

PARADIGMAS DE PROGRAMACIÓN

PROYECTO SEMESTRAL DE LABORATORIO

actualizado al 25/09/2022

Laboratorio 2 (Paradigma Lógico - Lenguaje Prolog)

Versión 1.0

(Cambios menores pueden incorporarse en futuras versiones a fin de aclarar o corregir errores)

(Sus dudas las puede expresar en este mismo enunciado, incluso puede responder a preguntas de compañeros en caso de que conozca la respuesta)

Enunciado General: Procure consultar los aspectos generales del proyecto de laboratorio en el [documento general](#).

Fecha de Entrega: Ver calendario clase a clase donde se señala el hito

Objetivo del laboratorio: Aplicar conceptos del paradigma de programación funcional usando el lenguaje de programación Scheme en la resolución de un problema acotado.

Resultado esperado: Programa para generar un conjunto válido de cartas del juego Dobble y además permitir a un grupo de usuarios jugar.

Profesor responsable: Víctor Flores (al hacer consultas en este documento, procurar hacer la mención a @Victor Flores Sanchez para que las notificaciones de sus consultas lleguen al profesor correspondiente)

Recomendaciones: El laboratorio está diseñado como un conjunto de ejercicios a abordar bajo cada paradigma. En este sentido, el desarrollo del laboratorio constituye un espacio para practicar y prepararse además para la evaluación de cátedra del correspondiente paradigma. Por tanto, se recomienda incorporar en sus hábitos de estudio/trabajo el desarrollo de las funcionalidades de forma diaria. En total el laboratorio 2 lista N funcionalidades a cubrir. Para alcanzar la nota de aprobación, se deben cubrir las A primeras y para la nota máxima se pueden cubrir M más. Procure destinar tiempo para analizar y hacer una propuesta de diseño para el laboratorio completo antes de proceder a la implementación de la solución.

Requerimientos No Funcionales. Algunos son ineludibles/obligatorios, esto quiere decir que al no cumplir con dicho requerimiento, su proyecto será evaluado con la nota mínima.

1. **(obligatorio) Autoevaluación:** Incluir autoevaluación de cada uno de los requerimientos funcionales solicitados.
2. **(obligatorio) Lenguaje:** La implementación debe ser en el lenguaje de programación Prolog en base a una programación principalmente declarativa.
3. **(obligatorio) Versión:** Usar Swi-prolog versión 8.4 o superior.
4. **(obligatorio) Standard:** Se deben utilizar predicados estándar del lenguaje. No emplear bibliotecas externas.
5. **(1 pts) Documentación:** Todos los predicados deben estar debidamente comentados. Indicando descripción del predicado, tipo de algoritmo/estrategia empleado (ej: fuerza bruta, backtracking, si aplica) argumentos de entrada (dominio) y argumentos de salida (recorrido).
6. **(1 pts) Organización:** Estructurar su código en archivos independientes. Un archivo para cada TDA implementado y uno para el programa principal donde se dispongan sólo los predicados requeridos en el apartado de requerimientos funcionales. Debe usar el predicado “module” y “use_module”.
7. **(2.5 pts) Historial:** Historial de trabajo en Github tomando en consideración la evolución en el desarrollo de su proyecto en distintas etapas. Se requieren **al menos 10 commits** distribuidos en un periodo de tiempo **mayor o igual a 2 semanas (no espere a terminar la materia para empezar a trabajar en el laboratorio. Puede hacer pequeños incrementos conforme avance el curso)**. Los criterios que se consideran en la evaluación de este ítem son: fecha primer commit, fecha último commit, total commits y máximo de commits diarios. A modo de ejemplo (y solo como una referencia), si hace todos los commits el día antes de la entrega del proyecto, este ítem tendrá 0 pts. De manera similar, si hace dos commits dos semanas antes de la entrega final y el resto los concentra en los últimos dos días, tendrá una evaluación del 25% para este ítem (0.375 pts). Por el contrario, si demuestra constancia en los commits (con aportes claros entre uno y otro) a lo largo del periodo evaluado, este ítem será evaluado con el total del puntaje.
8. **(obligatorio) Script de pruebas (pruebas_RUT_Apellidos.pl):** Incluir como parte de su entregable un archivo independiente al código donde muestre de forma completa, con la documentación correspondiente, el funcionamiento de su programa. Este archivo será similar al script de prueba proporcionado al final de este documento. Este archivo debe incluir los ejemplos provistos en el script de prueba de este enunciado además de 3 ejemplos por cada una de las funciones requeridas.
Solo se revisarán proyectos que incluyan este archivo.
9. **(obligatorio) Prerrequisitos:** Para cada funcionalidad solicitada se establecen prerrequisitos. Estos deben ser cumplidos para que se proceda con la evaluación del predicado implementado. Ej: Para evaluar el predicado “login”, debe estar implementado el predicado “register”.

Requerimientos Funcionales. Para que el requerimiento sea evaluado, DEBE cumplir con el prerequisite de evaluación y requisito de implementación. En caso contrario la función no será evaluada. El total de requerimientos permiten alcanzar una nota mayor que 7.0, por lo que procura realizar las funciones que consideres necesarias para alcanzar un 7.0. Si realizas todas las funciones y obtienes el puntaje máximo, la nota asignada será igualmente un 7.0. El puntaje de desborde se descarta.

1. **(0.5 pts TDAs).** Especificar e implementar abstracciones apropiadas para el problema. Recomendamos leer el enunciado completo y ver el material complementario presentado al comienzo de este enunciado a fin de que analice el problema y determine el o los TDAs y representaciones apropiadas para la implementación de cada uno. Luego, planifique bien su enfoque de solución de manera que los TDAs y representaciones escogidos sean aplicables a ambos tipos de funciones.

Para la implementación debe regirse por la estructura de especificación e implementación de TDA vista en clases: Representación, Constructores, Funciones/predicados de Pertenencia, Selectores, Modificadores y Otros predicados. Procurar hacer un uso adecuado de esta estructura a fin de no afectar la eficiencia de sus predicados. En el resto de los predicados se debe hacer un uso adecuado de la implementación del TDA (ej: usar selectores, modificadores, constructores, según sea el caso. No basta con implementar un TDA y luego NO hacer uso del mismo). **Solo implementar los predicados estrictamente necesarios dentro de esta estructura.**

A modo de ejemplo, si usa una representación basada en listas para implementar un TDA, procure especificar e implementar predicados específicos para selectores (realice implementaciones o establezca sinónimos con nombres que resulten apropiados para el TDA).

Dejar claramente documentado con comentarios en el código aquello que corresponde a la estructura base del TDA. Estructura bases que deberá considerar para el resto de las funciones corresponden a "image", "pixbit-d", "pixrgb-d" y "pixhex-d" que constituyen elementos centrales sobre el que se aplicaran los distintos predicados implementados.

Debe contar además con representaciones complementarias para otros elementos que considere relevantes para abordar el problema.

Especificar representación de manera clara para cada TDA implementado (en el informe y en el código a través de comentarios). Luego implementar constructores, funciones de pertenencia, y según se requiera selectores, modificadores y otros predicados que pueda requerir para los otros predicados requeridos a continuación.

Los predicados especificados e implementados en este apartado son complementarios (de apoyo) a los predicados específicos de los dos TDAs que se señalan a continuación. Su desarrollo puede involucrar otros TDAs y tantos predicados como sean necesarias para abordar los requerimientos.

2. **(0.6 pts) TDA image - constructor.** Predicado constructor de imágenes con bitmaps, hexmaps o pixmaps que incluye información de profundidad en cada pixel.

Nombre predicado	image
Prerrequisitos para evaluación	Haber especificado los TDAs según lo señalado en el requerimiento 1.
Requisitos de implementación	- Usar estructuras basadas en listas
Dominio	<p>Width (int) X Height (int) X [pixbit-d* pixrgb-d* pixhex-d*] X image</p> <ul style="list-style-type: none"> • Width: Corresponde al ancho de la imagen • Height: Corresponde al alto de la imagen <p>Para el tercer parámetro, el usuario puede especificar 0 o muchos (*) pixbit-d, pixrgb o pixhex-d de manera homogénea (no los puede alternar).</p>
Constructores requeridos para los otros TDAs	<p>pixbit-d <- x (int) X y (int) X bit ([0 1]) X depth (int))</p> <p>pixrgb-d <- x (int) X y (int) X r (C) X g (C) X b(C) X d (int)</p> <p>pixhex-d <- x (int) X y (int) X hex(String) X d (int)</p> <p>con C cubriendo valores entre 0 y 255</p>
Ejemplo de uso	<p>;imagen de 2 X 2</p> <p>pixrgb-d(0, 0, 10, 10, 10, 10, P1), pixrgb-d(0, 1, 20, 20, 20, 20, P2), pixrgb-d(1, 0, 30, 30, 30, 30, P3), pixrgb-d(1, 1, 40, 40, 40, 40, P4), image(2, 2, [P1, P2, P3, P4], I).</p> <p>;imagen de 2 X 2</p> <p>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I).</p>

3. (0.1 pts) **TDA image - bitmap?**: Predicado que permite determinar si la imagen corresponde a un bitmap-d.

Nombre predicado	image->bitmap?
Prerrequisitos para evaluación	constructor image
Requisitos de implementación	Implementación declarativa
Dominio	<i>image</i>
Recorrido	<i>boolean</i>
Ejemplo de uso	<pre>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->bitmap?(I). ;result: #t</pre>

4. (0.1 pts) **TDA image - pixmap?**: Predicado que permite determinar si la imagen corresponde a un pixmap-d.

Nombre predicado	image->pixmap?
Prerrequisitos para evaluación	constructor image
Requisitos de implementación	Implementación declarativa
Dominio	<i>image</i>
Recorrido	<i>boolean</i>
Ejemplo de uso	<pre>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->pixmap?(I). ;result: #f</pre>

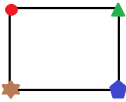
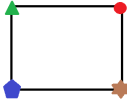
5. **(0.1 pts) TDA image - hexmap?:** Predicado que permite determinar si la imagen corresponde a un hexmap-d.

Nombre función	image->hexmap?
Prerrequisitos para evaluación	constructor image
Requisitos de implementación	Implementación declarativa
Dominio	<i>image</i>
Recorrido	<i>boolean</i>
Ejemplo de uso	<p>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->hexmap?(I).</p> <p>;result: #f</p>

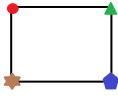
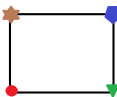
6. **(0.1 pts) TDA image - compressed?:** Predicado que determina si una imagen está comprimida.

Nombre predicado	image->compressed?
Prerrequisitos para evaluación	constructor image
Requisitos de implementación	Implementación declarativa
Dominio	<i>image</i>
Recorrido	<i>boolean</i>
Ejemplo de uso	<p>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->compressed?(I).</p> <p>;result: #f</p>

7. (0.3 pts) TDA image - flipH: Predicado que permite invertir una imagen horizontalmente.

Nombre predicado	image->flipH
Prerrequisitos para evaluación	constructor image
Requisitos de implementación	
Dominio	<i>image X image</i>
Ejemplo de uso	pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->flipH(I , I2).
Ilustración	<div> input:  </div> <div> output:  </div>

8. (0.3 pts) TDA image - flipV: Predicado que permite invertir una imagen verticalmente.

Nombre predicado	image->flipV
Prerrequisitos para evaluación	constructor image
Requisitos de implementación	
Dominio	<i>image X image</i>
Ejemplo de uso	<code>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->flipV(I, I2).</code>
Ilustración	<div><div>input:</div></div> <div><div>output:</div></div>

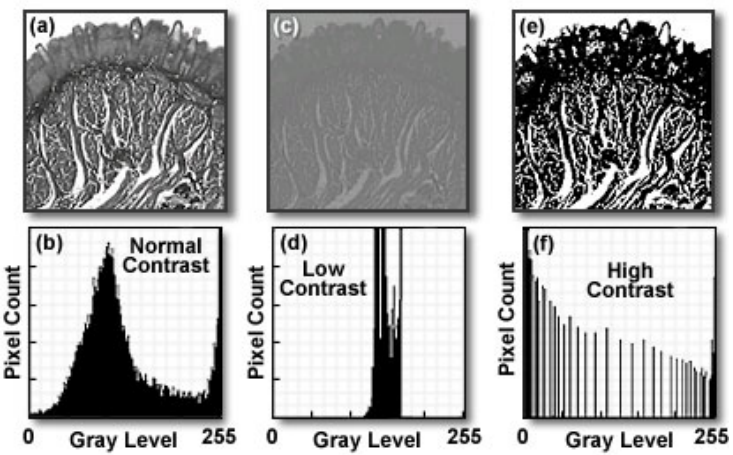
9. (0.2 pts) TDA image - crop: Recortar una imagen a partir de un cuadrante.

Nombre predicado	image->crop
Prerrequisitos para evaluación	image->flipV o image->flipH
Requisitos de implementación	Implementación declarativa
Dominio	$image \times x1 (int) \times y1 (int) \times x2 (int) \times y2 (int) \times image$
Ejemplo de uso	<p>image->crop(lmg1, 10, 10, 40, 40, lmg2).</p> <p>;donde lmg1 es una imagen creada con el constructor image e lmg2 es la imagen cortada resultante</p>
Ilustración	<div> </div>

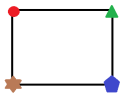
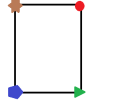
10. (0.5 pts) TDA image - imgRGB->imgHex: Transforma una imagen desde una representación RGB a una representación HEX.

Nombre predicado	image->RGBToHex
Prerrequisitos para evaluación	image->crop
Requisitos de implementación	Implementación principalmente declarativa
Dominio	<i>image X image</i>
Ejemplo de uso	pixrgb-d(0, 0, 10, 10, 10, 10, P1), pixrgb-d(0, 1, 20, 20, 20, 20, P2), pixrgb-d(1, 0, 30, 30, 30, 30, P3), pixrgb-d(1, 1, 40, 40, 40, 40, P4), image(2, 2, P1, P2, P3, P4, I1), (image->RGBToHex(I1, I2).

11. (0,5 pto) **TDA image - histogram**: Retorna un histograma de frecuencias a partir de los colores en cada una de las imágenes. Debe funcionar para bitmap-d, pixmap-d y hexmap-d.

Nombre predicado	image->histogram
Prerrequisitos para evaluación	image->RGBToHex
Requisitos de implementación	Implementación principalmente declarativa
Dominio	<p><i>image X histogram</i></p> <p><i>histogram - queda a criterio de cada estudiante si implementa el TDA histograma o bien, puede retornar una estructura que contenga la información requerida y que le permita abordar la función compress</i></p>
Ejemplo de uso	<p>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), histogram(I, Histograma).</p>
Ilustración	<p>Grayscale Histograms and Contrast Levels in Digital Images</p>  <p style="text-align: center;">Figure 7</p> <p>Fuente: https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/digitalimagebasics/</p>

12. (0,7 pto) TDA image - rotate90: rota la imagen 90° a la derecha.

Nombre predicado	image->rotate90
Prerrequisitos para evaluación	image->histogram
Requisitos de implementación	Implementación principalmente declarativa
Dominio	<i>image X image</i>
Ejemplo de uso	pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->rotate90(I, I2).
Ilustración	<div> input:  </div> <div> output:  </div>

- 13. (0,5 pts) TDA image - compress:** Comprime una imagen eliminando aquellos pixeles con el color más frecuente. La imagen comprimida resultante solo se puede manipular con las otras funciones una vez que haya sido descomprimida a partir de la función señalada más adelante.

Nombre predicado	image->compress
Prerrequisitos para evaluación	image->rotate90 image->histogram
Requisitos de implementación	Implementación declarativa La imagen comprimida debe poder descomprimirse con el predicado de compresión indicado más adelante.
Dominio	<i>image X image</i>
Ejemplo de uso	pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->compress(I, I2).

14. (0,3 pts) TDA image - changePixel: Permite reemplazar un píxel en una imagen.

Nombre predicado	image->changePixel
Prerrequisitos para evaluación	image->rotate90 image->histogram
Requisitos de implementación	Implementación declarativa Puede recibir cualquier tipo de píxel (según el tipo de imagen)
Dominio	<i>image x pixel x image</i>
Ejemplo de uso	pixrgb-d(0, 0, 10, 10, 10, 10, P1), pixrgb-d(0, 1, 20, 20, 20, 20, P2), pixrgb-d(1, 0, 30, 30, 30, 30, P3), pixrgb-d(1, 1, 40, 40, 40, 40, P4), image(2, 2, P1, P2, P3, P4, I1), pixrgb-d(0, 1, 54, 54, 54, 20, P2_modificado), image->changePixel(I, P2_modificado, I2).

- 15. (0,1 pts) TDA image - invertColorRGB:** Predicado que permite obtener el color simétricamente opuesto en cada canal dentro de un pixel.

Nombre predicado	image->invertColorRGB
Prerrequisitos para evaluación	image->rotate90 image->histogram
Requisitos de implementación	Implementación declarativa
Dominio	<i>pixrgb-d</i>
Recorrido	<i>pixrgb-d</i>
Ejemplo de uso	<p>pixrgb-d(0, 0, 10, 10, 10, 10, P1), pixrgb-d(0, 1, 20, 20, 20, 20, P2), pixrgb-d(1, 0, 30, 30, 30, 30, P3), pixrgb-d(1, 1, 40, 40, 40, 40, P4), image(2, 2, P1, P2, P3, P4, I1), invertColorRGB(P2, P2_modificado), image->changePixel(I, P2_modificado, I2).</p> <p>;ejemplos de cambios de color: 0 -> 255 ; 254 -> 1 ; 2->253</p>

- 16. (0,5 pts) TDA image** - image->string: Predicado que transforma una imagen a una representación string. La transformación depende de si la imagen es bitmap-d, hexmap-d o pixmap-d, en caso de que sus pixeles sean *pixbit* entonces debe visualizarse como 0 o 1, en caso de ser *pixrgb* debe mostrarlo como sus 4 valores RGB y en caso de ser *pixhex* debe mostrarse como su valor en hexadecimal.

Nombre predicado	image->string
Prerrequisitos para evaluación	image->changePixel y al menos una de las siguientes: image->rotate90, image->invertColorRGB
Requisitos de implementación	Implementación declarativa Procure usar \n \t y otros en la construcción del string, para dar formato no usar display. La el predicado debe generar un string el cual puede después ser pasado como argumento al predicado display
Dominio	<i>image X string</i>
Ejemplo de uso	pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->string(I, Str), display(Str).

- 17. (1 pts) TDA image - depthLayers:** Predicado que permite separar una imagen en capas en base a la profundidad en que se sitúan los píxeles. El resultado consiste en una lista de imágenes donde cada una agrupa los píxeles que se sitúan en el mismo nivel de profundidad. Además, en las imágenes resultantes se sustituyen los píxeles que se encuentran en otro nivel de profundidad por píxeles blancos (255,255,255).

Nombre predicado	image->depthLayers
Prerrequisitos para evaluación	image->string
Requisitos de implementación	Implementación declarativa
Dominio	<i>image x image list</i>
Ejemplo de uso	<p>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 10, PD), image(2, 2, [PA, PB, PC, PD], I), depthLayers (I, LI).</p> <p>;en este caso en LI queda una lista conteniendo 3 imágenes de 2 x 2 cada una, pues todos los píxeles están en profundidades diferentes excepto el último que tiene la misma profundidad que el primer píxel.</p>

18. (1 pts) **TDA image - decompress**: Predicado que permite descomprimir una imagen comprimida. Para esto se toma como referencia el color más frecuente a fin de reconstruir todos los píxeles que fueron eliminados en la compresión.

Nombre predicado	image->decompress
Prerrequisitos para evaluación	image->string
Requisitos de implementación	Implementación declarativa
Dominio	<i>image x image</i>
Ejemplo de uso	<p>pixbit-d(0, 0, 1, 10, PA), pixbit-d(0, 1, 0, 20, PB), pixbit-d(1, 0, 0, 30, PC), pixbit-d(1, 1, 1, 4, PD), image(2, 2, [PA, PB, PC, PD], I), image->compress(I, I2), image->decompress(I2, I3).</p> <p>;en este caso I3 sería la imagen original "I".</p>

Script básico Pruebas

El código presentado a continuación contiene ejemplos que le permitirán probar todas las funciones. No obstante, estas funciones no cubren todos los escenarios posibles. Estos ejemplos le servirán como referencia para su autoevaluación, sin embargo se recomienda que pueda variar los ejemplos y probar distintos escenarios. **Recuerde que en su entrega final debe ampliar este script de pruebas con al menos 3 ejemplos más por función. Esto quiere decir, que su script de prueba (archivo principal) debe contener todos los ejemplos listados a continuación (sin cambios) además de los suyos.**

Las definiciones de imágenes enumeradas de 1 hasta N corresponden a funciones constantes. En este caso se aplican para hacer el código más legible evitando una composición directa de las funciones, lo que hace que cada una de las evaluaciones de funciones resulte más confusa.

;img1
19.