

Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática
Paradigmas de Programación



Proyecto semestral de Laboratorio:

Paradigma Lógico

Autor: Franco Barría
Profesor: Miguel Truffa

Santiago, 2022

Tabla contenidos

1. Introducción	3
1.1 descripción del problema	3
1.2 descripción del paradigma	4
2. Desarrollo	4
2.1 Análisis del problema	4
2.2 Diseño de la solución	5
2.2.1 TDA pix	5
2.2.2 TDA image	5
2.3 Aspectos de implementación	6
2.4 Instrucciones de uso	6
2.5 Resultados y autoevaluación	6
3. Conclusión	7
4. Anexo	8

1. Introducción

En los últimos años, el campo de la computación ha evolucionado de manera exponencial, debido a diferentes investigaciones orientadas a este ámbito. Por estas mismas investigaciones, se han creado diferentes dialectos para lograr operar con máquinas, así reinando el modelo de programación imperativa, pero no solo siendo el único paradigma existente.

Así es como surge el propósito de este informe, el cual es el segundo de tres que serán analizados a lo largo de este semestre, de diferente paradigma cada uno. En el informe que se lee hoy, se adentrará en el paradigma lógico, el cual es parte del modelo de programación declarativo, además, se mostrará el uso de este mediante la construcción de una simulación de un programa modificador de imágenes, así como Photoshop o Gimp. Esto será logrado mediante el lenguaje de programación prolog, cuyo avance será documentado con la herramienta GIT (en github.com), la cual servirá para lograr generar un registro de los cambios que se generarán a lo largo de esta experiencia.

1.1 Descripción del problema

Para generar un análisis correcto del problema, en primer lugar se debe contextualizar en que se basa un programa manipulador de imágenes como Gimp, lo cual se basa principalmente en lograr modificar imagenes pixel por pixel, teniendo diferentes formatos de imágenes, estas herramientas han crecido tanto que son capaces de otorgar diferentes tipos de modificaciones en la imagen, tales como difuminaciones, copiar objetos dentro de sí, e inclusive lograr crear nuevas imágenes en base a una principal.

Para lograr desarrollar esta simulación, no se desarrollaran todas las herramientas que contiene un programa manipulador de imagenes actual, pero si las básicas, debido a que no se cuenta con un lapso de tiempo amplio (como los años de desarrollos de photoshop) para lograr una aplicación tan avanzada para manipular imágenes, pero sí las necesarias para demostrar un dominio suficiente en el uso del lenguaje declarativo que se presenta. Cada operación que realice el programa será definida como TDA (Tipo de Dato Abstracto), empezando por un TDA constructor que defina mediante listas el formato de imagen (binaria, RGB o hexadecimal), para dar origen a un TDA al igual constructor que se encargue de crear la imagen.

1.2 Descripción del paradigma

El modelo de programación lógico, consiste en el uso de declaraciones para construir un programa, las cuales se van analizando mediante operadores lógicos y álgebra de Boole. Se debe considerar que en esta forma de programar, en primer lugar hay que crear una base de conocimientos, la cual estará compuesta de **predicados**, después, para operar con estos predicados, se deben crear **cláusulas**, las cuales tendrán diferentes hechos y reglas.

Cuando el programa se está ejecutando, se debe operar mediante **consultas**, en base al programa que se ha diseñado mediante los predicados y cláusulas, y a partir de estas consultas, el programa irá operando y obteniendo los resultados que se requieren, hay que considerar, que las únicas reglas válidas dentro del programa, son las que están descritas, por lo cual eso se considera “el mundo”, y aquí yace uno de los principales problemas de prolog, “el mundo cerrado”, en el que el cual, si no se prevé un predicado, o una cláusula que podría ser verdadera, prolog, al hacer la consulta en la base de conocimientos, devolverá un resultado falso.

2. Desarrollo

2.1 Análisis del problema

Para cubrir lo que necesita el programa, ocuparemos diversos TDAs para construir el programa, los cuales resolverán diferentes problemas que irán apareciendo mediante la creación de la simulación, así como la creación de imágenes, el recorte de ellas, o el historial de cambios.

En el desarrollo de la solución, en primer lugar se debe generar una imagen, para así modificarla, esto será en base a un conjunto de elementos cuales serán entregados al programa, los cuales definirán el tipo de imagen, la ubicación de un pixel y el valor que este tendrá, para posteriormente ser operado con las funciones creadas.

Las funciones que serán capaz de ejecutar el programa serán:

- Crear imagen según su formato.
- Comprobar el formato de una imagen.

- Saber si una imagen está comprimida.
- Voltar una imagen horizontal y verticalmente.
- Cortar una imagen.
- Transformar el formato de una imagen.
- Saber el historial de una imagen.
- Rotar una imagen.
- Invertir colores de una imagen.
- Comprimir una imagen.

2.2 Diseño de la solución

Para generar una solución eficiente al problema planteado, se crearán 2 TDAs principales que son indispensables para la simulación, los cuales serán los encargados de darle la estructura principal al programa, que es la creación de imágenes, una vez ya creados estos TDAs, se puede avanzar creando TDAs nuevos que permitan la modificación de las imágenes, pero como ya dicho antes, es necesario en primer lugar tener la base de los siguientes TDAs.

2.2.1 TDA pix

Para la creación del simulador, es necesario crear un TDA que sea capaz de crear los diferentes tipos de formatos de imágenes, como pixbit, el cual es una imagen de tipo binaria, con valor en cada píxel de 0 o 1; pixrgb, el cual es una imagen de tipo RGB, en la cual en cada pixel tiene 3 valores diferentes entre 0 a 255, representando cada color, rojo, verde y azul; y por último, pixhex, el cual en cada pixel tiene un valor de tipo Hexadecimal, el cual será representado con un String. Para lo propuesto anteriormente, se ocupará un constructor con un formato de listas, cada formato tendrá su TDA diferente en diferente archivo, cuales serán exportados al archivo principal del programa, en el cual se ejecutará el simulador

2.2.2 TDA image

Este TDA juega un rol fundamental, debido a que es el que creará la imagen según el tipo de formato que se explica en el punto anterior, este TDA tendrá el largo de la imagen, el ancho, la lista de píxeles y por último, el nombre de la imagen. Al igual que el TDA pix, también estará en un archivo aparte que será exportado al principal, en el cual habrá más funciones de modificaciones de imágenes, como rotaciones y otras alteraciones.

2.3 Aspectos de implementación

El sistema operativo que fue empleado para desarrollar este programa, fue Windows 10, y también, el compilador que fue ocupado para este proyecto es SWI-Prolog versión 8.4.3.1, este compilador trae consigo herramientas que, como por ejemplo, van analizando lo que se va implementando al código para encontrar posibles errores, como la herramienta trace, que sirve para ir trazando el código paso por paso para ver en qué lugar hay un comportamiento inesperado. También, está la posibilidad de crear archivos .pl independientes en los cuales se puede importar a un archivo .pl principal mediante la función import_module.

2.4 Instrucciones de uso

Para ocupar el programa, en primer lugar se debe acceder al archivo de script de pruebas, llamado “pruebas_20668934K_BarriaRosas” y posteriormente ejecutar las funciones mediante las consultas. En el mismo script de pruebas, se presentan consultas para hacerle al programa. Si se quiere consultar la documentación de cada función, se deben abrir los archivos por separado, ya que cada función está comentada con su propia documentación.

Todas las funciones presentes en el programa, están totalmente implementadas.

2.5 Resultados y autoevaluación

Los resultados que se obtuvieron en esta experiencia, fueron los resultados esperados, aunque no se logró completar el código a la perfección, se logró cubrir la mayoría de las TDAs propuestas, por consiguiente, el programa es operativo y funcional, y listo para ocupar por un usuario.

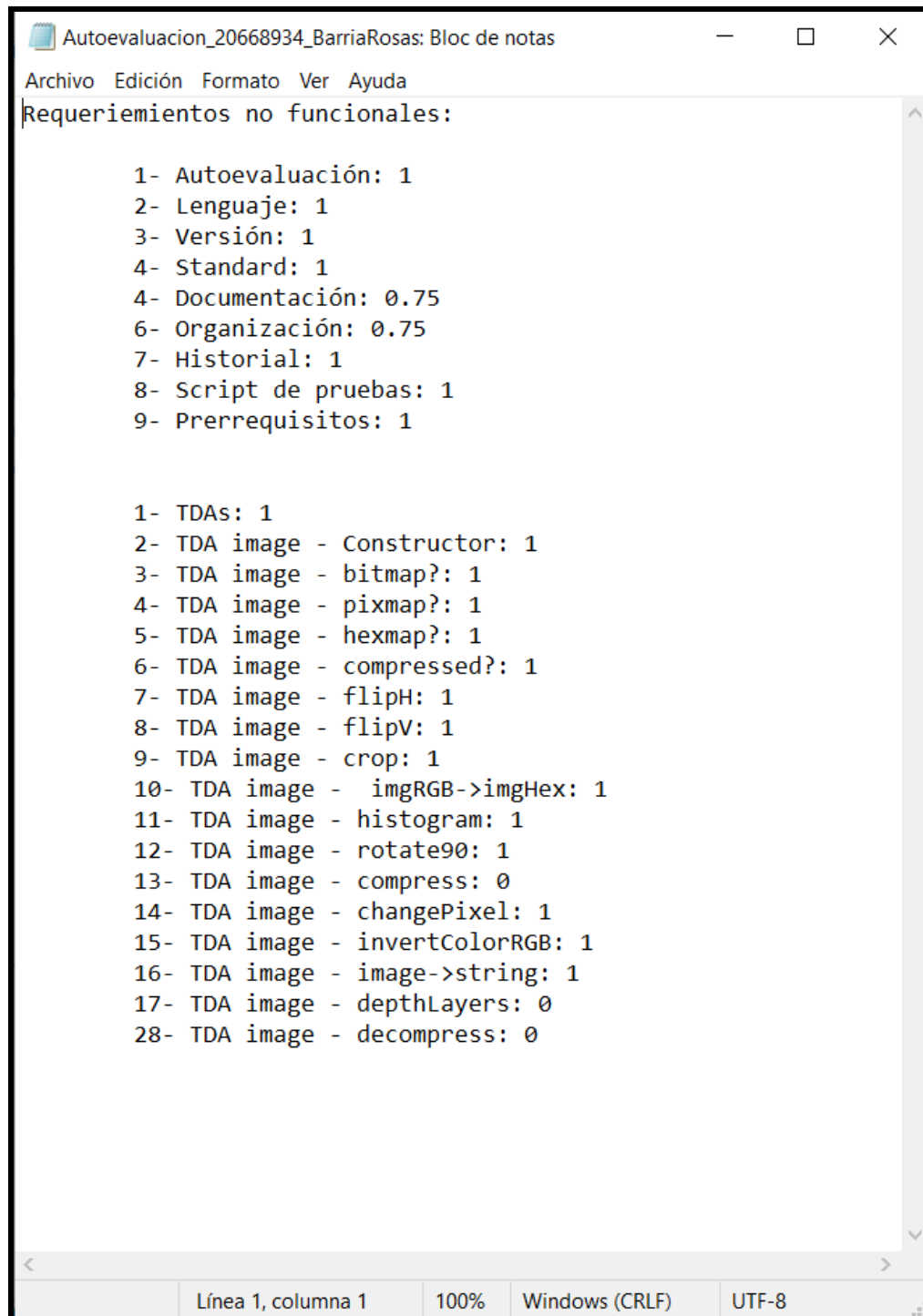
Como autoevaluación, se adjunta en el anexo el contenido de Autoevaluacion.txt, el cual tiene como contenido la autoevaluación por el autor. La escala de evaluación es 0 si no se implementó, 0.25 si se tuvieron problemas, 0.5 si está implementado regularmente, 0.75 si está implementado parcialmente y 1 si funciona correctamente.

3. Conclusión

Finalizando este informe, a lo largo del problema planteado se han encontrado demasiadas limitaciones que fueron haciendo que el trabajo realizado sea menos efectivo, esto es debido totalmente al cambio del paradigma de programación, más no es un problema debido a que aprender nuevos modelos de programación es el objetivo del curso, el cual es un total desafío.

Por otro lado, en la realización de este segundo laboratorio, se encontraron diferencias claras entre el paradigma actual, y el pasado, principalmente en la sintaxis, donde el anterior paradigma era puramente funcional (además de la notación polaca), este era puramente de cláusulas y reglas. Desde el punto de vista propio, el paradigma funcional se hizo más fácil de implementar, debido a que tiene una forma más exacta de llegar al resultado, y pocas formas de llegar a este, en cambio prolog, difiere mucho de la forma que se quiera acceder al resultado, y sobretodo el recorrido de listas, por esto mismo, llegar a la implementación total del paradigma lógico no fue logrado con efectividad, y esta es la razón de porque el programa no tiene todas las funcionalidades propuestas, pero por otro lado, sí plantea las TDAs a resolver, las cuales serán útiles para el aprendizaje de otros paradigmas, y al ya tener la segunda versión de este simulador, queda claro lo que se debe hacer y lo que no para el último laboratorio de java. En cuanto la experiencia obtenida a lo largo de este laboratorio, fue de total ayuda para la formación como programador, ya que se amplía totalmente el conocimiento frente a otras formas de programar.

5. Anexos



```
Autoevaluacion_20668934_BarriaRosas: Bloc de notas
Archivo Edición Formato Ver Ayuda
Requerimientos no funcionales:

1- Autoevaluación: 1
2- Lenguaje: 1
3- Versión: 1
4- Standard: 1
4- Documentación: 0.75
6- Organización: 0.75
7- Historial: 1
8- Script de pruebas: 1
9- Prerrequisitos: 1

1- TDAs: 1
2- TDA image - Constructor: 1
3- TDA image - bitmap?: 1
4- TDA image - pixmap?: 1
5- TDA image - hexmap?: 1
6- TDA image - compressed?: 1
7- TDA image - flipH: 1
8- TDA image - flipV: 1
9- TDA image - crop: 1
10- TDA image - imgRGB->imgHex: 1
11- TDA image - histogram: 1
12- TDA image - rotate90: 1
13- TDA image - compress: 0
14- TDA image - changePixel: 1
15- TDA image - invertColorRGB: 1
16- TDA image - image->string: 1
17- TDA image - depthLayers: 0
28- TDA image - decompress: 0

Línea 1, columna 1 100% Windows (CRLF) UTF-8
```

Anexo: Autoevaluación