Documento de Diseño - Estructura de Datos

Presentado por:

Juan Esteban Urquijo

Ángel David Talero

Iván Alejandro Martínez

Presentado a:

Andrea Rueda Olarte

Pontificia Universidad Javeriana

Estructura de datos

Bogotá D.C, abril 22 de 2022

Correcciones primera entrega:

Problema	**Corrección**
Para el criterio mediana, no se reconoce el comando, es decir, no se pueden generar proyecciones usando la mediana.	El criterio 'mediana' fue añadido, se calcula un vector con todos los valores del píxel en las coordenadas I, J de cada una de las capas, luego se ordena este vector y se halla el dato de en medio, o el promedio de los datos de en medio en el caso de un vector de elementos pares
Las proyecciones en Y y Z no están quedando bien, parecen como cortadas por la mitad	El algoritmo fue modificado de tal manera que las proyecciones se realicen capa por capa (como en la dirección de las x) sin importar la dirección, para esto, las proyecciones en 'y' y 'z' realizan una rotación de 90° sobre volumen previa a la rotación, esto facilita los cálculos, permite un código más limpio y soluciona el problema en la proyección
En la implementación, la definición de las librerías no es correcta. La cabecera de cada TAD debe estar en un archivo .h	Se cambió la extensión de los archivos por *.h* y se actualizaron los '*header guards'* e *#includes*
El promedio en X está muy bajito, como si se estuviera dividiendo sobre más de lo esperado.	Como se mencionó anteriormente, el algoritmo de proyección fue reescrito desde 0, el promedio se calcula dividiendo por el tamaño de la cola de capas a promediar
En las direcciones 'X' y 'Y', al utilizar el criterio 'mínimo', los valores no son los esperados.	Todas aquellas imágenes que tengan un borde negro tendrán proyecciones vacías en 'Y' y 'Z', porque el borde tiene el mínimo valor posible (0), es por este motivo que el valor 0 ya no es tomado en cuenta para hallar el valor mínimo y al final del cálculo, si ningún valor era diferente de 0, el píxel a colocar en esa posición es 0 para así evitar una imagen negativa (fondo blanco)
El TAD Proyección es innecesario	El TAD fue removido y el procedimiento encargado de realizar la proyección ahora es una función miembro del TAD Volumen, porque es el volumen quien se tiene que proyectar y tiene más sentido que esté ahí
La ayuda general sólo requiere el nombre (y si acaso los parámetros) de cada comando.	Se modificó el comando para que muestre el nombre y una breve descripción porque lo consideramos necesario, para ver el uso de un comando en específico es necesario usar el comando: ayuda <comando>

Descripción de entradas

Dentro de las entradas a nuestro sistema se encuentra por una parte una imagen o una serie de imágenes (un volumen) en formato PGM, un archivo de texto que representa el valor en escala de grises de cada uno de los píxeles que conforman la imagen, donde los valores más cercanos al 0 representan colores más cercanos al negro y los valores más cercanos al máximo representan colores más cercanos al blanco.

Descripción de Salidas

Las salidas del sistema son igualmente imágenes en formato PGM producto de la proyección 2D sobre un volumen de imágenes en la dirección indicada ya sea X,Y o Z y que contienen una serie de radiografías de un cerebro humano.

Diseño de los TADs

TAD Imagen
Conjunto Mínimo de Datos
formato_imagen, cadena de caracteres, representa el formato de la imagen
ancho, entero, representa el ancho de la imagen en píxeles
alto, entero, representa el alto de la imagen en píxeles
max_tam, entero, representa el valor del píxel más grande de la imagen
matriz_pixeles, matriz (vector de vectores) de bytes, representa los valores de gris de los pixeles en las coordenadas bidimensionales
nombre_archivo, cadena de caracteres, nombre del archivo

Comportamiento (Operaciones)
Imagen(), crea una nueva imágen vacía
Imagen(matriz_pixeles), crea una nueva imágen a partir de una matriz de escala de grises
Imagen(nombre_archivo), crea una nueva imágen a partir de un archivo
guardar_archivo(nombre_archivo), guarda la imagen actual en formato PGM
get_formato(), retorna el formato de la imagen cargada
get_ancho(), retorna el ancho de la imagen cargada
get_alto(), retorna el alto de la imagen cargada
get_max_tam(), retorna el valor del pixel mas grande de la imagen cargada
get_pixeles(), retorna la matriz de pixeles de la imagen cargada
get_nombre_archivo(), retorna el nombre del archivo de la imagen
set_formato(formato), establece un nuevo formato para la imagen
set_ancho(ancho), establece un nuevo ancho para la imagen
set_alto(alto), establece un nuevo alto para la imagen
set_max_tam(maxtam), establece un nuevo tamaño máximo de pixel para la imagen
set_pixeles(matriz_pixeles), establece un nueva matriz de pixeles que forma la imagen
set_nombre_archivo(nombre), establece un nuevo nombre para el archivo de la imagen

to_string(), mostrar informacion basica de la imagen en texto
matriz_vacia(ancho, alto), crea una matriz inicializada en 0
llenar_matrix(mtx, value), llena los contenidos de una matriz con el valor
reflejo_vertical(img), devuelve una copia de la imagen pero reflejada verticalmente

TAD Volumen
Conjunto Mínimo de Datos
volumen, cola de Imagen, representa las capas (Imágenes) que conforman el volumen
nombre_base, cadena de caracteres, representa el nombre base de las imágenes en la serie
tam_volumen, entero, representa a el número de imágenes en la serie de imágenes (capas)
ancho, entero, representa el ancho en pixeles de una capa
alto, entero, representa el alto en pixeles de una capa

Comportamiento (Operaciones)
Volumen(), constructor de un volumen vacío
Volumen(nombre_base, tam), carga todas las imágenes con el nombre base en un volumen
get_nombre_base(), retorna el nombre base de la imagen presente en el volumen
get_tam_volumen(), retorna el tamaño del volumen cargado
get_volumen(), retorna el volumen de imágenes cargado
get_ancho(), retorna el ancho de una imagen dentro del volumen
get_alto(), retorna el alto de una imagen dentro del volumen
set_nombre_base(nombre_base), establece un nuevo nombre base para las imágenes dentro del volumen
set_tam_volumen(tamVolumen), establece un nuevo tamaño para el volumen
set_volumen(volumen), establece un nuevo volumen
to_string(), imprime la información básica del volumen
crear_proyeccion(criterio, direccion, nombre_archivo), crea la proyeccion 2D del volumen con el criterio y direccion especificada y lo guarda en el archivo especificado

TAD Controlador
Conjunto Mínimo de Datos
imagen_cargada, apuntador a Imagen, guarda en memoria la Imagen cargada por el usuario
volumen_cargado, apuntador a Volumen, guarda en memoria el Volumen cargado por el usuario
Comportamiento (Operaciones)
cargar_imagen(argumentos), Carga en memoria la imagen especificada
cargar_volumen(argumentos), Cargar en memoria una serie ordenada de imágenes con un nombre base y un tamaño 'n_im' (Volumen)
info_imagen(argumentos), Muestra información básica de la imagen actualmente carga en memoria.
info_volumen(argumentos), Muestra información básica del volumen actualmente cargado en memoria.
proyeccion2D(argumentos), Generar una proyección 2D a partir de un volumen cargado en memoria y guardarlo en un archivo.

TAD Huffman
Conjunto Mínimo de Datos
ancho, entero sin signo de 2 byte, guarda el ancho de la imagen
alto, entero sin signo de 2 byte, guarda el ato de la imagen
maximo, entero, guarda la intensidad máxima de la imagen
arbol, ArbolCodificacion, apunta al nodo raíz del árbol de codificación
imagen, Imagen, guarda la imagen desde la que se crea la codificación de huffman
codigos,mapa, mapa de CodigoElemento, guarda los códigos de cada uno de los nodos del árbol
Comportamiento (Operaciones)
Huffman(imagen), construye la codificación de huffman de una imagen
Huffman(nombre_archivo, salida), decodifica un archivo huffman
guardar_archivo, guarda la imagen formada con la codificación de huffman en un archivo binario

TAD ArbolCodificacion
Conjunto Minimo de Datos
Raiz, apuntador a NodoCodificación, referencia a la raiz del arbol de codificación

Comportamiento
ArbolCodificacion(),constructor de un nodo del arbol
ArbolCodificacion(frecuencias),constructor con parámetros que recibe las frecuencias de los pixeles
~ ArbolCodificacion(),destructor un nodo del arbol

TAD CodigoElemento
Conjunto Minimo de Datos
Elemento, dato tipo plantilla, almacena el pixel de la imagen a codificar
Frecuencia, entero, almacena la frecuencia de repetición de un pixel de la imagen
Codigo,cadena de caracteres, representa el codigo que se le asigna al pixel para codificarlo
Comportamineto
CodigoElemento(), constructor por defecto
CodigoElemento(argumentos), constructor por defecto que recibe
CodigoElemento(argumentos), constructor por defecto
CodigoElemento(elemento,frecuencia,codigo), constructor que recibe parametros
To_string(),metodo to string

TAD NodoCodificación
Conjunto Minimo de Datos
Frecuencia,entero, almacena la frecuencia con la que se repite el pixel en la imagen
Comportamiento
NodoCodificacion(),constructor por defecto
NodoCodificacion(frecuencia), contructor

TAD NodoElemento
Conjunto Minimo de Datos
Dato,dato tipo plantilla, almacena el dato del codigo que contiene cada nodo
Comportamiento
NodoElemento(),constructor por defecto
NodoElemento,elemento, frecuencia)

TAD NodoFrecuencia
Conjunto Minimo de Datos
Hijoder, apuntador a NodoCodificacion, almacena la referencia del nodo como hijo derecho
HijoIzq, apuntador a NodoCodificacion, almacena la referencia del nodo como hijo izquierdo
Comportamiento
NodoFrecuencia(), constructor por defecto
NodoFrecuencia(frecuencia), constructor con parámetros que recibe la frecuencia de un pixel
Codigos_elementos(codigo), retornar un vector con el resultado del codigo teniendo en cuenta si es hijo derecho o izquierdo.

Diagrama TADs

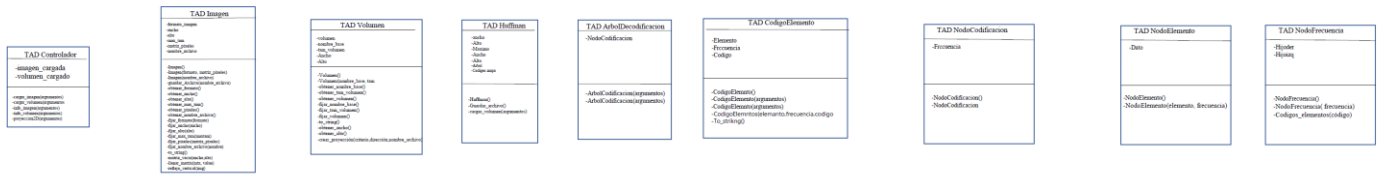
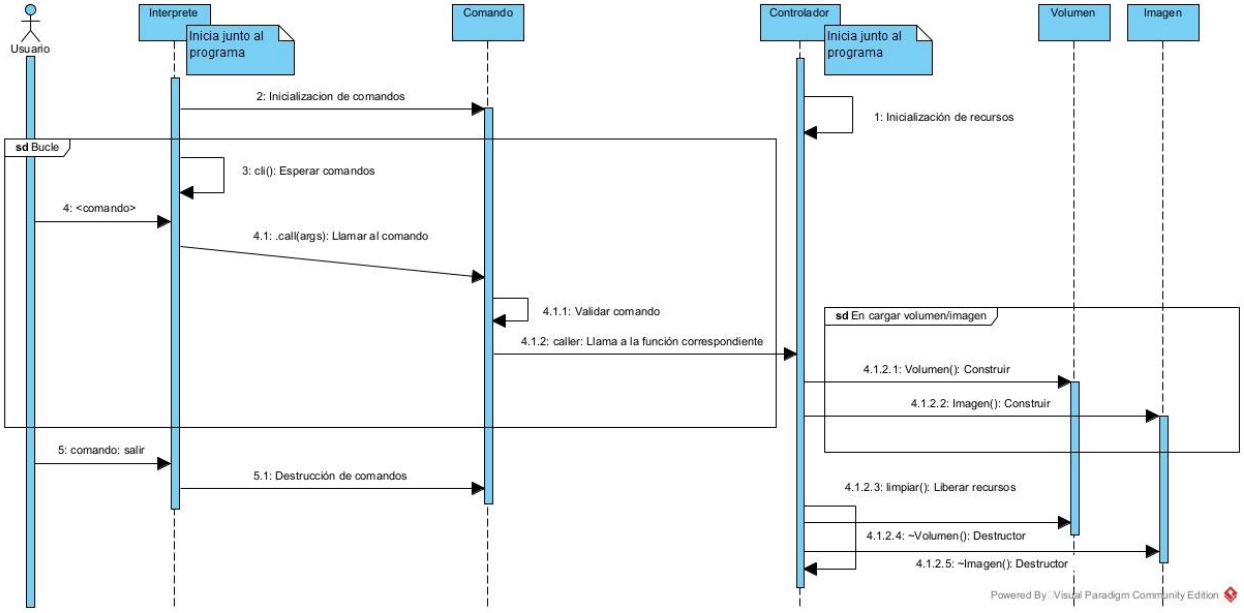
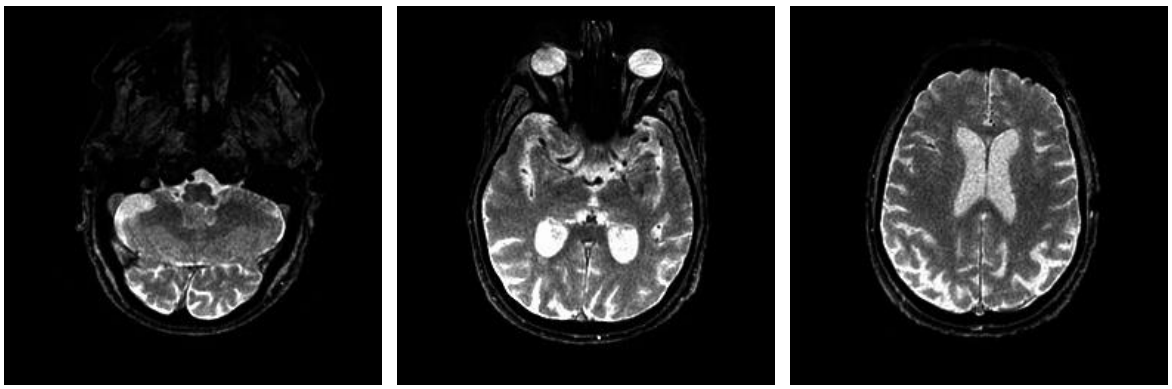


Diagrama de secuencias (Línea de comandos)

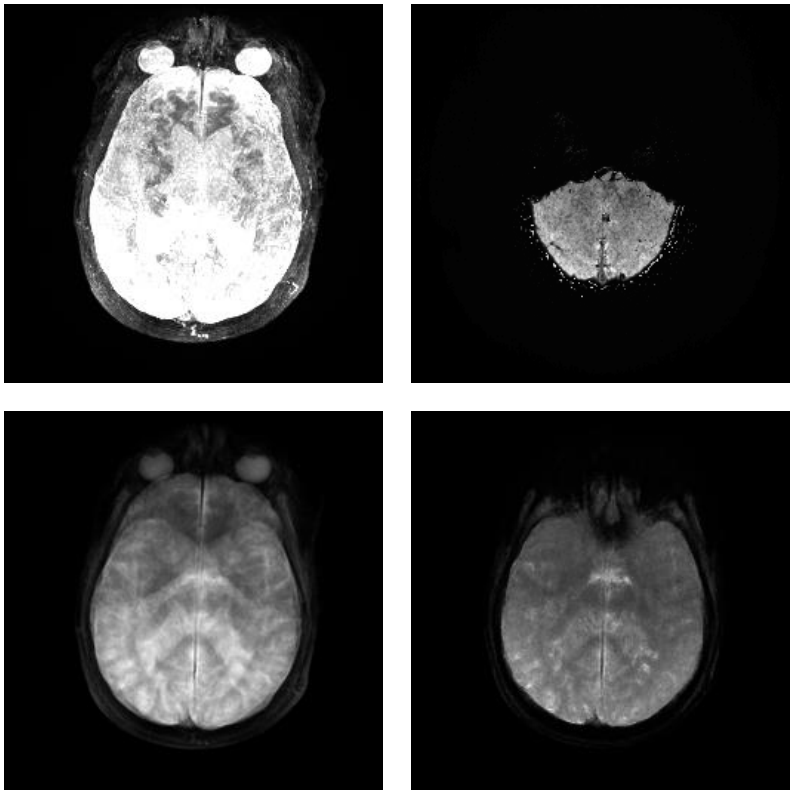

Usuario
Interprete
Inicia junto al programa
Comando
Controlador
Inicia junto al programa
Volumen
Imagen
2: Inicializacion de comandos
1: Inicialización de recursos
sd Bucle
3: cli(): Esperar comandos
4: <comando>
4.1: .call(args): Llamar al comando
4.1.1: Validar comando
4.1.2: caller: Llama a la función correspondiente
sd En cargar volumen/imagen
4.1.2.1: Volumen(): Construir
4.1.2.2: Imagen(): Construir
5: comando: salir
5.1: Destrucción de comandos
4.1.2.3: limpiar(): Liberar recursos
4.1.2.4: ~Volumen(): Destructor
4.1.2.5: ~Imagen(): Destructor
Powered By: Visual Paradigm Community Edition

Plan de pruebas Proyecciones 2D:
Para corroborar el funcionamiento de las proyecciones 2D se realizaron proyecciones con el volumen *IM-126-0002-epiT2*

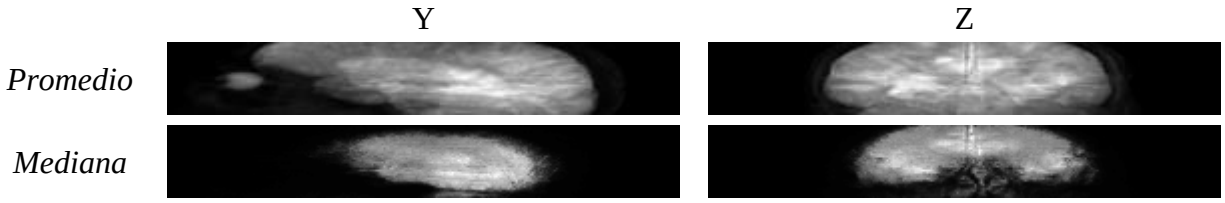
Algunas de las imágenes de *IM-126-0002-epiT2:*



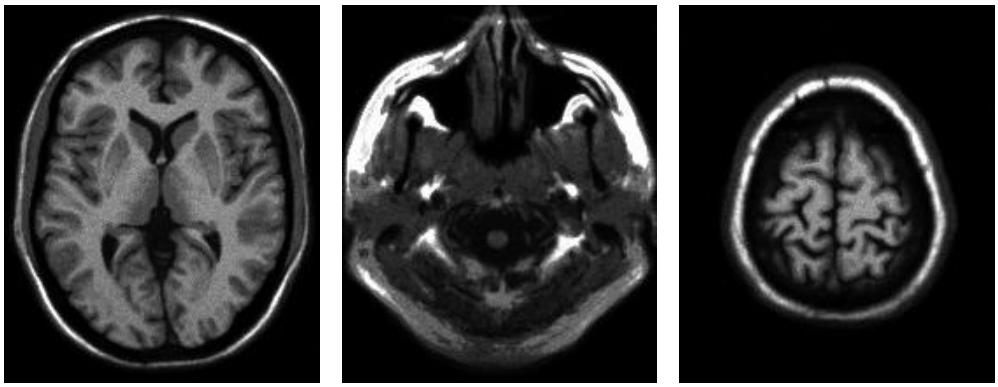
Proyecciones en 'x' (Maximo, Minimo, Promedio, Mediana)



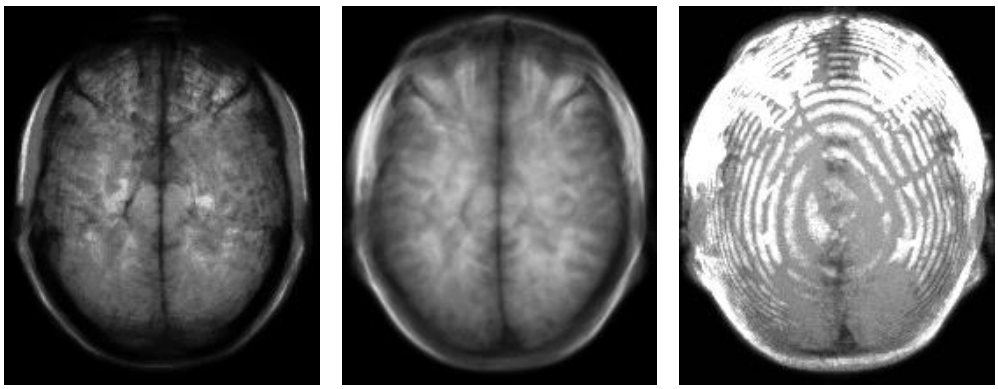
Algunas proyecciones 'Y' y 'Z':



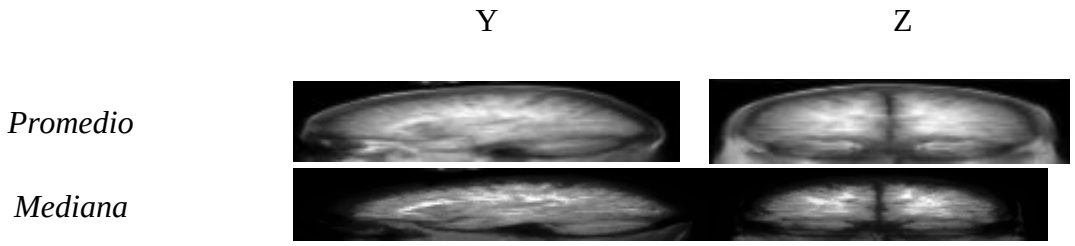
Algunas de las imágenes de *t1_icbm_5mm_:*



Algunas proyecciones en X (mediana, promedio, maximo):

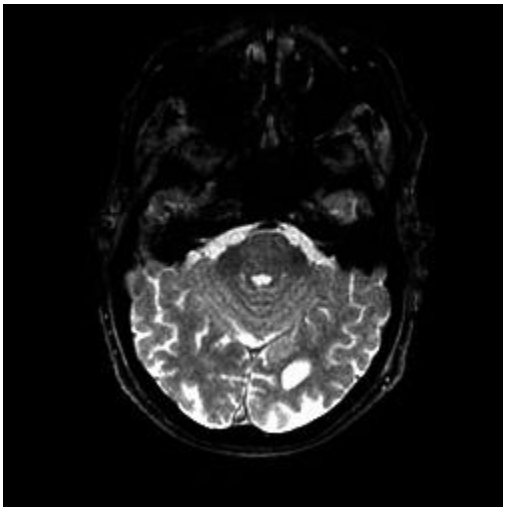


Algunas proyecciones 'Y' y 'Z':



Plan de pruebas Codificación y Decodificación:

Tenemos la siguiente imágen:



Esta imágen tiene una altura y anchura de 256px y un valor de píxel máximo de 255.

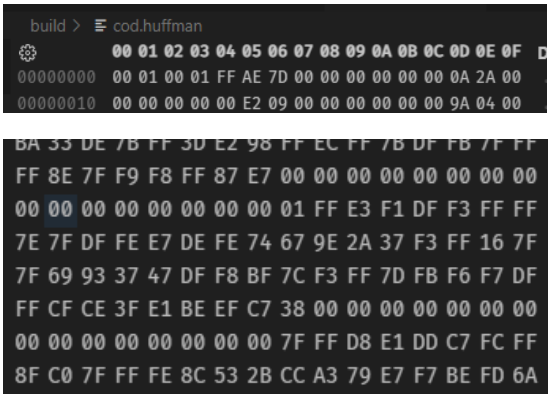
1. Se carga la imágen con el comando *cargar_imagen*

```
$ cargar_imagen imagen.pgm
(proceso satisfactorio) La imagen 'imagen.pgm' ha sido cargada
$
```

2. Se utiliza el comando *codificar_imagen* para generar la codificación de huffman.

```
$ codificar_imagen cod.huffman
(proceso satisfactorio) La imagen en memoria ha sido codificada exitosamente
$
```

Ahora se tiene el siguiente archivo binario:

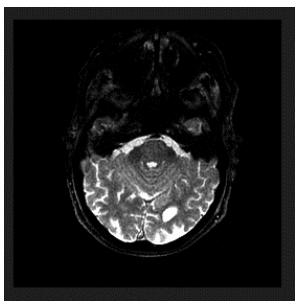


Como se puede ver en el editor hexadecimal, los primeros dos bytes corresponden a la anchura (0x0100 hex ó 256 en decimal, los bytes están invertidos por la arquitectura BigEndian del computador) y los siguientes 2 bytes a la altura, el siguiente byte al valor máximo (0xFF hexadecimal, 255 decimal), luego están las 255 frecuencias correspondientes a cada intensidad posible codificadas en valores de 8 bytes, y al final del archivo se puede ver la codificacion binaria.

Para decodificar el archivo utilizamos el comando *decodificar_archivo*

```
$ decodificar_archivo cod.huffman salida.pgm
(proceso satisfactorio) El archivo cod.huffman ha sido decodificado exitosamente
$
```

Que nos devuelve como resultado la imagen original:



El resultado de la codificación y decodificación reduce significativamente el peso del archivo.

```
$ du -sh imagen.pgm
152K    imagen.pgm
(0) angel at AngelgoLapt
$ du -sh cod.huffman
36K     cod.huffman
(0) angel at AngelgoLapt
$ du -sh salida.pgm
152K    salida.pgm
```