Informe Laboratorio 1
Paradigma funcional
Programación en Lenguaje Racket

Nombre: Iván Alejandro Guajardo Arias.
Curso: Paradigmas de programación.
Sección: A-1.
Profesor: Gonzalo Martinez.

26 de Septiembre, 2022.

Tabla de contenidos:

2. Introducción

En el informe que se presenta a continuación, se expone el primer laboratorio del ramo "Paradigmas de programación"; el cual, se basa en el paradigma funcional, por medio del lenguaje educativo, Racket. El proyecto consiste en desarrollar un programa que facilite la manipulación de imágenes con un enfoque simplificado, utilizando el, ya mencionado, lenguaje Racket y su IDE, DrRacket.
Para lograr el objetivo general del laboratorio, se necesita entender el enunciado desde la perspectiva funcional y, por supuesto, guiarse por las funciones solicitadas como requisitos funcionales básicos. Dentro de los pasos consecuentes, se pueden encontrar los siguientes:

1. Entender el propósito del proyecto y abstraer el problema, para así implementar los TDA pertinentes para la simulación del editor de imágenes.
2. Plantear las funciones pertinentes que faciliten el desarrollo del proyecto, de modo que podamos abordarlo desde un paradigma totalmente funcional.
3. Utilizar correctamente las recursiones vistas en clases, con el fin de poder implementar correctamente las funciones ya mencionadas sin desviarnos de nuestro paradigma propuesto.

En lo que viene de informe, se verá la descripción del paradigma, se hará un marco teórico que facilite la comprensión del problema y, facilite así, el análisis de este. De esta manera, se podrá comprender además cómo el paradigma funcional puede facilitar la resolución de problemas de manera efectiva.

3. Descripción del paradigma

Empezaremos por definir el paradigma funcional. ¿A qué corresponde el susodicho?

“Cuando nos encontramos desarrollamos software utilizando este paradigma, estaremos trabajando principalmente con funciones, evitaremos los datos mutables, así como el hecho de compartir estados entre funciones.” (García Pérez, E., Abril, 2019, ¿Qué es la programación funcional?, https://codigofacilito.com/articulos/programacion-funcional) .
Es un enfoque basado en el uso de funciones. Estas funciones, son primordiales para la resolución de cualquier problema, basándonos en “¿qué queremos hacer?” más que en el cómo. Esto se traduce al uso de distintos tipos de recursión, listas y otros; sin poder definir variables, ya que todo queda en las funciones al llamarlas.

3.1 Marco teórico

Lo más básico para poder resolver el problema, sería empezar por el proceso de abstracción. Dicho proceso se tiende a considerar como el traducir un problema a la generalidad, llevándolo a distintos niveles y subprocesos que faciliten el desarrollo de este. De este modo, se puede plantear un algoritmo concreto que soluciona el problema propuesto en un inicio.
Ahora, sería prudente si también se menciona lo que serían los subprocesos. En el caso del paradigma funcional, podría considerarse como cada proceso que se realice a un nivel menor para poder solucionar distintos problemas. De este modo, se pueden abordar parcialmente y luego complementarse para realizar el proceso final, desde las capas ya mencionadas, que además permiten ir probando y corrigiendo sin errores de arrastre mayores.

Por otro lado, también es importante describir el trabajo en que se basa Racket. Racket es un derivado de Lisp, el cual, por su parte, tiene un enfoque muy grande en el procesamiento de listas. Así pues, se consideran principalmente las listas como herramienta de trabajo para las funciones, haciendo uso de distintos tipos de datos en su interior. Además, vale la pena mencionar el cómo cada función implica su propia localidad, implicando entonces la existencia de datos sólo en su interior, como puede ser el caso de, incluso, otra función de menor orden.

4. Problemática

4.1 Descripción del problema

El problema que el enunciado propone, consiste en el desarrollo de un software que permita la manipulación de imágenes, así como photoshop o GIMP. Sin embargo, este tiene un enfoque simplificado de lo que serían dichos programas, pero permitiría igualmente el uso de funciones tales como la rotación de imagen, voltearla, transformarla, editarla, etc. Otra diferencia destacable entre los softwares mencionados y el propuesto por el laboratorio, es que este último tendría un enfoque principalmente en imágenes RGB/RGB-D y representaciones sencillas de las imágenes, como listas de píxeles u otras.

4.2 Análisis del problema

A grandes rasgos, el problema consiste en presentar un software que permita la edición de imágenes, como los ya señalados. Entonces, lo primero sería considerar que las imágenes consisten en arreglos bidimensionales de píxeles, por lo que sería importante disponer de funciones que permitan la manipulación de los píxeles, sus valores, sus posiciones, etc. Considerando la perspectiva funcional que se le da al laboratorio, es que se propone el uso de listas para poder implementar una solución satisfactoria, teniendo en cuenta además el uso de recursiones para poder modificar las ya mencionadas.

4.3 Otras consideraciones

Se pueden ingresar **N** píxeles a cada imagen, considerando su ancho Width y su alto Height; esto implica entonces la posibilidad de una imagen infinitamente grande, o una diminuta. A partir de ello, es que se podrían considerar los valores Width y Height para poder implementar funciones de volteo, o la posibilidad de un TDA para cada píxel, de modo que se pueda abordar cada caso particular (bit, rgb y hex) de manera adecuada en su respectiva imagen. Por otro lado, también es importante considerar que existen funciones que pueden demandar sub-procesos similares, por lo que es recomendable implementar algunas funciones de orden menor que faciliten estas tareas.

De esta manera, se lograría finalmente crear imágenes en base a listas de píxeles, modificarlas en base a modificar los valores de los píxeles, convertirlas, preguntar de qué tipo son, etc.

5. Diseño de solución

5.1 Descripción de la solución

La solución propuesta, plantea que en un inicio se trabaje desde lo más básico hasta lo más complejo, empezando por píxeles de cada tipo, sus funciones generales y, desde ahí, retomar con lo que corresponde a imágenes como tal, que es lo que aparece en el enunciado del laboratorio. De esta manera, se definen los TDAs pixrgb-d, pixbit-d, pixhex-d y pixel antes que el de imagen, facilitando el posterior desarrollo TDA imagen.
Por su parte, se podría considerar cada TDA como una lista con sus respectivos valores, selectores, modificadores, etc. Finalmente, esto permitiría implementar funciones que faciliten la perspectiva del qué se quiere obtener, proyectando un resultado que cumpla con lo solicitado en cada aspecto de la evaluación.
En cada píxel, se considera un ID dado por el constructor, posición x, posición y, y sus respectivos valores que le dan un significado como píxel, más allá de su ubicación en la imagen y el tipo de píxel; de esta manera, pixrgb-d quedaría ("pixrgb-d" pos_x pos_y r g b d), por ejemplo. Posteriormente, se consideran también las funciones que caracterizan a cada tipo de dato abstracto, como su pertenencia y sus modificadores, que luego son de utilidad para la resolución de problemas propuestos por el enunciado.

No se considera como la solución más refinada, puesto que tiene sus limitaciones en cuanto a la pulcritud del código, sin embargo, cumple perfectamente con los requisitos propuestos en un inicio. Por su parte, cada función ha tenido su utilidad en su respectivo sector o, incluso en otras funciones de mayor grado.

5.2 TDAs

Cada píxel consiste en una lista que considera un ID, sus posiciones x e y, su(s) valor(es) y la profundidad de este. De esta forma, se consideran los ya mencionados como los siguientes ejemplos:
(ver tabla 1 en Anexos)

Por otro lado, es importante considerar que todos estos elementos comparten alguna característica y esto implica el uso de funciones similares, por lo que se consideró un TDA píxel que facilite el seleccionar y modificar dichos valores en común sin la necesidad de invadir al TDA imagen, el principal objetivo del laboratorio.

Finalmente, el TDA image consistiría en algo como lo que se presenta a continuación:
(ver tabla 2 en Anexos)

Siendo p, una cantidad N de píxeles a entregar, Width el ancho y Height el alto, que servirían para poder implementar algunas funciones de posicionamiento.

TDAs implementados:

a. image
b. pixrgb-d
c. pixhex-d
d. pixbit-d
e. pixel

6. Aspectos de implementación

La estructura a seguir por el proyecto contempla el uso de subprocesos que faciliten la implementación de los más complejos. De esta manera, se contemplan 5 archivos: main.rkt, TDA_image.rkt, TDA_pixrgb-d.rkt, TDA_pixhex-d.rkt, TDA_pixbit-d.rkt y TDA_pixel.rkt.
Estos facilitan el orden del código, ya que se implementaron numerosas funciones menores con el fin de realizar las metas propuestas.

Otro aspecto relevante, sería la prohibición de cualquier uso de programación bajo otro paradigma, como lo son por ejemplo las variables. Y los require-provide utilizados para poder armonizar el uso de funciones entre archivos.

7. Instrucciones de uso

Lo más relevante al momento de utilizar el programa, consiste en ingresar correctamente los valores. Cada función tiene definido su dominio y recorrido, también tiene comentada una descripción que facilita su comprensión para no cometer errores. Lo importante en esta experiencia es principalmente ir al archivo main.rkt y compilarlo, puesto que ahí se cuenta con el script de prueba y displays suficientes para mostrar cada ejemplo. Desde luego, no olvidar que todos los archivos deben ir en la misma carpeta para poder funcionar, ya que son codependientes y no funcionan por sí solos.

Finalmente, basta con dirigirse a la carpeta, abrir el programa main.rkt desde DrRacket y presionar el botón Run desde la esquina superior, a la derecha.

8. Resultados

8.1 Ideales

Idealmente, se espera un software de uso simple que permita trabajar píxeles representados como listas. Se logra de manera exitosa el realizar las siguientes acciones en píxeles e imágenes modificar, convertir, manipular, etc. Además, se hace correcto uso del proceso de abstracción para

proponer TDAs que permitan implementar cada función que se proponga dentro del enunciado mediante listas y recursiones principalmente (es esencial para el paradigma funcional, a diferencia de las variables, que derechamente no se deben usar).

8.2 Reales

Se lograron implementar las primeras 12 funciones del enunciado, hasta rotate90. Esto demuestra un buen aprendizaje a nivel estudiante, ya que se usó provechosamente el tiempo y se dio un correcto uso a los TDA, recursiones, listas y otros aspectos vistos en clase. Satisfactoriamente, funciona cada función como debería.

8.3 Posibles fallos en el estudiante

Principalmente, se destaca la falta de orden y claridad al momento de empezar con el proceso de abstracción, desde ahí, se derivan otras falencias como el desorden en funciones de menor relevancia y la abundancia de líneas innecesarias. Es posible que exista una mejor manera de plantear los TDA, aunque de todos modos resultó funcional lo entregado por el estudiante. (Considerar, por favor, que el estudiante entró al ramo 2 semanas tarde por un problema en el sistema).

8.4 Autoevaluación

En aspectos generales, se considera un buen desempeño en cuanto al estudiante. A pesar de ello, vale la pena mencionar que existe un desorden en sus TDAs y funciones, por lo que podría considerarse. Ver tabla 3, adjunta en Anexos.

9. Conclusión

Dentro de todo, se considera como un éxito a la primera experiencia realizada en el laboratorio de paradigmas, puesto que se cumplió con el mínimo de funciones solicitadas satisfactoriamente. Cada una es capaz de correr bien el script del enunciado y este se comporta de la manera esperada. La mayor limitación en este caso, fueron algunos factores externos al ramo y la falta de claridad que tuvo el estudiante desde un inicio, pues se consideró complicado el planteamiento de TDAs y cómo se trabajaría la imagen, principalmente. Después de algunos días de demora, el estudiante logra establecer un orden significativo y esto le permite avanzar lo suficiente; de todos modos, es importante mejorar las buenas prácticas que el estudiante posee para las siguientes entregas, de modo que se optimice el trabajo y se puedan cumplir mejor los plazos, a la vez que se obtenga mayor legibilidad para el código.

10. Anexos

Tabla 1

TDA	Plantilla	Ejemplo
pixbit-d	(type pos_x pos_y bit depth)	(“pixbit-d” 0 1 (0|1) 4)
pixrgb-d	(type pos_x pos_y r g b depth)	(“pixrgb-d” 0 0 255 16 176 74)
pixhex-d	(type pos_x pos_y r g b depth)	(“pixhex-d” 0 1 “#FF00AC” 89)

Tabla 2

image	(Width Height . p)	(2 2 p1 p2 p3 p4)

Tabla 3

TDA	**Valor binario (logrado | no logrado)**
TDA image - constructor	**logrado**
TDA image - bitmap?	**logrado**
TDA image - pixmap?	**logrado**
TDA image - hexmap?	**logrado**
TDA image - compressed?	**logrado**
TDA image - flipH	**logrado**
TDA image - flipV	**logrado**
TDA image - crop	**logrado**
TDA image - imgRGB->imgHex	**logrado**
TDA image - histogram	**logrado**

TDA image - rotate90	**logrado**
TDA image - compress	**no logrado**
TDA image - edit	**no logrado**
TDA image - invertColorBit	**no logrado**
TDA image - invertColorRGB	**no logrado**
TDA image - adjustChannel	**no logrado**
TDA image - image->string	**no logrado**
TDA image - depthLayers	**no logrado**
TDA image - decompress	**no logrado**

11. Bibliografía
    a. https://codigofacilito.com/articulos/programacion-funcional
    b. https://docs.google.com/document/d/1hUAooKwBj3TWv-yuzBZtNbuaC8iNkzOZdbLpD8P9B8c/edit#
    c. https://stackoverflow.com
    d. https://www.cosy.sbg.ac.at/~pmeerw/Watermarking/lena.html
    e. https://uvirtual.usach.cl/moodle/mod/page/view.php?id=156475
    f. https://uvirtual.usach.cl/moodle/course/view.php?id=10036
    g. Profesor Gonzalo Martinez
    h. https://www.youtube.com/watch?v=NwZPlLTK-UQ&list=PLTd5ehIj0goMswKV4Glhr4uixrhCmihIt