Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Estructura de datos P
Vacaciones de junio 2019
**Catedrático:** Ing. Luis Espino
**Auxiliar:** Javier Navarro

# Drawing canvas

*Practica de laboratorio*

## Contenido

# 1 Objetivos

## 1.1 Objetivo general

Que el estudiante aplique los conocimientos adquiridos sobre estructuras de datos en aplicaciones que permitan manipular la información de forma óptima para el ordenador y permita mostrarla de forma sencilla.

## 1.2 Objetivos específicos

- Utilizar estructuras de datos como matrices y árboles para almacenar la información en tiempo de ejecución.
- Generar reportes que representen el estado de la memoria de forma gráfica.
- Representar una estructura lógica (matriz) en una representación en la capa de aplicación.

# 2 Descripción

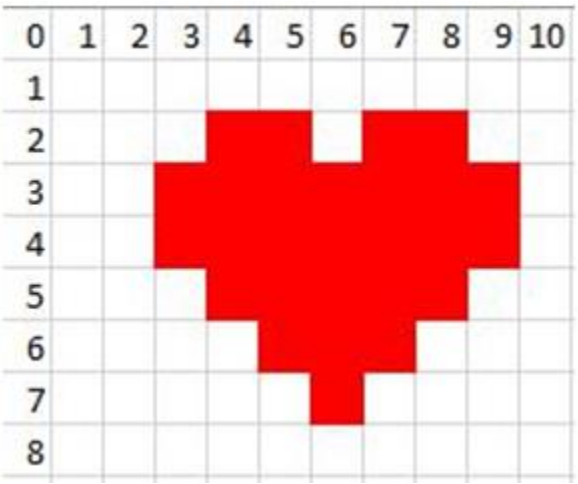
El sistema consiste en un generador de imágenes por capas, la aplicación contara con un conjunto de capas cargadas previamente y almacenadas en memoria para ser utilizadas, estas capas contendrán imágenes hechas con pixeles, cada capa contendrá elementos simples o compuestos, al colocar una capa sobre otra estas irán formando una imagen más completa.

El sistema es capaz de generar imagen seleccionando las capas deseadas

## 2.1 Capa (a nivel de aplicación)

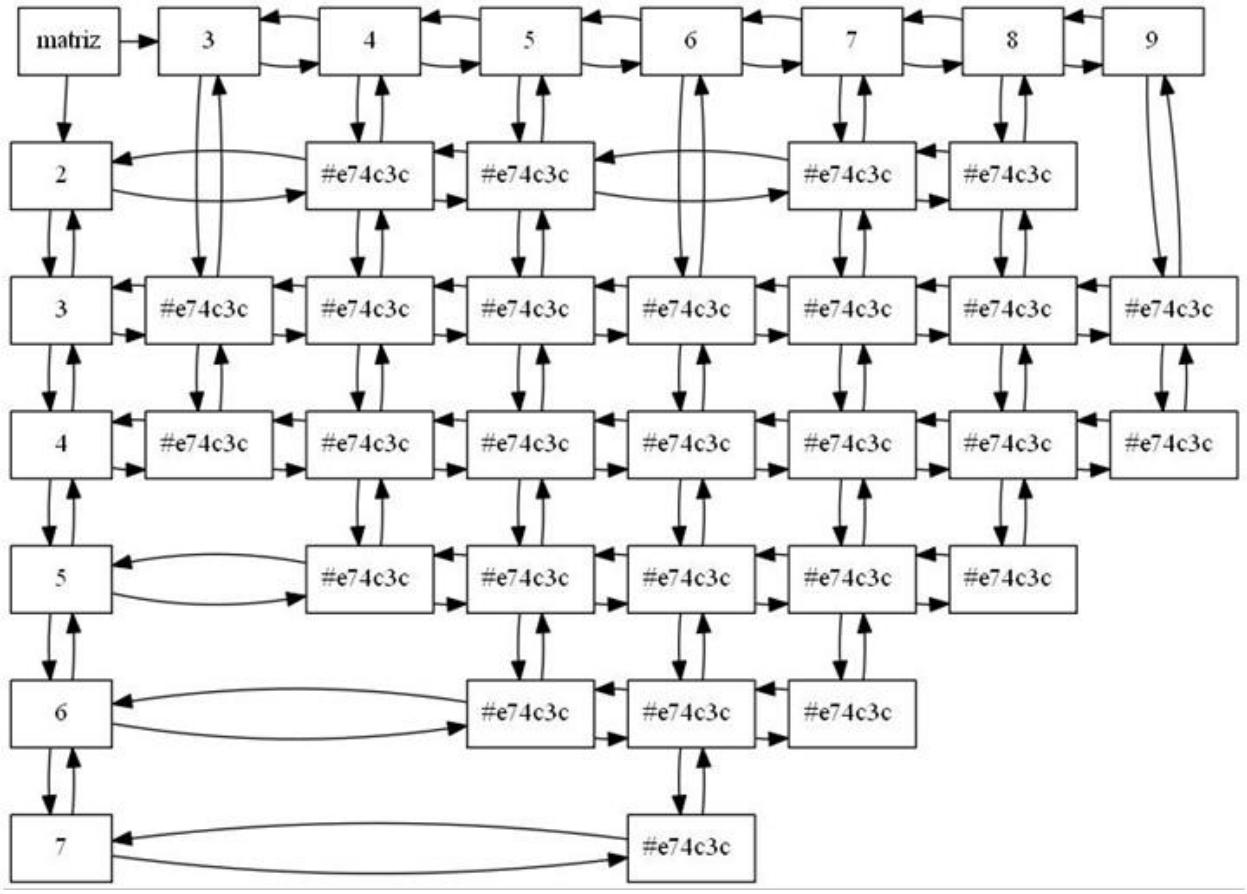
Una capa contiene la definición de elementos simples por ejemplo una piedra, un muro o elementos compuestos como una casa, un árbol. Las capas se almacenarán utilizando matrices dispersas dado que no existe una dimensión fija para las capas, en cada nodo se almacenará el color del píxel en Hexadecimal.

Por ejemplo:

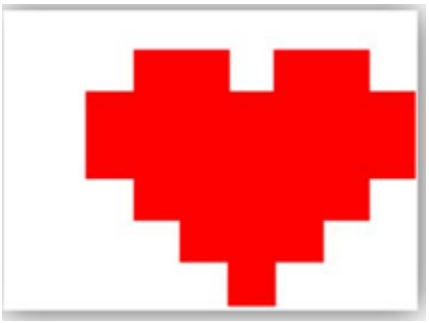


## 2.2 Capa (a nivel lógico)

La capa a nivel lógico estará representada por una estructura matricial, se propone una matriz dispersa o poco densa para dar solución a tal necesidad. Por lo tanto, una imagen como la anterior dentro de la matriz quedaría así:

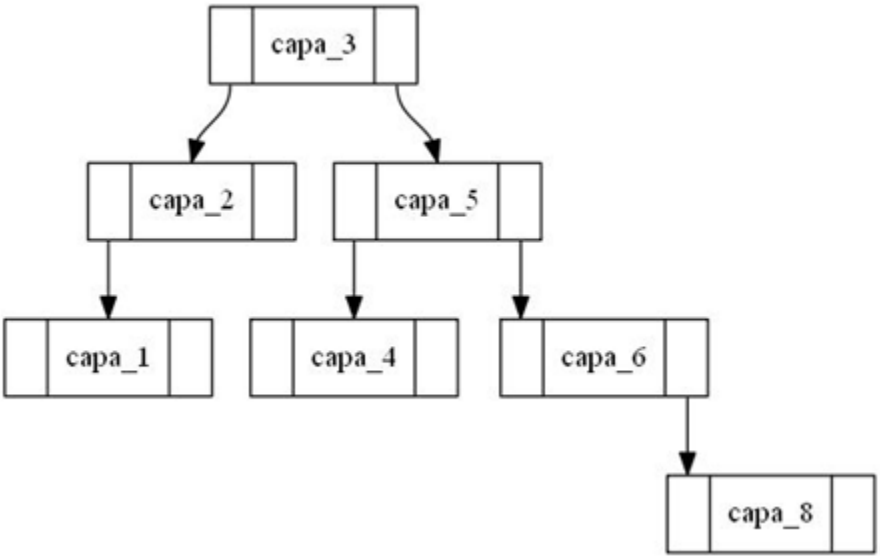



Si se generara una imagen únicamente de esta capa el resultado debería ser:



Consideraciones:

- En las áreas donde no existe color se mostrará blanco al generar la imagen.
- Las capas son únicas por lo cual una capa puede aparecer en cualquier imagen solo una vez estas tendrán un id que las identificará.
- Las capas deben almacenarse en un árbol binario de búsqueda (ABB).




Existirá un archivo de entrada que permitirá la carga masiva de capas, siguiendo la sintaxis:

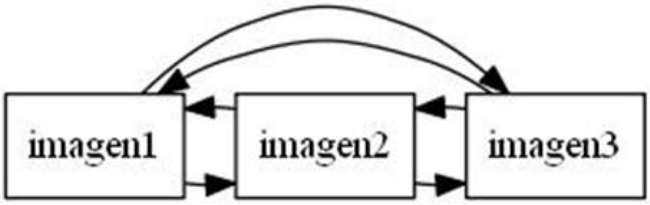
```
Id {
fila,columna,color;
...
fila columna, color;
}

Id {
fila,columna,color;
...
fila columna, color;
}
```

## 2.3 Imagen

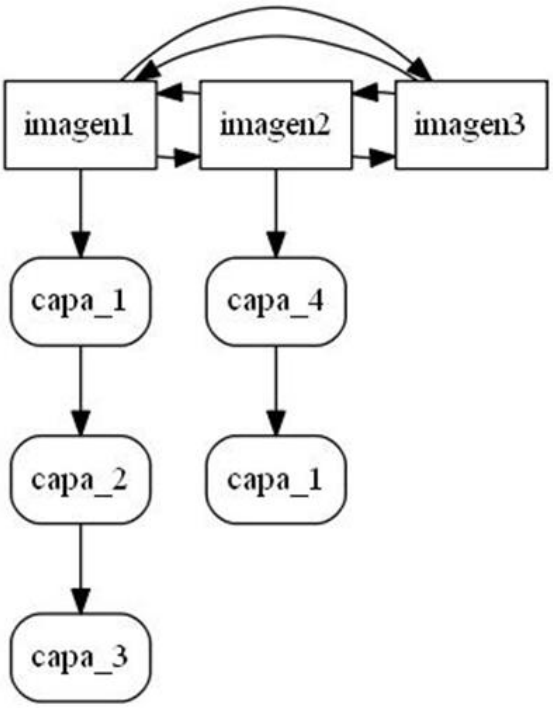
Una imagen es el resultado de superponer una o más capas, las imágenes se pueden generar de diferentes maneras que serán definidas más adelante.

La aplicación contara con un conjunto de imágenes predefinidas, cada imagen a la vez tiene una lista de capas que conforman la imagen, se almacenará en una lista circular doblemente enlazada ordenada según el id de la imagen.

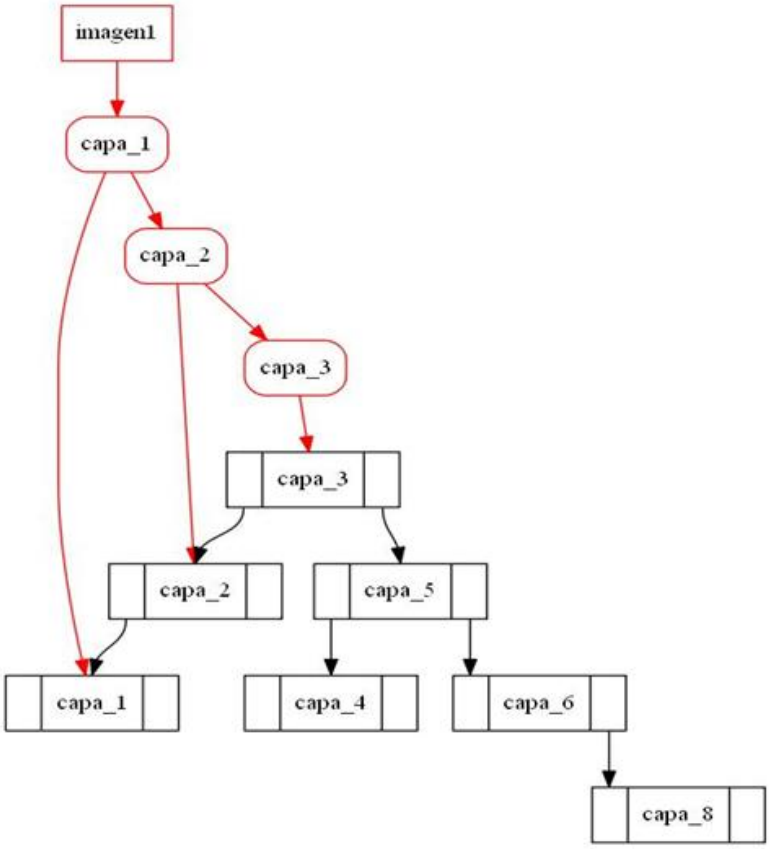



Cada imagen a su vez tendrá una lista de capas, el orden en el que se insertan las capas indica la forma en que se superpondrán las capas.

Por ejemplo, en la imagen2 la capa4 se colocará primero y sobre esta se debe colocar la capa1.



La lista de capas de cada imagen debe apuntar a la capa respectiva en el árbol:




La sintaxis del archivo de carga para imágenes será:

```
id{ idcapa, ... ,idcapa }
```

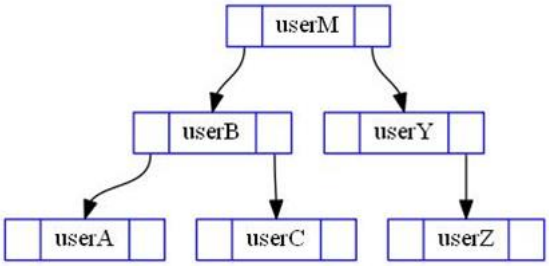
```
1{2,3,4}
2{4,1}
3{}
100{4,2,3,1}
50{1}
```

El archivo que contiene la definición de las imágenes tendrá la extensión *.im

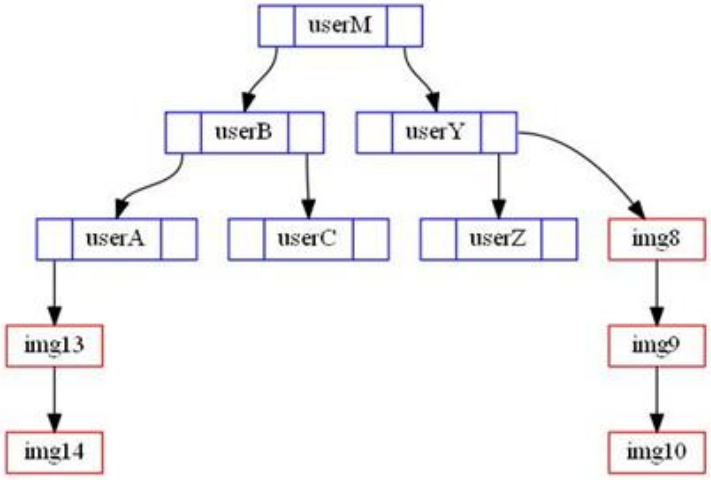
En el caso de las imágenes que no poseen capas, estas solo generaran un pixel negro al momento de generarse la imagen.

## 2.4 Usuarios

Dentro de la aplicación se manejan usuarios, los usuarios son capaces de agregar imágenes, los usuarios se almacenarán en un árbol AVL según su nombre.




Los usuarios pueden tener una lista de imágenes que han registrado. La lista de imágenes de los usuarios puede ser de enlace simple o doble quedando a su elección.




Ejemplo de archivo de carga para usuarios:

```
userM:;
userB:;
userA:13,14;
userC:;
userY:8,9,10;
userz:;
```

El orden de la carga de archivos será:

- capas
- imágenes
- usuarios

# 3 Funcionalidades

## 3.1 Generación de imágenes

### 3.1.1 Por recorrido limitado

En esta opción se indicará el número de capas a utilizar y el tipo de recorrido que se desea para generar la imagen: Preorden, inorden, postorden.

Por ejemplo, en el caso de que el número de capas a utilizar sea 3, entonces, las tres posibles opciones serian:

- Si el tipo fuera inorden se deberán superponer las capas 1,2,3;
- Si el tipo fuera preorden se deberán superponer las capas 3,2,1
- Si el tipo fuera postorden se deberán superponer las capas 1,2,4

### 3.1.2 Por lista de imágenes

Se ingresará el id de la imagen el cual deberá buscarse en la lista circular de imágenes y según la lista de capas que esta tenga se deberá generar la imagen.

### 3.1.3 Por capa

Se ingresará el id de la capa a graficar y se buscará en el árbol ABB.

### 3.1.4 Por usuario

Debe permitir buscar un usuario y seleccionar la imagen a graficar de su lista de imágenes.

## 3.2 ABC

### 3.2.1 Usuarios

La aplicación debe ser capaz de agregar nuevos usuarios, eliminar usuarios y modificar usuarios.

### 3.2.2 Imágenes

Agrega una nueva imagen seleccionando el usuario a quien se agregara tomando en cuenta que los id de imagen es único y de existir el id no permitirá agregarla, elimina imágenes primero seleccionando el usuario y luego el id de la imagen, actualizando la lista circular y la lista de imágenes del usuario a quien pertenecía la imagen.

## 3.3 Estado de la memoria

### 3.3.1 Ver lista de imágenes

Deberá mostrar la lista circular doble y la lista de capas de cada imagen

### 3.3.2 Ver árbol de capas

Deberá mostrar el árbol de capas

### 3.3.3 Ver árbol de capas espejo

Deberá mostrar el árbol de capas en espejo

### 3.3.4 Ver capa

Se indicará el número de capa a mostrar

### 3.3.5 Ver imagen y árbol de capas

Se debe seleccionar una imagen, se mostrará la lista de capas y el árbol de capas, mostrando el enlace de la lista de capas hacia el árbol ABB de capas.

### 3.3.6 Ver árbol de usuarios

Mostrará el árbol AVL de usuarios

## 3.4 Otros Reportes

- Top 5 de imágenes con más número de capas
- Todas las capas que son hojas
- Profundidad de árbol de capas
- Mostar capas en postorden
- Listar las capas en: preorden, inorden, postorden
- Top 5 de usuarios con más imágenes
- Árbol espejo de usuarios
- Listar usuarios en recorrido: preorden, inorden, postorden, por niveles.

# 4 Entregables

- Código Fuente
- Ejecutable
- Manual Técnico
- Manual Usuario

# 5 Observaciones

- Lenguaje de programación a utilizar: C++
- Sistema Operativo: Libre
- IDE: Libre.
- Practicas con segmentation default o stack overflow tendrán nota de cero puntos.
- La gráfica debe mostrarse dentro de la aplicación, no buscarse en carpetas ajenas. La aplicación que no genere gráfica no podrá ser calificada, se debe utilizar Graphviz.
- Todas las estructuras deben de ser realizadas por el estudiante, sin el uso de librerías específicas de ningún IDE o Framework.
- Durante la calificación se harán preguntas para validar que el estudiante realizo el proyecto, de no responder correctamente anulara la nota obtenida en la o las secciones en la que aplique tal concepto.
- La aplicación será compilada y ejecutada al momento de la calificación.

# 6 Fecha y modo de entrega:

- Sábado 22 de junio antes de las 8:00 horas vía Classroom, adjuntar entregables en un comprimido con nombre <carnet>.zip
- Si no se cumple con la hora de entrega, se tendrá una penalización de 20%.
- Si no se entrega en Classroom, no se calificará.
- Copias tendrán nota de 0 puntos y serán reportadas al catedrático y a la Escuela de Sistemas.

REFERENCIA: ROCAEL ISIDRO, ENUNCIADO DE VACACIONES 2016 DEL CUAL SE EXTRAJO LA IDEA DEL PRESENTE.