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

“Laboratorio 2: Editor de imágenes con Prolog”

Catalina Jofré León.

Profesor: Miguel Truffa.

Fecha: 2 de noviembre de 2022.

TABLA DE CONTENIDOS

1- INTRODUCCIÓN

En el presente informe se explicará detalladamente el segundo laboratorio del semestre de la materia de Paradigmas de Programación, además de la solución planteada e implementada, y su paradigma asociado. Este proyecto trata de la aplicación del paradigma lógico en el lenguaje de programación Prolog, utilizando la plataforma online o la aplicación de escritorio, ambas obtenidas de la pagina web "https://www.swi-prolog.org", para la creación de un programa de edición de imágenes que permite diversas operaciones en estas; por esto mismo es necesario una buena comprensión de la problemática y de un adecuado uso de las herramientas entregadas, tales como paradigma lógico, abstracciones del problema, el uso de Tipos de Dato Abstracto o TDA por sus siglas en ingles y el uso de recursión.

El desarrollo de este informe consta de una breve explicación sobre el paradigma lógico, para luego tener el primer acercamiento al proyecto con una contextualización del problema a resolver, junto con un análisis de este y para comprender razonamiento utilizado para la solución y sus fundamentos, todo esto enlazado en la siguiente parte donde detalladamente se explicará la implementación aplicada y como el paradigma fue fundamental en la abstracción del problema; teniendo lista la solución se encuentra un instructivo de uso del programa además de los resultados y finalizando con una conclusión que representa un análisis total de la realización del objetivo y una comparación con el paradigma del laboratorio anterior.

Como principal objetivo de este trabajo se encuentra:

- Aplicar conceptos del paradigma de programación lógico usando el lenguaje de programación Prolog en la resolución de un problema acotado, el cual es la realización de un programa tipo GIMP simplificado.

2- SOBRE EL PARADIGMA

El paradigma lógico es un tipo de programación declarativa, que permite formalizar hechos del mundo real, derivado de los fundamentos de la teoría de la lógica proposicional, de la cual se desprenden las "Clausulas de Horn", que son predicados con una sola conclusión por clausula, además de un conjunto de premisas que derivan en la veracidad de la conclusión, la cual es "verdad" si todas sus premisas lo son. Un programa con este paradigma se forma de diversos predicados definidos por clausulas, que forman una base de conocimiento sobre la que se hacen consultas por consola, esta base se forma de hechos, que representan la información expresada como relaciones entre datos, y por reglas lógicas que permiten deducir consecuencias a partir de combinaciones entre los hechos y, en general, otras reglas. Durante la ejecución de un programa se van evaluando y combinando todas las reglas lógicas de la base de conocimiento para lograr los resultados esperados, en este caso se utiliza el mecanismo de evaluación llamado "backtracking", es decir, encontrar soluciones a problemas que tienen una solución completa, en los que el orden de los elementos no importa, y se tienen que satisfacer restricciones, haciendo uso de listas y recursividad; hay dos tipos de metas, las primarias las que son consultadas directamente por el intérprete en la base de conocimiento y las secundarias, que son las que se consultan de forma interna por las metas primarias; en caso de que no se encuentre el resultado esperado en esta consulta se retorna un "false" como respuesta, lo que indica que no se ha encontrado un hecho definido que sea capaz de satisfacer esta pregunta, enlazado a esto, también puede afectar en el resultado final la falta de uso de " , " (comas), al utilizar predicados de forma continua, y la falta de " ." (punto), que indica el termino del predicado o el hecho, además del uso de mayúsculas en los argumentos de entrada y salida.

3- DESCRIPCION DEL PROBLEMA

Como se mencionó anteriormente, el objetivo y problema a solucionar es realizar un programa de edición de imágenes, como lo son GIMP y Adobe Photoshop, pero con un enfoque simplificado, este programa debe realizar diversas operaciones en una imagen como rotar, invertir, recortar, comprimir, obtener un histograma de los colores y cambiar color de pixeles definidos, entre otras. Este proyecto se concentrará principalmente en el manejo de imágenes del tipo "RGB-D", es decir, imágenes con colores y profundidad, las letras hacen referencia a "RED", "GREEN", "BLUE" y "DEPTH", siendo esta ultima la que contiene la información de la profundidad en un espacio tridimensional, con los datos de estos 4 elementos más otros dos que son el alto y ancho se puede conformar una imagen completamente. Se tienen tres tipos de imágenes a considerar, las cuales son:

- **Bitmap-d:** Imágenes en blanco y negro, cuya unidad principal es el pixbit compuesto por un 1 que llevado a los colores es blanco o el 0 que corresponde al negro.
- **Hexmap-d:** Imágenes de colores pero con representación de 6 valores hexadecimales, su unidad principal es el pixhex, un ejemplo de representación hexadecimal es "#FF00FF" que es fucsia.
- **Pixmap-d:** Al igual que el conjunto de pixeles anterior, el pixmap-d corresponde a una imagen con colores, pero con la diferencia de que estos se representan de la forma RGB o "Rojo", "Verde" y "Azul" en un pixbit, unidad principal de pixmap, un ejemplo de color escrito en RGB es el color fucsia (255 0 255).

Teniendo en consideración el ancho de la imagen más el alto, y estos tres tipos de pixeles se puede trabajar completamente en la modificación de una imagen aplicando las diversas operaciones mencionadas.

4- ANALISIS DEL PROBLEMA

Principalmente la problemática consta de la manipulación de imágenes aplicando las mismas operaciones en todas sin importar el tipo de imagen que sea, por lo que las implementaciones deben ser universales o tener claramente diferenciadas las condiciones para aplicar la operación según el tipo de imagen, cabe recalcar que cuando se habla de "imagen" se habla de la información que contiene una, es decir, alto, ancho y la lista de pixeles, es decir una representación simplificada de una imagen, por lo mismo todas las consultas de los predicados se realizaran directamente en la consola de Prolog, usando solo la biblioteca interna y sin hacer uso de variables. Antes de pensar en las intervenciones a realizar en la fotografía, hay que conocer como es la representación dada a esta problemática, según el tipo de imagen:

- **Imagen:** Lista compuesta de tres elementos principales, ancho y alto de la imagen, y una lista de pixeles, todo esto representado de la forma "Width(int) Height(int) [pixbit-d|pixrgb-d|pixhex-d]", esta construcción es posible solo si los pixeles están implementados:
  - **Pixbit-d:** Lista constituida de cuatro números enteros presentados como "X(int) Y(int) Bit([0|1]) Depth(int)", lo que demostrado en un ejemplo seria "pixbit(0, 2, 1, 12,P), esto significa un pixel ubicado en la posición (0,2) de color blanco y de profundidad 12.
  - **Pixhex-d:** Lista formada por cuatro valores donde tres son enteros y el que esta ubicado en la tercera posición corresponde a un string, la representación dada en Prolog es la siguiente, "X(int) Y(int) Hex(String) Depth(int)", un ejemplo de esto sería, "pixhex(3, 5, "#FFFF00", 34,P)", es decir, un pixel ubicado en la posición (3,5) de color amarillo con profundidad 34.
  - **Pixrgb-d:** Construido en Scheme como una lista de seis elementos enteros, mostrado como "X(int) Y(int) R(C) G(C) B(C) Depth(int)", donde los elementos ubicados en la posición tres, cuatro y cinco corresponden a los colores rojo, verde y azul respectivamente y la "C" que los acompaña significa el intervalo numérico que puede tomar este valor, que va desde 0 a 255, como ejemplo se tiene 'pixrgb(3, 6, 255, 0, 255 ,65, P)", un pixbit de posición (3,6) de color fucsia y profundidad 65.

Con esto claro, se puede proceder con las operaciones y la realización de los TDA que se deben realizar para cumplir a cabalidad con el laboratorio:

- Image: Constructor de la imagen, con ancho, alto y lista de pixeles.
- imageIsBitmap: Predicado que ve si una imagen es bitmap-d.
- imageIsHexmap: Predicado que ve si una imagen es hexmap-d.
- imageIsPixmap: Predicado que ve si una imagen es pixmap-d.
- imageIsCompressed: Predicado que ve si una imagen esta comprimida.
- imageFlipH: Permite invertir la imagen horizontalmente.
- imageFlipV: Permite invertir la imagen verticalmente.
- imageCrop: Corta la imagen en un cuadrante definido.
- imageRGBToHex: Transformar la imagen desde una Pixmap a una Hexmap.
- imageToHistogram: Analiza la cantidad de veces que se repite cada color en la imagen.
- imageRotate90: Rota la imagen 90° a la derecha.
- imageCompress: Comprime la imagen eliminando los pixeles que tienen el color más frecuente.
- imageChangePixel: Predicado que permite cambiar un pixel de la imagen a elección.
- imageInvertColorRGB: Cambia el color de la imagen RGB por el color simétricamente opuesto de este.
- imageToSstring: Transforma una imagen a una representación tipo string.
- imageDepthLayers: Permite la separación de capas de profundidad, dando de resultado una lista de imágenes de diversas profundidades cuyas imágenes se sustituyen los pixeles de otro nivel por color blanco.
- imageDecompress: Descomprime una imagen comprimida.

5- DISEÑO DE LA SOLUCIÓN

5.1 CREACION DE LOS TDA.

Comenzando con el proceso de la solución lo primero que se debe realizar, es la creación de los Tipos de Datos Abstractos a utilizar para las operaciones solicitadas, cuya estructura es la siguiente, Representación, Constructores, Funciones de Pertenencia, Selectores, Modificadores y otras funciones asociadas al TDA.

5.1.1 TDA IMAGE

TDA fundamental del laboratorio, pues todas las operaciones son aplicadas en las imágenes de esta implementación. Su representación es una lista formada por ancho y alto y otra lista de pixeles, lo que reproduce una imagen; el constructor recibe “Ancho”, “Alto” y una lista con los pixeles previamente definidos, y así formar una lista de salida que contiene estos elementos para su futuro uso. Además cuenta con las otras estructuras del TDA.

5.1.2 TDA PIXBIT

El TDA PIXBIT representa la formación de pixeles cuyo color puede tomar valores entre 1 y 0, además incluye la posición de cada pixel, representado con las letras "X" e "X" y la profundidad de estos, es la unidad fundamental de una imagen llamada bitmap-d, su constructor produce de una lista que recibe las posiciones X e Y, un Bit y la Profundidad. Cuenta también con las otras estructuras de un TDA.

5.1.3 TDA PIXHEX

El TDA PIXHEX representa la unidad inicial de una imagen HEXMAP, cuyos colores se presentan con un valor escrito en hexadecimal, e incluye la posición de cada pixel, representado con las letras "X" e "Y", y la profundidad de estos, el constructor de este TDA recibe cuatro elementos, X, X, un string que representa el color, Hex, y la Profundidad. Además cuenta con las estructuras propias de un TDA.

5.1.4 TDA PIXRGB

El TDA PIXRGB representa la unidad inicial de una imagen PIXMAP expresada con pixeles que toman valores del espectro RGB (Red=Rojo, Green=Verde, Blue=Azul), para representar su color, estos valores se mueven entre el 0 y el 255, incluye la posición de cada pixel, representado con las letras "X" e "Y", y la profundidad de estos, su constructor recibe seis elementos en el siguiente orden, posición X, posición Y, color Rojo, color Verde, color Azul y Profundidad. Además cuenta con las estructuras propias de un TDA.

Para poder resolver gran cantidad de predicados de una forma eficiente, se trabajó de lo particular a lo general, aplicando la siguiente técnica:

1- Se toma un pixel de la lista de pixeles y con este se realizan las operaciones necesarias para que el predicado sea capaz de entregar la solución correcta.
2- Teniendo listo como se aplicaría la solución en un solo pixel se procede a aplicarlo en el resto de la lista de pixeles a través de uno o más predicado que funcionan de forma recursiva.
3- En el predicado donde se recibe la imagen, se realiza el llamado del constructor “image” y así poder obtener la información relevante para la solución, se pueden utilizar el ancho, alto y la lista de pixeles, al mismo tiempo o solo alguno de ellos, luego de este llamado, se proceden a aplicar los predicados antes creados, para obtener la lista de pixeles modificada, y finalmente se vuelve a llamar al predicado “image”, para crear la imagen final con el ancho, alto y esta nueva información obtenida.

Un comentario importante respecto al planteamiento de las soluciones fue analizar el comportamiento de los pixeles de forma matemática, y así poder buscar soluciones eficientes. Para comprender el funcionamiento de la recursión en Prolog se estudió una pagina que tenía 99 ejercicios distintos utilizando este lenguaje, su enlace respectivo se encuentra en la bibliografía.

6- ASPECTOS DE IMPLEMENTACION

En este proyecto se utilizó el programa “SWI-Prolog” versión 8.4.3, en su versión de escritorio para comprobar el funcionamiento de los predicados y del script de pruebas, ya que ese sería el formato oficial; para la creación de los predicados se utilizó la versión online de Prolog, encontrado en “https://swish.swi-prolog.org”, cuyo uso se dio en el navegador “Opera GX” .

Cada archivo de los TDA implementados, tiene una estructura de 2 partes, la primera parte abarca los predicados auxiliares o aquellas que son usadas para completar algunos predicados, y los hechos, y la siguiente parte son los predicados pedidos en el laboratorio, se decidió por esta estructura debido a que no debe haber un archivo tipo “main” y cada predicado debe ir con su TDA asociado. Es importante mencionar que los archivos al inicio tienen el predicado ”module (nombre_archivo,[predicado/aridad])” para poder compartir su interior con el resto de los archivos, el archivo del TDA image y el script tienen el predicado “use_module(nombre_archivo)” para poder importar los predicados correspondientes. Son un total de 5 archivos: “TDA_image_20244128_JofreLeon.pl”, “pruebas_20244128_JofreLeon.pl”, “TDA_pixbit_20244128_JofreLeon.pl”, “TDA_pixhex_20244128_JofreLeon.pl” y “TDA_pixrgb_20244128_JofreLeon.pl”.

7- INSTRUCCIONES DE USO

El primer paso, y el más importante, corresponde a la verificación de que los 5 archivos anteriormente nombrados se encuentren en la misma carpeta, de lo contrario no podrá funcionar el script de pruebas, con esto verificado, se procede a realizar las consultas encontradas en el archivo “pruebas_202441289_JofreLeon.pl”, que es donde se encuentran los ejemplos de cada predicado solicitado, para esto es necesario abrir el programa “SWI-Prolog”, y escribir el siguiente comando para poder visualizar el resultado a plenitud, “set_prolog_flag(answer_write_options,[max_depth(0)]).” ,ya que muchas veces las listas resultantes son simplificadas y no se puede ver completamente su interior; luego de esto se procede a apretar el apartado donde

dice “file”, luego “consult” y seleccionar el archivo mencionado desde la ventana emergente, una vez ejecutado se puede proceder a realizar los ejemplos de las operaciones, que están escritas en este mismo archivo, dependiendo del largo estas pueden estar comentadas por un símbolo “ % “ o entre “/* */”, se deben pegar y ejecutar desde la consola procurando que no haya alguno de los símbolos anteriores, en el anexo se encuentran imágenes demostrativas.

8- RESULTADOS OBTENIDOS

Los resultados obtenidos fueron satisfactorios, ya que se pudo completar todo lo solicitado en el laboratorio. El programa funciona bien excepto en algunos casos donde las condiciones de las funciones no previeron situaciones específicas que se observaron al aplicar los ejemplos del script de prueba. Un inconveniente notorio fue encontrado en “imageCrop”, ya que en algunos casos toma pixeles que no se encuentran entre los parámetros definidos, otro inconveniente se encuentra en “imageDepthLayers”, ya que al aplicar la separación de las imágenes e incluir el resto de los pixeles blancos, se producía una duplica de pixeles que se encontraban en la misma posición, esto dado en la situación donde había una imagen con profundidades repetidas, en el caso donde cada profundidad era distinta no se produce este error, ya que se eliminó previamente el pixel correspondiente de la posición (0,0), que es el lugar que toma el pixel con color, también al cambiar un pixel con “imageChangePixel”, este queda en la última posición porque no los ordena luego del cambio. En el anexo se puede ver una tabla con comentarios.

9- CONCLUSIONES

Para finalizar con este proyecto se puede concluir que el objetivo se cumplió completamente, ya que se pudo aplicar los conceptos del paradigma lógico haciendo uso del lenguaje Prolog en la resolución de un problema, en este caso varios predicados que es servían para modificar y analizar imágenes de manera simplificada, esto logró que la materia respecto este paradigma fue aprendida e interiorizada de mejor manera, ya que durante la realización de este trabajo la misma materia que fue aprendida en las clases anteriores como él TDA y las materia de las clases más actuales relacionadas con Prolog y la recursión fueron aplicadas y puestas en práctica durante la realización de este trabajo. Se podría decir que este laboratorio no tuvo tantas complicaciones como el anterior ya que el pensamiento aplicado que fue utilizado para la versión pasada, constaba de un punto de vista matemático que pudo adaptarse e incluso mejorarse en esta ocasión. Se espera que los conocimientos obtenidos con Prolog y el paradigma lógico puedan ser usados en el futuro, ya que esta directamente conectado con la inteligencia artificial, algo que es muy investigado el día de hoy.

Algo que demostró lo diferente que es este paradigma respecto al paradigma funcional, es que es más fácil de usar y de implementar con distintos propósitos, desde la recursión que su implementación no requiere tantas condiciones, acceder a un elemento especifico de una lista de listas se puede hacer simplemente con poner guiones bajo en los elementos que no importan, por decirlo de una forma simple, también existe el concepto de variable que lo hace mas amigable respecto a Scheme, ya que es algo bastante conocido y manejado en diversos cursos de la carrera. Respecto a la plataforma Git se encontró bastante útil ya que sirve para guardar el avance sin perder información relevante que pudo haber sido borrada sin por error o que se borró porque no se creía importante, una ocasión donde su uso fue más allá que hacer commit por la nota final, fue cuando se hacia la documentación, 3 predicados fueron eliminados sin querer, pero fueron encontrados en el historial de registros de git.

10- BIBLIOGRAFIA Y REFERENCIAS

Documentation. (s. f.). SWI Prolog. Recuperado 31 de octubre de 2022, de https://eu.swi-prolog.org/pldoc/doc_for?object=root

Facts, rules, goals and queries. (s. f.). Recuperado 1 de noviembre de 2022, de http://www.ablmcc.edu.hk/%7Escy/prolog/pro02.htm

Marker, G. (2022, 16 febrero). *Que es programación lógica?* Tecnología + Informática. https://www.tecnologia-informatica.com/que-es-programacion-logica/

Merino, M. (2020, 9 agosto). *El lenguaje Prolog: un ejemplo del paradigma de programación lógica*. Genbeta. https://www.genbeta.com/desarrollo/lenguaje-prolog-ejemplo-paradigma-programacion-logica

P-99: Ninety-Nine Prolog Problems. (s. f.). Recuperado 31 de octubre de 2022, de https://www.ic.unicamp.br/%7Emeidanis/courses/mc336/problemas-prolog/

11- ANEXOS

Tabla de funciones.

Predicado	Grado de alcance	Resultados
Image	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageIsBitmap	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageIsHexmap	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageIsPixmap	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageIsCompressed	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageFlipH	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageFlipV	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageCrop:	99%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto, en algunos casos falla.
imageRGBToHex	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageToHistogram	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageRotate90	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageCompress	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageChangePixel	98%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto, pero realiza el cambio del pixel en la ultima posicion.
imageInvertColorRGB	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageToSstring	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.
imageDepthLayers	95%	Los resultados están parcialmente correctos, ya que falta eliminar algunos pixeles repetidos.
imageDecompress	100%	Cada una de las pruebas resulto satisfactoria y se pudo entregar el resultado correcto.

Manual de uso

Ver que los archivos estén todos en la misma carpeta.

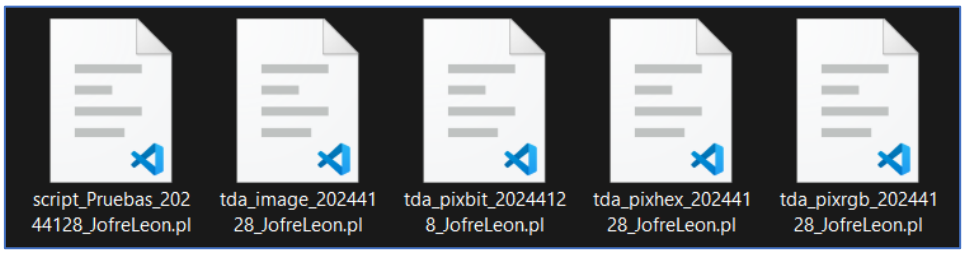

Ilustración 1: Archivos en la misma carpeta.

Abrir el programa SWI-Prolog para realizar las consultas.

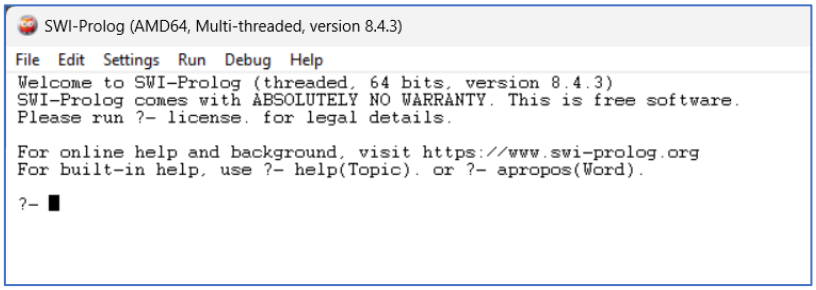

Ilustración 2: Consola SWI-Prolog.

Escribir el comando "set_prolog_flag(answer_write_options,[max_depth(0)]).", para poder ver las listas completas.

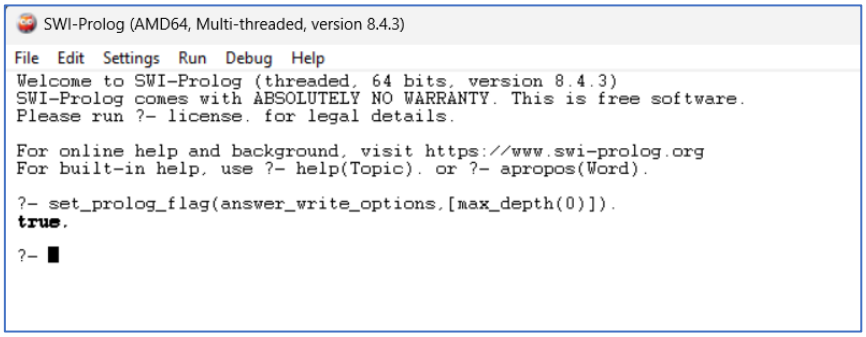

Ilustración 3: Comando a ingresar.

Apretar donde dice file y luego consult.

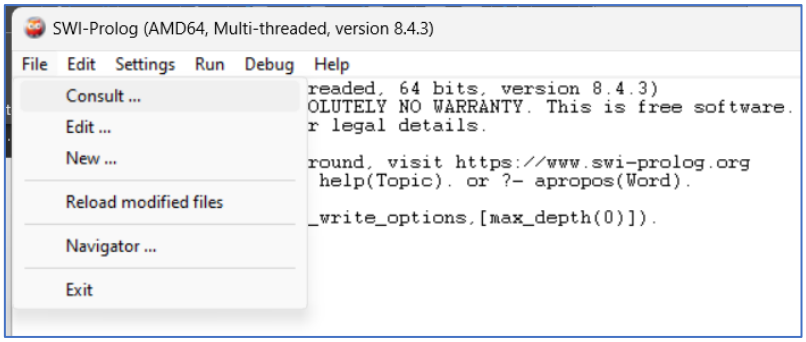

Ilustración 4: Como consultar un archivo.

Cuando se abra la ventana emergente elegir el archivo que dice, “pruebas_202441289_JofreLeon.pl”, y abrir.

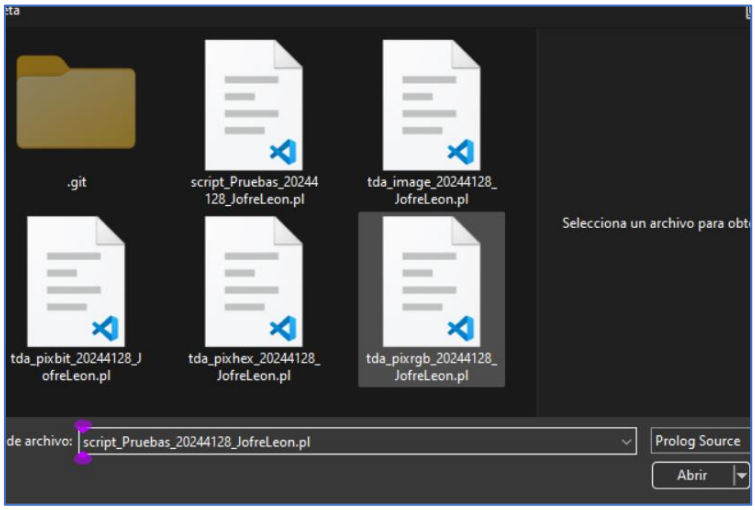

Ilustración 5: Archivo a elegir cuando se abra la ventana emergente.

Luego de esto se puede comenzar a hacer las consultas, copiando desde el archivo de pruebas y pegando en consola.

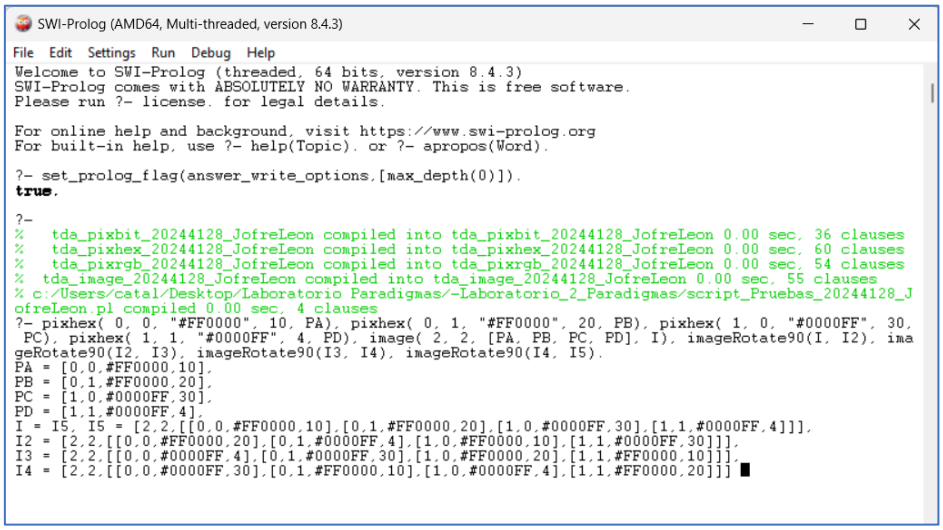

Ilustración 6: Ejemplo de cómo realizar una consulta en la consola.

Hay dos formas de comentarios, una que comienza con “%” y otra que esta entre “/* */”; sin importar cual sea el tipo de comentario hay que tener cuidado de no seleccionar los símbolos.

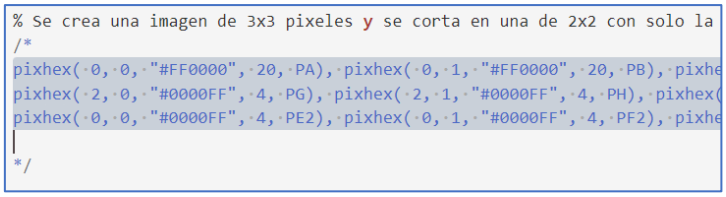

Ilustración 7: Ejemplo comentado entre "/ */"*

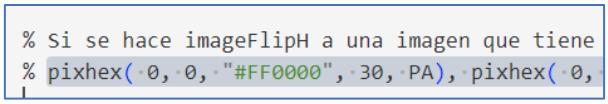

Ilustración 8: Ejemplo comentado con " % ".