PONTIFICIA UNIVERSIDAD JAVERIANA

ESTRUCTURA DE DATOS

ENTREGA#0 PROYECTO

PRESENTA:

Diego Fernando Zabala,

Brandon Garcia Rodriguez

Santiago Camargo Trujillo

David Villareal

PROFESOR

John Jairo Corredor Franco

BOGOTA D.C

2025

- Resumen

El problema que aborda este proyecto es la falta de una interfaz interactiva que permita la manipulación sencilla de archivos de imagen en formato PGM. Se identifica la necesidad de contar con una herramienta que facilite la carga, inspección, procesamiento y transformación de imágenes y volúmenes en entornos de procesamiento de datos. La solución propuesta es una consola interactiva que implementa comandos específicos para la manipulación de imágenes. Se utilizan estructuras de datos eficientes como unordered_map para la gestión de comandos. Los resultados obtenidos incluyen una interfaz flexible que permite la interacción intuitiva con imágenes y archivos codificados.

- Introducción

El procesamiento de imágenes es una tarea fundamental en diversas disciplinas como la visión por computadora, la medicina y la ingeniería. Sin embargo, herramientas especializadas pueden ser complejas de usar, especialmente para usuarios sin experiencia en software de procesamiento. Por ello, se desarrolla una consola interactiva que permite cargar, manipular y transformar imágenes de manera sencilla. Esta herramienta permite realizar operaciones como la carga de imágenes, consulta de información, segmentación, proyecciones 2D y codificación/decodificación de archivos.

- **Diseño**

El diseño del sistema se basa en la implementación de un **Tipo Abstracto de Datos (TAD)** denominado ConsolaInteractiva, el cual encapsula la lógica de interacción con el usuario y la gestión de comandos.

ConsolaInteractiva

Atributos:

- comandos: Mapa unordered_map<string, function<void(const vector<string>&)>> que asocia nombres de comandos con sus respectivas funciones.
- imagen_cargada: string que almacena el nombre de la imagen cargada en memoria.
- ancho: int que representa el ancho de la imagen cargada.
- alto: int que representa el alto de la imagen cargada.

Operaciones:

1. **ejecutar()**: Ejecuta la consola y procesa los comandos ingresados por el usuario.
2. **procesar_comando(const string& entrada)**: Separa el comando y sus argumentos y lo ejecuta si es válido.

3. **ayuda(const vector<string>& args)**: Muestra la lista de comandos disponibles o información detallada de un comando.
4. **salir(const vector<string>& args)**: Termina la ejecución de la consola.
5. **cargar_imagen(const vector<string>& args)**: Carga una imagen PGM en memoria.
6. **info_imagen(const vector<string>& args)**: Muestra información de la imagen cargada.
7. **leer_dimensiones_pgm(const string& nombre_archivo, int &ancho, int &alto)**: Lee las dimensiones de una imagen en formato PGM.
8. **proyeccion2D(const vector<string>& args)**: Genera una proyección 2D de la imagen cargada.
9. **decodificar_archivo(const vector<string>& args)**: Decodifica un archivo en formato .huf y lo guarda como .pgm.
10. **cargar_volumen(const vector<string>& args)**: Carga un volumen de datos en memoria.
11. **info_volumen(const vector<string>& args)**: Muestra información del volumen cargado.
12. **codificar_imagen(const vector<string>& args)**: Codifica una imagen en formato .huf.
13. **segmentar(const vector<string>& args)**: Segmenta una imagen en base a semillas y grafos.
14. **clr(const vector<string>& args)**: Limpia la pantalla de la consola.
15. **descripcion_comando(const string& comando, bool detallado)**: Retorna la descripción de un comando en particular.

Condiciones:

- Si no hay una imagen cargada, algunos comandos deben impedir su ejecución y mostrar un mensaje de error.
- El formato de los archivos debe ser validado antes de procesarlos.
- La consola debe ejecutarse en un bucle continuo hasta que el usuario ingrese el comando salir.

- **Análisis**

Este diseño sigue un enfoque modular que facilita la extensibilidad, ya que los comandos están almacenados en una estructura unordered_map, lo que permite agregar nuevas funcionalidades sin modificar demasiado el código base.

Estructura principal

- ConsolaInteractiva contiene:

- Un **mapeo de comandos** (unordered_map<string, function<void(const vector<string>&)>>), donde las claves son los nombres de los comandos y los valores son punteros a funciones que ejecutan las operaciones.
- Atributos para manejar el **estado de la imagen cargada**, como imagen_cargada, ancho y alto.
- Funciones específicas para manipular imágenes en formato PGM y archivos .huf.

2. Flujo de Ejecución

El programa sigue este flujo principal:

1. **Inicio**: Se ejecuta ejecutar(), que abre un bucle donde se reciben comandos del usuario.
2. **Procesamiento de comandos**:
    a. procesar_comando(entrada): Separa el comando y sus argumentos.
    b. Busca el comando en el unordered_map.
    c. Si el comando existe, ejecuta la función correspondiente; si no, muestra un error.
3. **Ejecución de funciones según el comando**:
    a. Ejemplo: Si el usuario escribe cargar imagen.pgm, se ejecuta cargar_imagen({"imagen.pgm"}).
4. **Finalización**: Cuando el usuario escribe salir, el bucle termina y la consola se cierra.

unordered_map para los comandos

El uso de unordered_map es clave para la eficiencia de la consola, ya que permite una búsqueda en **O(1)** en promedio para acceder a las funciones.

Manejo de archivos con ifstream

Las funciones que leen archivos (leer_dimensiones_pgm) usan ifstream, lo cual es eficiente para leer archivos pequeños.
Para archivos grandes, se podría mejorar usando **lectura en bloques**.

Manejo de imágenes en memoria

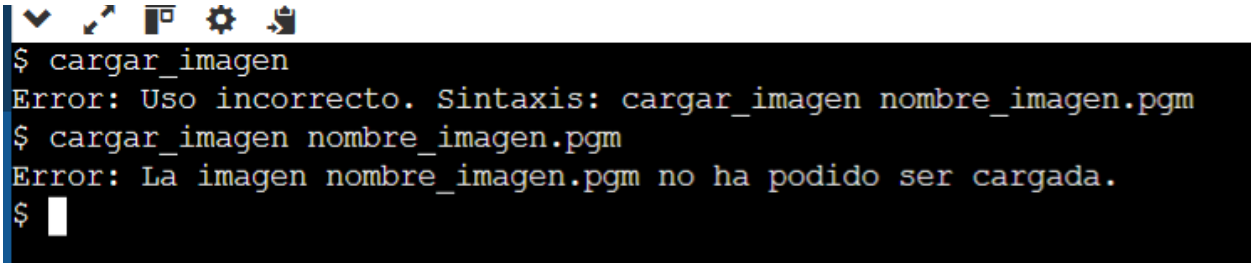
El código actual solo almacena el nombre y dimensiones de la imagen, pero no su contenido en memoria.
Si se requiere procesamiento más avanzado, se podría usar una estructura como:

vector<vector<int>> datos_imagen;

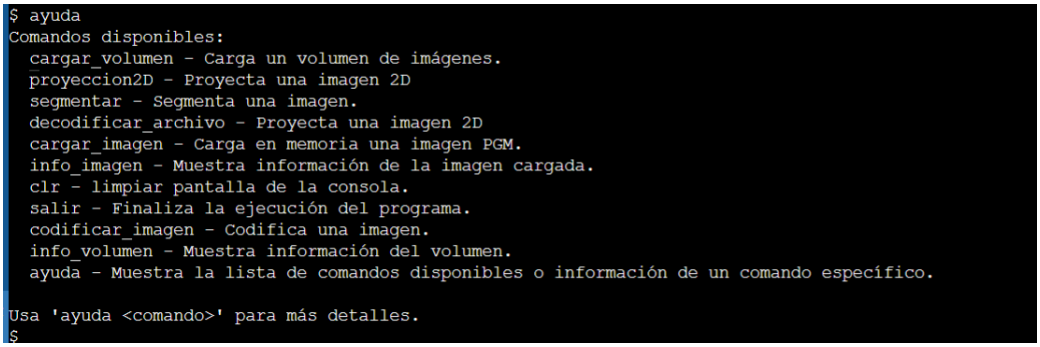
Donde cada píxel de la imagen se almacena en una matriz.

- **Plan de pruebas**

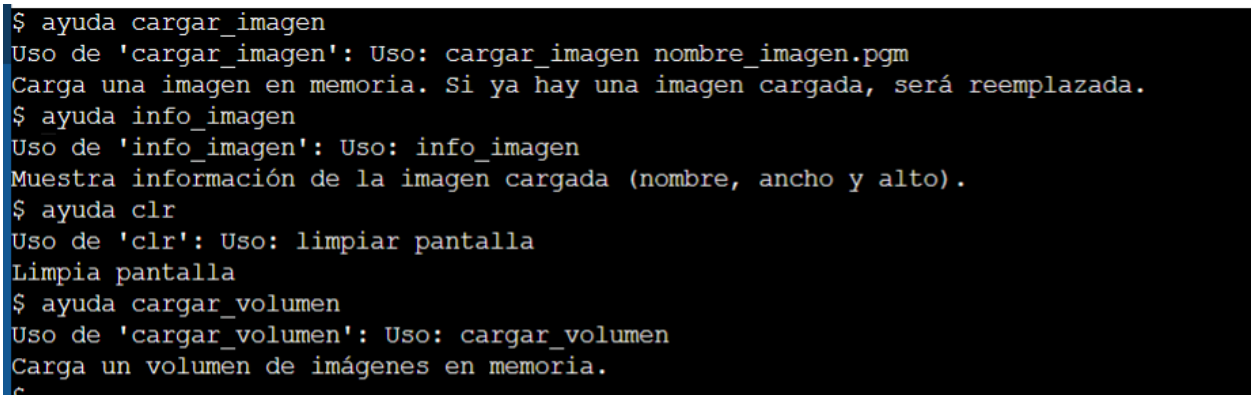
Para las pruebas de este código usaremos el compilador online GBD en el cual probaremos que los comandos están funcionando a nivel de ejecución.



El comando de cargar_imagen funciona perfectamente en este caso aun no funciona dado que no existe una imagen.pgm para que cargar imagenes esté funcionando tal como funciona la lógica para leer archivo pgm.



Este comando de ayuda es el cual nos muestra cuales son los comandos disponibles para la ejecución por medio de terminal, además este mismo comando puede mostrar ayuda por cada comando



Y en este momento para cada uno de los comandos solo ejecutara como ”funcionando” o mostrara error puesto que aún no se han implementado los métodos necesarios para que la función cumpla con lo que se requiere.

- **Conclusiones**

La consola interactiva propuesta ofrece una solución sencilla y eficiente para la manipulación de imágenes en formato PGM. Se ha diseñado de manera modular, permitiendo la adición de nuevos comandos de forma sencilla. La utilización de estructuras como unordered_map para la gestión de comandos garantiza una ejecución rápida. Además, la implementación de funciones como leer_dimensiones_pgm optimiza el manejo de archivos. En futuras mejoras, se podrían integrar métodos para soportar otros formatos de imagen y ampliar las capacidades de segmentación y procesamiento de datos.