# Relleno

Funciones del programa:

- **void init(void)**
- **void pixel(int x,int y)**
- **void LineaBres(int xa, int ya, int xb, int yb)**
- **void lineSegment(void)**
- **int main(int argc, char** argv)**
- **void onMouse(int button, int state, int x, int y)**
- **void onMotion(int x, int y)**
- **bool compara(color colorA,color colorF)**
- **void inunda(int x, int y)**

## Main

El programa comienza la ejecución con la función main(), la cual maneja todo lo referente a la correcta ejecución del programa.

## Init (void)

Se encarga de Iniciar la ventana, pero a su vez dando otros atributos a la misma dejando todo listo para poder trabajar con la ventana.

## Pixel (int x, int y)

Función que pinta un pixel en las coordenadas x, y que le llegan por parámetro.

## lineSegment (void)

Función encargada de estar forzando el dibujado de las figuras durante la ejecución del programa.

## Void onMouse (int button, int state, int x, int y)

Función encargada de tomar la posición inicial del mouse y pintar el primer píxel.

En caso de dar click derecho, toma las coordenadas de la semilla que se utilizara para comenzar el relleno de la figura.

## Void onMotion (int x, int y)

En esta función es donde se calculan los píxeles actuales del mouse mientras este se encuentre en movimiento para realizar su trazado del lápiz mediante la función de LineaBres.

**Void LineaBres (int xa, int ya, int xb, int yb)**

Función que dibuja una serie de puntos en formato de línea recta, desde las coordenadas iniciales (xa,ya) hasta las coordenadas finales (xb,yb). Su trazado es determinado mediante el algoritmo de Bresenham.

**bool compara(color colorA, color colorF)**

Toma los dos colores que llegan por parámetro y al ser estructuras de color RGB, compara cada color con el equivalente del otro color. En caso de que todos los componentes sean iguales (mismo color) regresa verdadero, en caso contrario falso.

**Void inunda (int x, int y)**

Función en la cual se toma la semilla que llega por parámetro y a través de una comparación del color del fondo con el que se va rellenar y otra comparación del color de relleno con el color del borde de la figura, nos permite hacer llamadas recursivas a esta misma función tomando de 1 semilla otras 4 que son los puntos que están cercanos a la misma.

Una manera de no hacer la recursión es utilizar algún TDA (a poder ser memoria dinámica) para estar almacenando los valores de los puntos que son válidos a pintar y después ir vaciando el TDA y pasar a pintar los puntos almacenados, con esto resuelve el problema de la sobrecarga de recursión, pero sigue con el problema de evaluar y pintar varias veces un solo punto.