bitmap?, Pixmap? y Hexmap?** se idea de forma que una función selector consiga el pixel y mediante una función pertenencia verifique si ese pixel es de un tipo especifico, aplicando este proceso a cada pixel de la imagen.
- Respecto a las funciones **FlipH** y **FlipV,** estas modifican las coordenadas de los pixeles dependiendo si es en horizontal o vertical (se modifica y en función del alto y x en función del ancho) de forma de recursión de cola que junta todos los pixeles volteados en una lista, luego de eso se ordenan de forma que queda representado bien el giro.
- Para la función **Crop**, se pensó en comparar si un pixel estaba dentro del cuadrante dado, si es así, ese pixel estaba dentro del recorte, luego se deberían rescalar los pixeles y la imagen para que no se descordinen el ancho, alto y las coordenadas de los pixeles.
- En **imgRGB->imgHex**, se obtiene los canales de cada pixel (R, G, B), y estos se convierten a hexadecimal, luego se unen en un solo string y se integran nuevamente a sus respectivos pixeles.
- **Rotate90** es una función que realiza un giro vertical de la imagen, para luego invertir las coordenadas de cada pixel, así efectuando un giro hacia la derecha.
- Para **Compress**, se obtiene gracias a un histograma, cual es el color mas repetido de la imagen, y con ese dato, construir nuevamente la lista de pixeles, pero omitiendo los que tengan ese respectivo color.

Estas serían las metodologías de algunas funciones. Para estas, se anexarán ejemplos de cómo funcionan visualmente.

6) Aspectos de implementación

Para la implementación de este proyecto, se utilizaron 4 diferentes archivos conectados entre sí, que son para los 4 respectivos TDAs, están conectados ya que algunas funciones son necesarias en otros archivos. La biblioteca usada fue la de Racket, proporcionada por el mismo programa de DrRacket, esto debido a que sus funciones que no están en Scheme son de alta utilidad, aparte de estar permitidas para este trabajo. Y como se dictó, se trabajó únicamente en el programa DrRacket.

7) Instrucciones de uso

Para poder hacer uso de las funciones, primero hay que tener todos los archivos necesarios dentro de una sola carpeta, (TDAs) y tener instalado DrRacket versión 8,6, por lo contrario, no se podrá ejecutar ninguna operación. Para poder empezar a realizar funciones, se puede abrir tanto el archivo de TDA-Image como el Script de pruebas, debido a que ambos archivos incorporan todo lo necesario. Una vez ejecutado, apretar Run en el programa DrRacket, lo cual abrirá la consola, donde se podrán ejecutar las diferentes funciones. En el archivo de Script de pruebas hay varias imágenes ya creadas a modo de ejemplo, pero para crear una desde 0, tiene la siguiente estructura:

```
> (define imagen (image 2 2 (pixbit-d 0 0 0 10)(pixbit-d 0 1 1 11)(pixbit-d 1 0 1 12)(pixbit-d 1 1 0 13)))
```

```
Representacion visual: ancho/alto
                    0          1
                0  (0 0 1 10)  (0 1 1 11)

                1  (1 0 1 12)  (1 1 0 13)
```

Como se ve, así se crea una imagen, en este caso es una imagen de 2x2 de tipo bitmap.
Con esta imagen ya creada, podemos hacer uso de las diferentes funciones que nos permite este programa. Todas obtendrían el resultado esperado si se cumplen los parámetros requeridos de cada función, por ejemplo, no se podría hacer una función imgRGB->imgHex a una imagen que es de tipo bitmap o hexmap, debido a que esta función no esta pensada para ese tipo de imágenes. Si se intenta hacerlo, esto daría como ejemplo:

```
> (imgRGB->imgHex imagen)
list-ref: index too large for list
index: 4
in: '(0 0 0 10)
```

Pero en caso contrario todas las funciones deberían entregar un resultado acorde a lo ingresado.

8) Resultados y autoevaluación:

En términos de objetivos cumplidos, se implementaron correctamente todas las funciones menos la función Decompress, que no está implementada, debido a falta de tiempo y dudas no respondidas en su intento de creación, por lo cual si se quiere hacer (decompress imagen), devolverá la misma imagen. Por lo tanto, el grado de alcance de las funciones debería ser total, exceptuando funciones el cual el Dominio ingresado es erróneo al que se requiere en dicha función. En el Script de pruebas se hicieron pruebas de todas las funciones, los ejemplos que utilicen decompress no obtendrán el resultado esperado.
Como autoevaluación, se piensa que se cumplieron los objetivos y se realizaron correctamente las funciones, exceptuando decompress. Respecto a los puntajes elegidos se encuentran en el archivo Autoevaluación.txt donde se verá que en el punto decompress tiene 0pts debido a lo dicho anteriormente.

9) Conclusión

Finalmente, luego de haber realizado el trabajo, se puede concluir que los objetivos de esta experiencia se han completado. Los cuales serían tanto comprender y manejar el lenguaje de programación Scheme Racket y aprender el paradigma funcional y ejecutar sus diferentes aplicaciones como recursiones, composiciones, composiciones, etc. en términos de funciones. Objetivos los cuales se ven reflejados en el trabajo en si realizado.
Una de las mayores dificultades al realizar esta experiencia fue aprender y familiarizar con el paradigma y a su par con el lenguaje de programación, ya que son formas un poco distintas de como veníamos abordando nuestros paradigmas de programación de los anteriores cursos de programación, donde trabajamos con otros paradigmas ni si quiera sin saber su término.
Producto de esto, surgían errores que uno no comprendía a la hora de hacer el código, ya sea por estar implementando mal el uso de paréntesis o por mal uso de funciones propias del lenguaje, y eso siguió efectuándose ya en las ultimas funciones las cuales eran difíciles de abordar ya que se requerían muchas funciones propias que funcionasen entre si para poder finalmente conseguir lo esperado. Agregando a esto, el uso de Git no fue muy intuitivo al principio, pero se logró familiarizar con su uso. Otra dificultad era el tiempo de dedicación al laboratorio, y en opinión personal si no hubiera sido por la pasada semana de receso, se hubiera complicado mas aun el desarrollo de esta experiencia que ya de por si es ardua.

Ya fuera de las dificultades, se lleva un sentimiento de satisfacción al haber podido completar la experiencia de buena forma y se toma en cuenta que esto servirá como preparación para las próximas experiencias y quizás facilite un poco el desarrollo de estas en términos de entender los paradigmas y los lenguajes.

10) Referencias

- Image representation — Data Quality Explored. QuaXP (s. f.)., de https://www3.tuhh.de/sts/hoou/data-quality-explored/2-1-1-imagere-presentation.html

- Introduction to Digital Images. Standford University, de https://web.stanford.edu/class/cs101/image-1-introduction.html

- House, D. H. (2014, 8 enero). Digital Image Basics. Clemson University. Recuperado, de https://people.cs.clemson.edu/~dhouse/courses/405/notes/pixmaps-rgb.pdf

- Vista de La Programación Funcional: Un Poderoso Paradigma. (2008) Universidad Católica de Salta. de http://revistas.ucasal.edu.ar/index.php/CI/article/view/182/159

- Racket v8.6. (2022, 10 agosto), de https://blog.racket-lang.org/2022/08/racket-v8-6.html

11) Anexo

Tabla de algunas funciones propias.

Nombre función	Donde se usa	Tipo de función	Breve explicación
int>0?	Pixeles	De pertenencia	Verifica si la entrada es un entero y si es mayor o igual a 0.
gety	Múltiples funciones	Selector	Funcion que obtiene la coordenada y de un pixel.
getx	Múltiples funciones	Selector	Funcion que obtiene la coordenada x de un pixel.
getD	Múltiples funciones	Selector	Funcion que obtiene la profundidad de un pixel.
getR	Múltiples funciones	Selector	Funcion que obtiene el canal R de un pixrgb-d
getG	Múltiples funciones	Selector	Funcion que obtiene el canal G de un pixrgb-d
getB	Múltiples funciones	Selector	Funcion que obtiene el canal B de un pixrgb-d
homologo?	image	De pertenencia	Verifica si los elementos de la lista de entrada son de un mismo tipo de pixel (pixbit-rgb-hex-d)
dentrodelarea?	image	De pertenencia	Verifica si las coordenadas x e y de cada pixel estan dentro del alto y ancho de la imagen
getancho	Múltiples funciones	Selector	función que recibe el ancho de una imagen.
getalto	Múltiples funciones	Selector	función que recibe el alto de una imagen
getpixeles	Múltiples funciones	Selector	función que recibe la lista de pixeles de una imagen.
ordenarx	Múltiples funciones	Modificador	función que ordena de menor a mayor la coordenada x de una lista de pixeles
ordenary	Múltiples funciones	Modificador	función que ordena de menor a mayor la coordenada y de una lista de pixeles
ordenarpixeles	Múltiples funciones	Modificador	función que ordena de menor a mayor la coordenada x e y de una imagen
mymaprecur	FlipH – FlipV	Otras funciones	Versión recursiva de la función map