
UNIVERSIDAD
DE SANTIAGO
DE CHILE

**Departamento de Ingeniería en Informática**

**Paradigmas de la Programación**

# Laboratorio 2

Editor de Imágenes

**Álvaro Valenzuela**

Licenciatura en Ciencias de la Ingeniería

12 de noviembre de 2022
Profesor: Roberto González

# Índice

# 1. Introducción

En este laboratorio tenemos el desafío de abordar la interpretación y desarrollo de un sistema de edición de imágenes utilizando el paradigma lógico. Recordemos que como gestor de imágenes nos referimos a un software capaz de manipular los colores, dimensiones y otras características de una imagen. En el desarrollo de este informe se describirá la problemática y solución del desafío dividido en los siguientes títulos:

**Descripción del problema** Analizaremos el problema a tratar

**Descripción del paradigma** Describiremos los principales alcances del paradigma

**Análisis del problema** Analizaremos el problema en profundidad

**Diseño de la solución** Desglosaremos la implementación escogida

**Aspectos de la implementación** Se mostrará algunos alcances específicos de la implementación

**Instrucciones de uso** Como se utiliza el programa desarrollado

**Resultados y autoevaluación** Que cosas funcionas versus las que no

**Conclusiones** Aspectos para rescatar y aprendizajes obtenidos en el transcurso del desarrollo del laboratorio

Sobre el paradigma a tratar, en esta ocasión debemos utilizar el lógico. A diferencia del laboratorio anterior (Funcional) en este carecemos de funciones que nos resultaron de bastante ayuda al momento de encapsular funcionalidades. En este paradigma solamente podemos hacer uso de una base de conocimientos y/o implementaciones basadas en listas para lograr representar TDAs. Esto se verá en mayor detalle en el transcurso del laboratorio.

# 2. Descripción del Problema

Actualmente existen diversos softwares para manipular imágenes. Entre ellos podemos destacar Photoshop, GIMP, Paint, entre otros. Estos softwares no hacen más que manipular el conjunto de píxeles que, al renderizarse en una pantalla, dan forma a lo que conocemos como una imagen. Dentro de estos programas destaca el nivel de detalle en el cual nosotros podemos editar una imagen. Por ejemplo, podemos cambiar los colores de los píxeles, redimencionar, recortar imágenes, invertir imágenes, etc. Dado que la lista de posibles combinaciones de acciones es bastante amplia, en este laboratorio se replicarán dichas acciones pero de manera simplificada. Para ello, es importante conocer los 3 tipos de píxeles que tendremos que abordar a lo largo de laboratorio. Estos son:

1. RGB (Red, Green, Blue)
2. HEX (Hexadecimal)
3. BIT (Bit)

Adicionalmente, estos 3 tipos de píxeles deben contener información sobre su profundidad. Este valor adicional debe estar en el rango de 0 a 255.

## 3. Descripción del Paradigma

El paradigma lógico a diferencia del funcional tiene enfoque en realizar consultas sobre una base de conocimientos. Este concepto fue aplicado por primera vez en los años 70 por Colmerauer en la universidad de Marcella [1]. De aquí emergen conceptos que luego son utilizados en herramientas que utilizamos hoy en día. Ejemplo de esto es SQL con su estructura y sintaxis de consultas sobre una fuente de verdad.

Como se mencionaba anteriormente, es importante conocer que en Prolog existe una base de hechos y reglas que operan sobre esta base de hechos (Clausulas). Prolog únicamente tendrá bajo su conocimiento lo que se declare como verdad en las Clausulas. Esto se conoce como el supuesto del mundo cerrado.

También, no solamente nos podemos basar en lo que se declare como verdad en estas clausulas, si no que podemos generar reglas que ayudan a generar restricciones logicas sobre la base de conocimientos. Por ejemplo podemos declarar que existen 3 tipos de autos todos con colores distintos. En base a esto podemos generar una regla que nos permita capturar todos los autos que cumplan con cierto color.

Por ultimo, una de las grandes ventajas de este paradigma es que al ser un enfoque orientado a las consultas este nos permite abstraernos de la lógica de implementación y para así solo enfocarnos en la información que nos interesa obtener desde una base de conocimientos.

## 4. Análisis del Problema

Dado que trabajaremos en un sistema de edición de imágenes y el elemento principal de estas para su representación es el Píxel, podemos representar dicha entidad en un TDA para ser utilizado a lo largo del laboratorio. Es importante recordar que para este laboratorio existen 3 tipos de píxeles y estos deben ser parte del sistema. El píxel RGB-D Debe contener información de sus 3 canales mas la profundidad. El píxel Pixhex debe contener el valor numérico del color en formato hexadecimal y el píxel PixBit debe contener información de su color en un solo bit.

Para estos tipos de píxeles es necesario conocer su posición dentro del plano en donde se representarán. Podemos imaginarnos esto como un plano XY en donde la combinación de las coordenadas serán la posición de cada uno de estos píxeles. Este plano será otro TDA dentro del sistema dado que nos permitirá abstraernos de los elementos que contenga

Con esta información base, ya podemos dar origen a una imagen. En resumen, se trataría de una estructura de datos que contiene información de su posición y de su color. Dicho esto, podemos conocer los requerimientos que deben ser abordados a lo largo del laboratorio. Estos son algunos de ellos:

1. Recortar Imagen
2. Invertir Imagen
3. Comprimir una imagen en base al color de mayor frecuencia
4. Convertir a Hexadecimal
5. Visualizar la imagen
6. Rotar la imagen
7. Histograma

8. Descomprimir una imagen

9. Aplicar operaciones a un área seleccionada

## 5. Diseño de la solución

El enfoque y abstracciones seleccionadas sobre el problema ya señalado van en torno a separar lo que se conoce como un sistema de edición de imágenes en diversas capas de TDAS en donde abordaremos lo mas atómico de un pixel a lo mas general. Es importante mencionar que para esta representación no se hará uso de una base de conocimientos como tal, si no que se hara uso de las listas como estructura de datos.

La abstracción desarrollada tiene las siguientes capas. Primero, tendremos TDAS que representarán los tres tipos distintos de píxeles (RGB-D, Hex-d, Bit-d). Luego, tendremos TDAS específicos para cada uno de ellos que representarán la posición en un plano (PixRGB-D. PixHex-d, PixBit-d). Luego necesitamos agrupar estos Pixeles en una lista y agregarle el ancho y largo de la imagen que vamos a representar. Esta será la estructura base que le daremos nombre como "Image".

Por ultimo, todas las demas operaciones que se desarrollen serán en torno al tda previamente definido. Por ejemplo si queremos voltear una imagen, necesitaremos como input la imagen como tal.

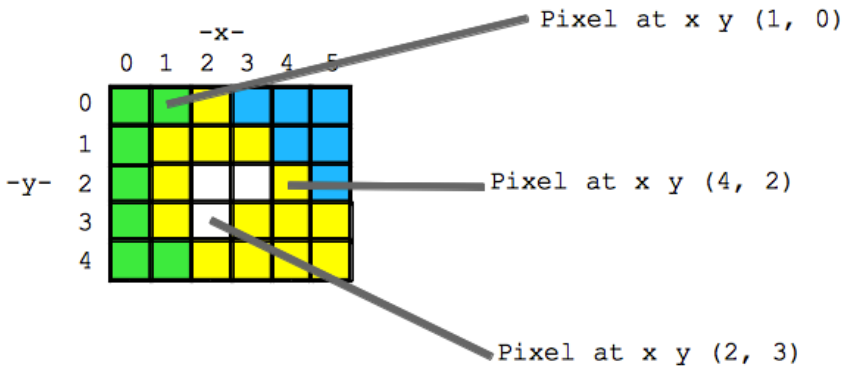


Figura 1: Representación lógica de una imagen en un plano

## 6. Aspectos de implementación

Para el desarrollo de este laboratorio se utilizó VSCode para la elaboración de código y la terminar para la ejecución de este mismo y también se requirió el apoyo de swi-prolog online para hacer debug en el código. Esto ya que las extensiones de VSCode no son tan robustas como la interfaz de esta página.

## 7. Instrucciones de uso

El archivo Pruebas_191118898_AlvaroValenzuela.pl es en donde se encuentran todas las consultas que podemos realizar sobre el archivo TDA_imagen. Luego, como es descrito anteriormente, se dispone de un conjunto de consultas en el archivo Pruebas que pueden ser utilizadas para realizar la construcción de las listas

# 8. Resultados y autoevaluación

| Requerimiento | Grado de alcance |
|---|---|
| image | 100 % |
| bitmap? | 100 % |
| pixmap? | 100 % |
| hexmap? | 100 % |
| compressed? | 0 % |
| flipH | 100 % |
| flipV | 100 % |
| crop | 100 % |
| imgRGB->imgHEX | 100 % |
| histogram | 0 % |
| rotate90 | 100 % |
| compress | 0 % |
| edit | 0 % |
| invertColorBit | 0 % |
| invertColorRGB | 0 % |
| adjustChannel | 0 % |
| image->string | 0 % |
| depthLayers | 0 % |
| decompress | 0 % |

## 8.1. Funciones no implementadas

Las funciones no se implementaron debido a dificultados al momento de interpretar el requerimiento en el paradigma lógico. Un ejemplo puede ser el requerimiento *Histogram*. Aquí la principal dificultad fue lograr extraer y agrupar datos en una lista.

# 9. Conclusiones

A lo largo del desarrollo de este laboratorio fue interesante conocer como el paradigma lógico es capaz de abstraernos de la implementación dura para así enfocarnos en el problema en si. Otros aspectos destacables de este paradigma es el concepto de backtracking. Esto fue de gran ayuda al momento de construir los tdas ya que olvidamos de la implementación iterativa para dejar a prolog que se encargue de esta lógica.

# Referencias

[1] uchile. Prolog. `https://users.dcc.uchile.cl/~abassi/IA/Prolog.html`. (Accessed on 11/12/2022).