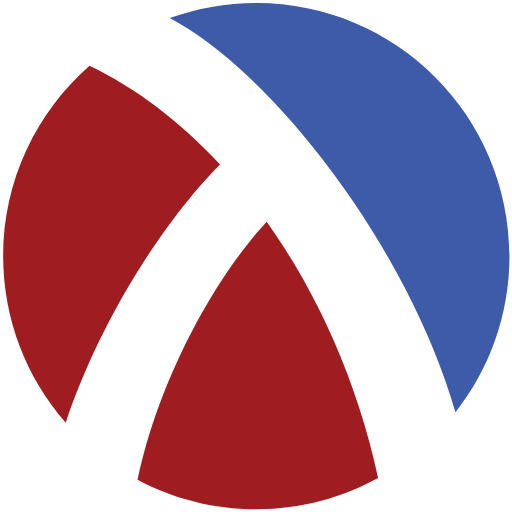
**INFORME DE LABORATORIO 1: SIMULACIÓN DE UN PROGRAMA PARA REALIZAR UN TRATAMIENTO DE IMÁGENES SIMPLIFICADO**

****

Nombre: Aracely Castro V.

Profesor: Roberto Gonzales I.

Asignatura: Paradigmas de Programación (2/2022)

Índice

[1. INTRODUCCIÓN 3](#_Toc115122934)

[**1.1** **DESCRIPCIÓN DEL PROBLEMA** 3](#_Toc115122935)

[**1.2** **DESCRIPCIÓN DEL PARADIGMA** 3](#_Toc115122936)

[2. DESARROLLO 4](#_Toc115122937)

[**2.1** **ANALISIS DEL PROBLEMA** 4](#_Toc115122938)

[**2.2** **DISEÑO DE LA SOLUCIÓN** 5](#_Toc115122939)

[**2.3** **ASPECTOS DE IMPLEMENTACIÓN** 6](#_Toc115122940)

[**2.4** **INSTRUCCIONES DE USO** 6](#_Toc115122941)

[**2.4.1 EJEMPLOS DE USO** 6](#_Toc115122942)

[**2.4.2 RESULTADOS ESPERADOS** 7](#_Toc115122943)

[**2.4.3 POSIBLES ERRORES** 7](#_Toc115122944)

[**2.5** **RESULTADOS Y AUTOEVALUACIÓN** 7](#_Toc115122945)

[**2.5.1** **RESULTADOS** 7](#_Toc115122946)

[**2.5.2** **AUTOEVALUACIÓN** 7](#_Toc115122947)

[3. CONCLUSIÓN 7](#_Toc115122948)

[4. BIBLIOGRAFÍA Y REFERENCIAS 8](#_Toc115122949)

[5. ANEXOS 8](#_Toc115122950)

# **INTRODUCCIÓN**

Con el objetivo de aplicar conceptos del paradigma de programación funcional usando el lenguaje de programación Scheme en la resolución de un problema acotado, en las siguientes secciones se dará detalle sobre el proceso de desarrollo de un simulador de un tratamiento de imágenes simple mediante el uso de un paradigma funcional a través del compilador Dr. Racket versión 8.6. El informe constará de una breve introducción de cómo surgió el problema, una descripción del paradigma utilizado, el análisis del problema y como fue el diseño de las soluciones para algunas funciones, sus aspectos de implementación, las instrucciones necesarias para compilar el archivo con el script de pruebas junto con ejemplos de algunas funciones, los resultados y la autoevaluación. Finalmente, se dará una conclusión respecto a todo lo anterior.

## **DESCRIPCIÓN DEL PROBLEMA**

Tal y como se describe en el laboratorio se busca hacer una simulación de un programa para el tratamiento de imágenes de forma simplificada como GIMP y Adobe Photoshop. Este simulador permitiría crear imágenes con distintos formatos de píxeles sobre los cuales se pueden aplicar distintas operaciones como, por ejemplo, recortar una imagen, invertir una imagen, obtener histograma de la imagen, editar una imagen, entre otros. Para implementar esto, se deben tener en cuenta que:

Una imagen RGBD o Pixmap-d es una imagen donde cada uno de sus píxeles contiene información de espacio de colores y profundidad (R)ed, (G)reen, (B)lue y (D)epth. Cada color cubre valores entre 0 y 255 formando en conjunto un color del espectro RGB. La profundidad, en cambio, ofrece información más detallada de la imagen como espacio tridimensional.

Una imagen Bitmap-d es una imagen donde cada uno de sus píxeles tiene asociado solo un color, negro (1) o blanco (0) representando si hay tinta o no, cada píxel también tendría una profundidad asociada.

Una imagen Hexmap-d es similar a una imagen Pixmap-d con la diferencia que los valores RGB son representados en un string hexadecimal.

## **DESCRIPCIÓN DEL PARADIGMA**

Un paradigma funcional es un enfoque para programar en un computador basado únicamente en funciones. Cada función tiene una entrada (dominio) que es transformado hacia una salida (recorrido). En el paradigma funcional no existe el concepto de almacenar o actualizar variables y utiliza notación lambda con operaciones prefijas. Cabe señalar que el lenguaje para el compilador Dr. Racket no es puramente funcional, pues tiene aspectos de programación imperativa.

Al paradigma funcional le importa más el resultado que el desarrollo, es decir, profundiza en el ¿Qué? y no el ¿Cómo?, lo que implica un mayor nivel de abstracción.  Algunos elementos importantes a la hora de construir un código utilizando el paradigma funcional son:

**Composición de funciones:** Es cuando defines una función que tiene como entrada uno o más funciones distintas.

**Recursividad:** Es un método de solución a problemas que pueden depender de soluciones más pequeñas, la función se puede definir en términos de sí mismo y está conformado por una condición de borde y descomposición recursiva. Existen distintos tipos de recursividad como son: recursión natural, recursión de cola y recursión arbórea.

**Funciones Anónimas:** son funciones que se expresan sin nombre utilizando solamente cálculo lambda, quien recibe una entrada y da un resultado solo una vez.

**Currificación:** Es la evaluación de una función de “n” argumentos que tras currificarse es transformada a la evaluación de una secuencia de funciones de un argumento. En la currificación se ocupan funciones anónimas bajo el concepto de funciones de orden superior.

**Funciones de orden superior:** Corresponden a aquellas funciones que reciben como entrada una función, o bien su salida es para otra función. Una composición de funciones es una función de orden superior.

# **DESARROLLO**

## **ANALISIS DEL PROBLEMA**

Se identifica que para hacer un tratamiento de imágenes simples en Dr. Racket hay que considerar los siguientes elementos fundamentales:

- **Imagen:** Una lista que contiene dos enteros y una lista de píxeles. (Int X Int X List)

- **Pixrgb-d:** Un píxel del tipo Pixmap-d, representado como dos enteros que guardan las posiciones del píxel en la imagen, tres enteros que representan el color del píxel y un entero que guarda la profundidad del píxel en la imagen. (Int X int X int X int X int X int)

- **Pixbit-d**: Un píxel del tipo Bitmap-d, representado como dos enteros que guardan las posiciones del píxel en la imagen, un entero que representa el bit y otro que guarda la profundidad. (Int X Int X Int X Int)

- **Pixhex-d:** Un píxel del tipo Hexmap-d, representado como dos enteros que guardan las posiciones del píxel en la imagen, un string que representa el color y un entero que guarda la profundidad del píxel en la imagen. (Int X Int X String X Int)

Además de las operaciones:

* **image:** Crea de imágenes tipo Bitmap-d, Hexmap-d o Pixmap-d.
* **bitmap?, pixmap? y hexmap?:** Verifica si la imagen es de tipo Bitmap-d, Pixmap-d o Hexmap-d respectivamente
* **compressed?:** Verifica si la imagen sufrió una compresión
* **flipH y flipV:** Invierte una imagen horizontal y verticalmente respectivamente
* **crop:** Recorta una imagen a partir de un cuadrante definido por dos puntos
* **imgRGB->imgHex:** convierte una imagen Pixmap-d a Hexmap-d
* **histogram:** Retorna un histograma de frecuencias a partir de los colores de una imagen
* **rotate90:** rota la imagen 90° a la derecha.
* **compress:** Comprime una imagen eliminando aquellos píxeles con el color más frecuente.
* **edit:** Permite aplicar funciones especiales a las imágenes.
* **adjustChannel:** Función que permite ajustar cualquier canal de una imagen pixmap, incluido el canal de profundidad d.
* **image->string:** Función que transforma una imagen a una representación string.
* **depthLayers:** Función que permite separar una imagen en capas con base a la profundidad en que se sitúan los píxeles.
* **decompress:** Función que permite descomprimir una imagen comprimida.

Se pide que cada TDA (Tipo de Dato Abstracto) se implemente con lo necesario para cumplir con los requerimientos funcionales y que en el archivo principal se contenga las funciones con un script de pruebas para probar las funciones.

## **DISEÑO DE LA SOLUCIÓN**

Por otra parte, se distinguen algunos casos particulares para soluciones de algunos requerimientos funcionales utilizando compilador Dr. Racket como lo son:

Función **histogram:** Para cada TDA se creó su propio histograma para obtener la información relevante. En el caso de Bitmap-d se implementó una lista con dos sublistas donde se guarden la cantidad de veces que se repite el 0 y el 1 de la imagen. En el caso de Pixmap-d y Hexmap-d, se tuvo que definir dos funciones envueltas para cada uno, uno para eliminar un valor repetido del formato de píxeles (usando recursión natural) y otro para contar las veces que se repiten los elementos en el formato (usando recursión cola), esto con el objetivo de recopilar en una lista de sublistas cada elemento con la cantidad de veces que se repite en la imagen. Cabe señalar que, si la entrada es una imagen comprimida, esta será descomprimida antes de obtener el histograma. Ver la Figura N°1, Figura N°2 y Figura N°3 en ANEXOS para poder ver la salida de histogram con imágenes Pixmap-d, Bitmap-d y Hexmap-d. P.8.

**Función compress:** Para poder comprimir se pensó en una forma de cambiar la información del píxel más repetido con otro dato distinto, de modo que se pueda descomprimir con la función decompress más tarde. Para el caso de una imagen Bitmap-d se eligió reemplazar el valor del bit más repetido por -1 y entonces si se quería volver a la forma original bastó con comprobar si la imagen comprimida le falta el bit 0 o el bit 1 y reemplazar a la forma original en consecuencia. En el caso de una imagen Hexmap-d se eligió reemplazar el píxel repetido por una lista con tres números, cada número representando en valor color rojo, verde y azul del string, con esto para volver a la imagen original se podría recuperar la lista y con base a ella crear el string original para reemplazarlo donde corresponde. Finalmente, para comprimir una imagen Pixmap-d se optó por reemplazar el valor entero del espectro RGB por su equivalente a string hexadecimal y así, para volver a la forma original, bastaría con leer el string de cada color y transformarlo a número antes de colocarlo donde se necesite. Ver la Figura N°4, Figura N°5 y Figura N°6 en ANEXOS para poder ver la salida de compress con imágenes Pixmap-d, Bitmap-d y Hexmap-d. P.9.

**Función flipH:** Para implementar flipH se creó una función envuelta flipH-formato que modifica las coordenadas <y> de cada píxel de modo que parezca que la imagen se invirtió horizontalmente, utilizando recursión natural se cambia <y> de mayor a menor por cada fila del formato de píxeles. Para lograr aquello se comenzó con preguntar si la coordenada (x, y) que se buscaban cambiar existían en el formato de píxeles (en el caso de que la imagen ingresada fuera un crop) y se volvió a llamar a la función con argumentos distintos dependiendo del caso hasta retornar null cuando no haya más píxeles. La función envuelta retorna el formato de los píxeles invertidos horizontalmente y se ordenan las coordenadas con la función auxiliar ordenar\_formato para, finalmente, ser ingresado en la imagen con la función auxiliar modificar\_formato\_image. Cabe señalar que flipV y rotate90 se implementó un enfoque de solución similar con variaciones en los cambios de coordenadas (x, y).  Ver la Figura N°7 en ANEXOS para poder ver la salida de flipH-formato en una imagen 2x2. P.10.

## **ASPECTOS DE IMPLEMENTACIÓN**

El compilador utilizado fue Dr. Racket versión 8.6, no se utilizaron bibliotecas externas ni se usaron funciones set!, esto para pensar las soluciones a los requerimientos funcionales usando paradigma funcional y utilizando solo las funciones básicas de Dr. Racket. Mientras se desarrollaban los requerimientos funcionales, se crearon los siguientes TDAs cuya estructura respecto el orden de representación, constructor, pertenencia, selectores, modificadores y otras funciones:

**TDA Image:** Corresponde a una imagen. Representación: (int x int x list). Ver Tabla N°1 en ANEXOS para ver su implementación, P.13.

**TDA Pixbit-d:** Corresponde a un píxel bitmap. Representación: (int x int x bit ([0|1]) x int). Ver Tabla N°2 en ANEXOS para ver su implementación, P.14.

**TDA Pixhex-d:** Corresponde a un píxel hexmap. Representación: (int x int x string x int). Ver Tabla N°3 en ANEXOS para ver su implementación, P.14.

**TDA Pixrgb-d:** Corresponde a un píxel pixmap. Representación: (int x int x int x int x int x int). Ver Tabla N°4 en ANEXOS para ver su implementación, P.15.

Ver la Figura N°8 en ANEXOS para poder ver en que archivos se importan otros archivos. P.11.

## **INSTRUCCIONES DE USO**

### **2.4.1 EJEMPLOS DE USO**

Para comenzar, se debe verificar que se tengan todos los archivos TDAs en una misma carpeta, de lo contrario el archivo pruebas\_21090869\_CastroVenegas no se ejecutará al compilar el programa con “Run”. Luego de ejecutado se podrán ver los ejemplos de cada una de las operaciones en el archivo como script de pruebas. Una observación importante es que si se desea crear una nueva imagen para probar las funciones, esta no debe tener el mismo nombre que una de las imágenes ya definidas y que las dimensiones y el número de píxeles en la imagen deben de ser correctos, todo esto para que no surja un error al compilar o llamar una función. También que los argumentos de las funciones estén bien ingresados. Por ejemplo, se podrá aplicar la función inverColorRGB a un píxel Pixmap-d, pero si se desea aplicar a una imagen se tiene que utilizar en conjunto con la función edit.

Ver Figura 9 en ANEXOS para ver ejemplos claros de ejecución de algunas funciones. P.11.

### **2.4.2 RESULTADOS ESPERADOS**

Se espera que se haya hecho un simulador de tratamiento de imágenes simple donde cada función no cause errores y compile el archivo con el script de pruebas correctamente.

### **2.4.3 POSIBLES ERRORES**

Puede que algunas funciones que reciben de entrada una imagen no den una salida correcta al ingresar una imagen que sufrió un crop.

## **RESULTADOS Y AUTOEVALUACIÓN**

### **RESULTADOS**

Las funciones funcionaron correctamente, aunque en medio de las pruebas se notaron algunos detalles como, por ejemplo, que no podías utilizar funciones flipV, flipH y rotate90 en una imagen que sufrió un crop o cuando en la función image->string ingresabas una imagen crop este no respetaba las coordenadas del píxel cuando se creaba el string. Todo lo anterior se modificó para arreglar esos errores.

### **AUTOEVALUACIÓN**

La Autoevaluación se realiza de la siguiente forma: 0: No realizado – 0.25: Funciona 25% de las veces – 0.5: Funciona 50% de las veces 0.75: Funciona 75% de las veces – 1: Funciona 100% de las veces. Para ver la tabla de Autoevaluación, ver la Tabla XX del anexo XX

En los requerimientos funcionales no se encontró errores, por lo que se considera de que funcionan el 100% de las veces.

# **CONCLUSIÓN**

Tras haber trabajado en implementar los requerimientos funcionales para un simulador de tratamiento de imágenes simple, ser puede decir que se cumplió el objetivo de aplicar conceptos del paradigma funcional usando el lenguaje de programación Scheme en la resolución de un problema acotado. Las complicaciones que hubo durante el desarrollo de las funciones fueron, en mayor parte, la necesidad de comprender como idear una solución antes de programarlo en Dr. Racket. También el descubrir como una función no consideraba una entrada de imagen distinta a la normal, como, por ejemplo, una imagen comprimida u otra que paso por un crop, lo que llevaría a tener que editar el código original para aceptar esos casos. Otro pequeño problema fue el progreso que se perdió en GitHup por no guardar correctamente al hacer un commit. Aparte de todo eso, los requerimientos funcionales se lograron implementar sin errores, o al menos no evidentes. Finalmente, se espera que conocimientos aprendidos, como el modo de encontrar una solución a un problema sin utilizar variables o el uso de funciones envueltas y buscar funciones de la biblioteca como se hizo con DR. Racket puedan de ser de utilidad para futuros proyectos, buscando para el próximo laboratorio ser más proactivo.

# **BIBLIOGRAFÍA Y REFERENCIAS**

Gonzales, R. (2022). “Proyecto Semestral de Laboratorio”. Paradigmas de Programación. Enunciado de Proyecto Online. Recuperado de: <https://docs.google.com/document/d/1hUAooKwBj3TWv-yuzBZtNbuaC8iNkzOZdbLpD8P9B8c/edit>

Gonzales, R. (2022). “3 – P. Funcional”. Paradigmas de Programación. Material de clases Online. Recuperado de: <https://uvirtual.usach.cl/moodle/course/view.php?id=10036&section=11>

Flatt, M. y Bruce, R. (2022). “The Racket Guide’’. The Racket Reference. Documentación Online. Recuperado de: <https://docs.racket-lang.org/guide/>

# **ANEXOS**

**Figura1: Función histogram de una imagen 3x3 bitmap**

Texto, Carta

Descripción generada automáticamente

**Figura2: Función histogram de una imagen 4x3 pixmap**

Tabla

Descripción generada automáticamente

**Figura3: Función histogram de una imagen 2x3 hexmap**

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**Figura 4: Función compress de una imagen 3x3 bitmap**

Texto, Tabla

Descripción generada automáticamente

**Figura 5: Función compress de una imagen 2x3 hexmap**

Texto

Descripción generada automáticamente

**Figura 6: Función compress de una imagen 4x3 pixmap**

Tabla

Descripción generada automáticamente

**Figura 7: Como cambia el formato de una imagen 2x2 con flipH-formato**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Figura 8: Diagrama de importación de TDAs**

Diagrama

Descripción generada automáticamente

**Figura 9: Ejemplos de uso de funciones**

Nota: En el script de pruebas se encontrarán ejemplos listos para ejecutar. Las siguientes figuras muestran algunos de estos en la consola.

Crear una imagen – constructor – (int int . pixeles)

Texto

Descripción generada automáticamente con confianza media

Tabla

Descripción generada automáticamente

Texto, Tabla

Descripción generada automáticamente con confianza media

Crear una imagen crop – modificador – (imagen x1 y1 x2 y2)

Texto

Descripción generada automáticamente

Función display image2 y image10 – otras funciones – (imagen función)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Tabla n°1: Funciones del TDA Image**

|  |  |  |
| --- | --- | --- |
| **Tipo de función** | **Nombre** | **Descripción** |
| Constructor | image | Crea una imagen |
| Pertenencia | pixmap? | Verifica si la imagen es Pixmap-d |
| Pertenencia | bitmap? | Verifica si la imagen es Bitmap-d |
| Pertenencia | hexmap? | Verifica si la imagen es Hexmap-d |
| Pertenencia | compress? | Verifica si la imagen sufrió una compresión |
| Pertenencia | encontrar\_pixel? | Verifica si una coordenada (x,y) existe en el formato de pixeles de la imagen |
| Selector | ancho\_image | Recupera el valor x de la imagen |
| Selector | largo\_image | Recupera el valor y de la imagen |
| Selector | largo\_pos\_x | Recupera la última posición x de la imagen |
| Selector | largo\_pos\_y | Recupera la última posición y de la image |
| Selector | encontrar\_pixel | Recupera un píxel del formato de pixeles de la imagen |
| Modificador | ordenar\_formato | Modifica el formato de pixeles según sus coordenadas (x,y) desde (0,0) |
| Modificador | modificar\_formato\_image | Modifica el formato de pixeles de la imagen |
| Modificador | modificar\_posicion\_pixel | Modifica las coordenadas (x,y) de un píxel |
| Modificador | flipH | Modifica la imagen de modo que su formato de pixeles se invierta horizontalmente |
| Modificador | flipV | Modifica la imagen de modo que su formato de pixeles se invierta verticalmente |
| Modificador | rotate90 | Modifica la imagen de modo que su formato de pixeles gire 90° a la derecha |
| Modificador | crop | Modifica la imagen dejando solo los pixeles dentro del cuadrante definido |
| Modificador | adjustChannel | Cambia el canal RGBD de un Pixmap-d |
| Modificador | imgRGB->imgHex | Convierte una imagen Pixmap-d a Hexmap-d |
| Modificador | compress | Comprime una imagen |
| Modificador | decompress | Descomprime una imagen comprimida |
| Modificador | edit | Edita una imagen con una función auxiliar |
| Otras funciones | invertColorRGB | Función auxiliar, invierte el color de un pixrgb-d |
| Otras funciones | invertColorBit | Función auxiliar, invierte el color de un pixbit-d |
| Otras funciones | image->string | Retorna una cadena de string de la imagen |
| Otras funciones | depthLayers | Retorna una lista de imágenes separados por profundidad |
| Otras funciones | incCh | Función auxiliar de adjustChannel, incrementa canal en uno |
| Otras funciones | histogram | Retorna un histograma de frecuencias a partir de los colores de una imagen |

**Tabla n°2: Funciones del TDA Pixbit-d**

|  |  |  |
| --- | --- | --- |
| **Tipo de función** | **Nombre** | **Descripción** |
| Constructor | pixbit-d | Crea un píxel pixbit-d |
| Pertenencia | pixbit-d? | Verifica que el píxel sea pixbit-d |
| Pertenencia | pixbit-d\_compressed? | Verifica si el píxel pixbit-d fue comprimido |
| Pertenencia | bit? | Verifica si el píxel tiene un bit |
| Pertenencia | compressed\_bit? | Verifica si el bit de un pixbit-d está comprimido |
| Selector | x\_bit | Recupera la coordenada x de un pixbit-d |
| Selector | y\_bit | Recupera la coordenada y de un pixbit-d |
| Selector | b\_bit | Recupera el valor bit de un pixbit-d |
| Selector | d\_bit | Recupera el valor profundidad de un pixbit-d |
| Modificador | cambiar\_x\_bit | Modifica la coordenada x de un pixbit-d |
| Modificador | cambiar\_y\_bit | Modifica la coordenada y de un pixbit-d |
| Modificador | cambiar\_b\_bit | Modifica el bit de un pixbit-d |
| Modificador | compress-formato-bit | Modifica el formato de pixeles pixbit-d de modo que este comprimido |
| Modificador | decompress-formato-bit | Modifica el formato de pixeles pixbit-d a su forma original tras ser comprimido |
| Otras funciones | histograma\_bit | Retorna un histograma de frecuencias a partir de bits de un formato de pixeles pixbit-d |
| Otras funciones | pixbit->string | Retorna una cadena de string de un formato de pixeles pixbit-d |

**Tabla n°3: Funciones del TDA Pixhex-d**

|  |  |  |
| --- | --- | --- |
| **Tipo de función** | **Nombre** | **Descripción** |
| Constructor | pixhex-d | Crea un píxel pixhex-d |
| Pertenencia | pixbit-d? | Verifica que el píxel sea pixhex-d |
| Pertenencia | pixhex-d\_compressed? | Verifica si el píxel pixhex-d fue comprimido |
| Pertenencia | compress\_hex? | Verifica si el string de pixhex-d está comprimido |
| Selector | x\_hex | Recupera la coordenada x de un pixhex-d |
| Selector | y\_hex | Recupera la coordenada y de un pixhex-d |
| Selector | hex | Recupera el string de un pixhex-d |
| Selector | d\_hex | Recupera el valor profundidad de un pixhex-d |
| Modificador | cambiar\_x\_hex | Modifica la coordenada x de un pixhex-d |
| Modificador | cambiar\_y\_hex | Modifica la coordenada y de un pixhex-d |
| Modificador | cambiar\_h\_hex | Modifica la representación del hexadecimal de un pixhex-d |
| Modificador | compress-formato-hex | Modifica el formato de pixeles pixhex-d de modo que este comprimido |
| Modificador | decompress-formato-hex | Modifica el formato de pixeles pixhex-d a su forma original tras ser comprimido |
| Otras funciones | histograma\_hex | Retorna un histograma de frecuencias a partir de bits de un formato de pixeles pixhex-d |
| Otras funciones | pixhex->string | Retorna una cadena de string de un formato de pixeles pixhex-d |

**Tabla n°4: Funciones del TDA Pixrgb-d**

|  |  |  |
| --- | --- | --- |
| **Tipo de función** | **Nombre** | **Descripción** |
| Constructor | pixrgb-d | Crea un píxel pixrgb-d |
| Pertenencia | pixrgb-d? | Verifica que el píxel sea pixrgb-d |
| Pertenencia | pixrgb-d\_compressed? | Verifica si el píxel pixrgb-d fue comprimido |
| Pertenencia | color\_igual? | Verifica si el píxel pixrgb-d tiene el mismo color que una lista de colores |
| Pertenencia | compress\_rgb? | Verifica si los colores de un pixrgb-d están comprimidos |
| Selector | x\_rgb | Recupera la coordenada x de un pixrgb-d |
| Selector | y\_rgb | Recupera la coordenada y de un pixrgb-d |
| Selector | getR | Recupera el valor del color rojo de un pixrgb-d |
| Selector | getG | Recupera el valor del color verde de un pixrgb-d |
| Selector | getB | Recuperar el valor del color azul de un pixrgb-d |
| Selector | getD | Recupera el valor profundidad de un pixrgb-d |
| Selector | color\_lista | Recupera en una lista los colores de un pixrgb-d |
| Modificador | cambiar\_x\_rgb | Modifica la coordenada x de un pixrgb-d |
| Modificador | cambiar\_y\_rgb | Modifica la coordenada y de un pixrgb-d |
| Modificador | setR | Modifica el valor del color rojo de un pixrgb-d |
| Modificador | setG | Modifica el valor del color verde de un pixrgb-d |
| Modificador | setB | Modifica el valor del color azul de un pixrgb-d |
| Modificador | setD | Modifica la profundidad de un pixrgb-d |
| Modificador | compress-formato-rgb | Modifica el formato de pixeles pixrgb-d de modo que este comprimido |
| Modificador | decompress-formato-rgb | Modifica el formato de pixeles pixrgb-d a su forma original tras ser comprimido |
| Otras funciones | pixrgb->string | Retorna una cadena de string de un formato de pixeles pixrgb-d |
| Otras funciones | histograma\_rgb | Retorna un histograma de frecuencias a partir de los colores de un formato de pixeles pixrgb-d |