



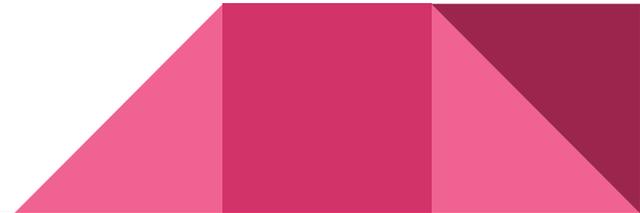
# Programación

Clase 3

División en subtareas

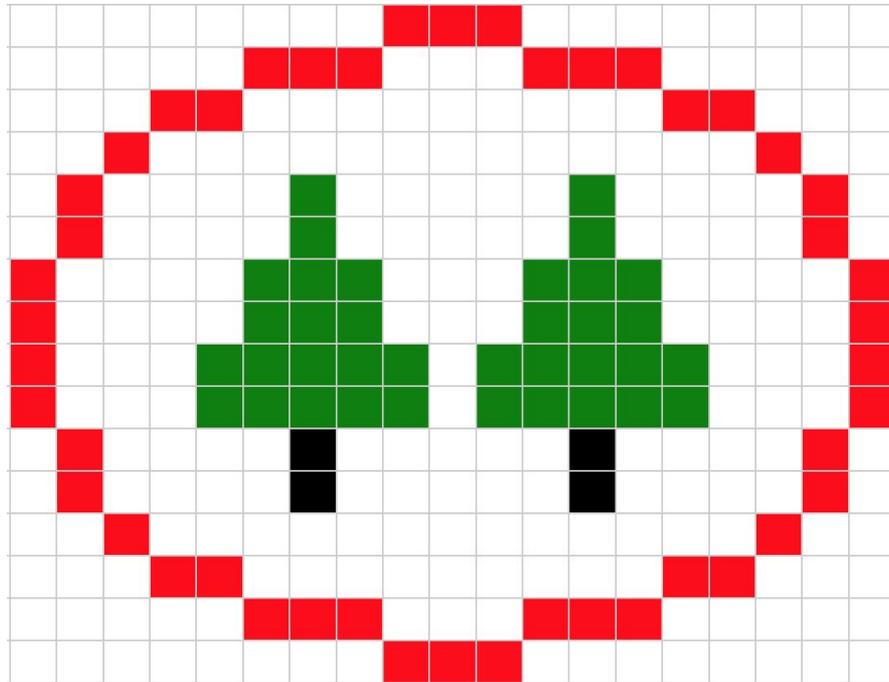
Universidad Nacional de Quilmes

# Precalentando motores

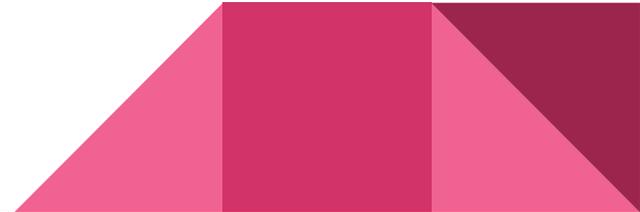
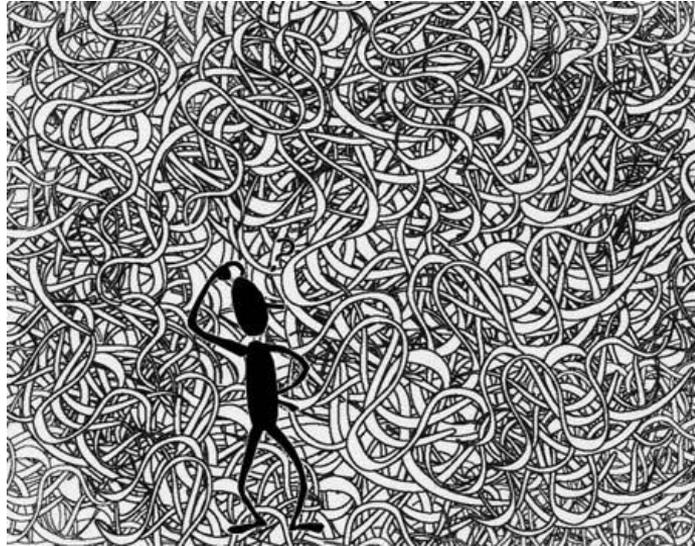


# Ejercicio: Logo con árboles

Realicemos el siguiente dibujo en QDraw.



# ¿Por dónde arrancamos?

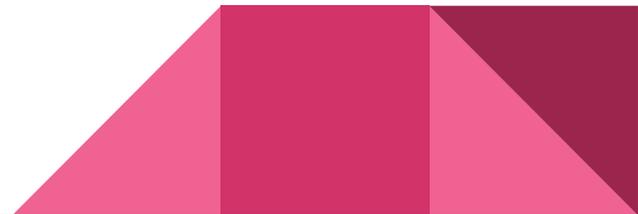




=



Recordemos siempre que **Programar**  
**es comunicar**



# Intentando comunicar la solución

Si le intentamos contar a alguien que es lo que hay que hacer, terminaremos diciendo:

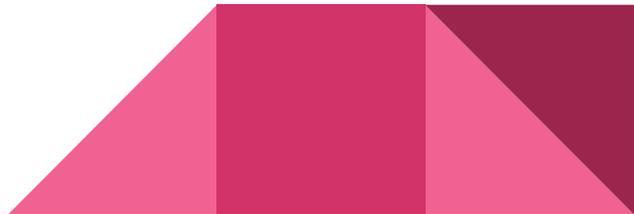
**arriba, arriba, arriba, pinta, arriba, pinta... etc.**

Esta solución presenta varios problemas:

- El código es confuso hasta para nosotrxs mismos.
  - La otra persona no tiene idea de qué le estamos hablando.
  - Los comentarios pueden ayudar a entender el código, pero no solucionan el problema de fondo.
  - Uno quisiera poder transmitir la idea de una forma más sencilla.
- 

# Lo que quisiéramos transmitir es algo del estilo:

```
programa {  
    dibujar óvalo externo en rojo  
    dibujar árbol derecho  
    dibujar árbol izquierdo  
    volver el cabezal a la posición inicial  
}
```



# Divide y vencerás



# Divide y vencerás

Al dividir el problema general en problemas más pequeños, podemos centrarnos en resolver cosas más sencillas, que requieren menos código y son más fáciles de razonar.

De esta forma es más fácil atacar problemas grandes, para arribar a una solución integral.



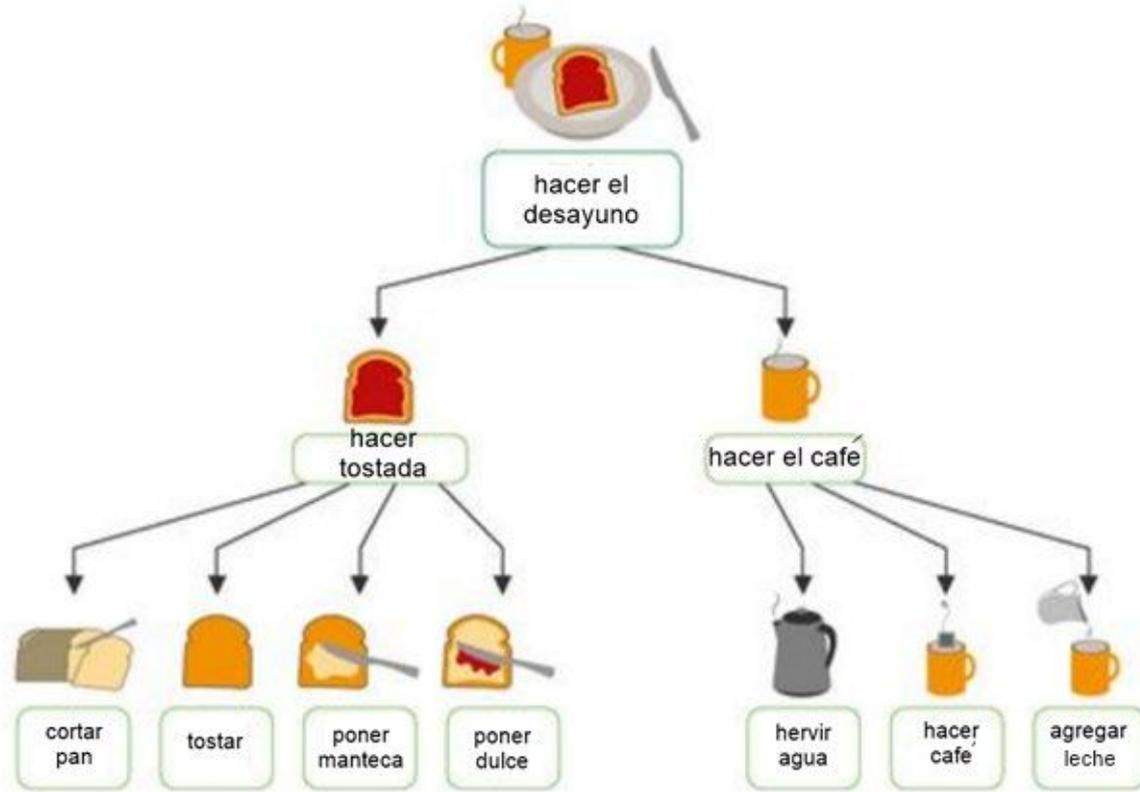
# Técnica Top-Down

Una manera de dividir el problema en partes más pequeñas, es mediante la técnica **Top-Down**, que consiste en un diagrama (cajitas) que grafica las distintas partes del problema en forma de árbol, donde cada cajita representa una porción del problema, el cual se vuelve a dividir en otras partes aún más pequeñas, y así sucesivamente.

Veamos un ejemplo

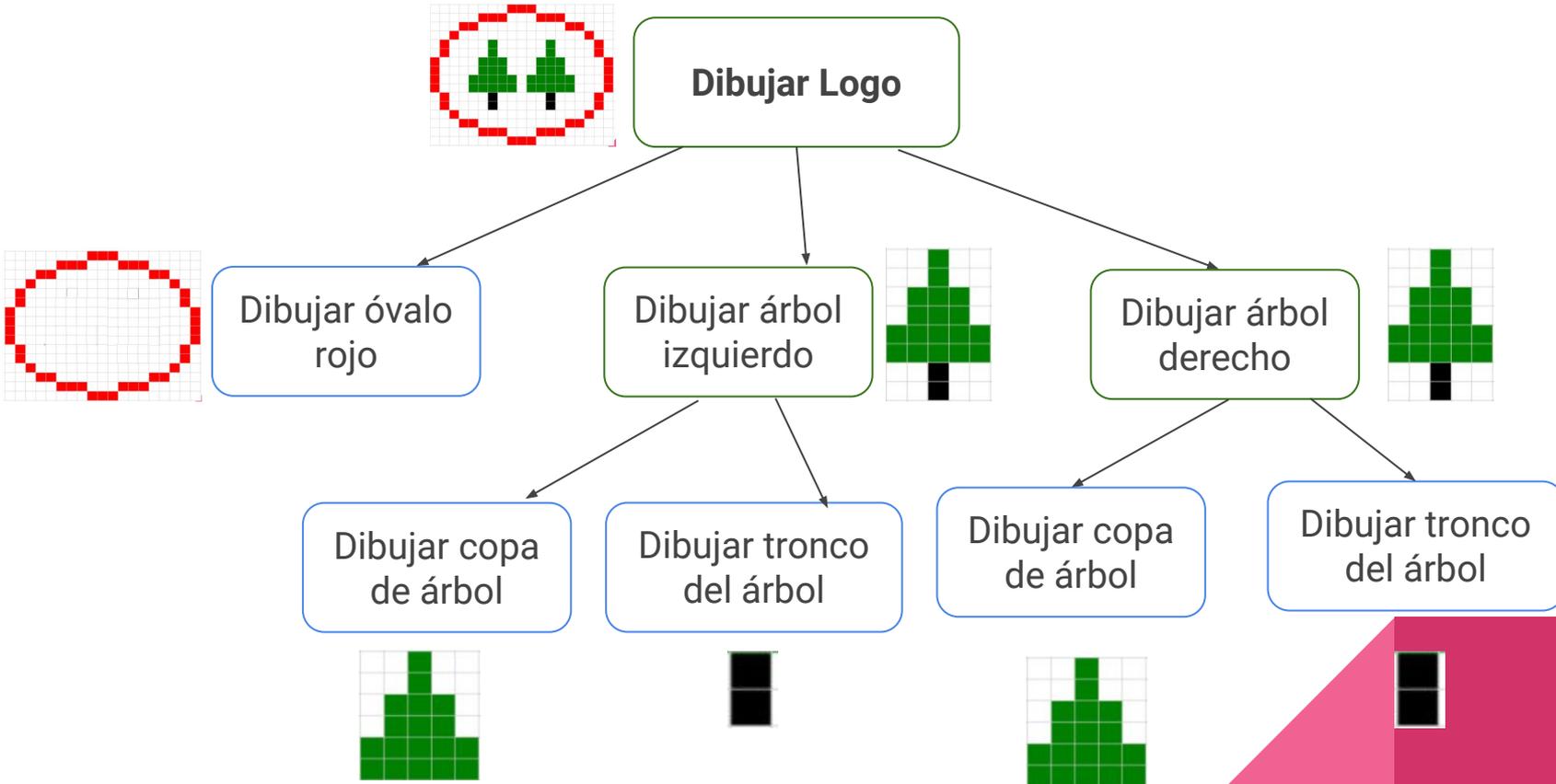


# Hacer el desayuno



**Fuente:** Imágen perteneciente al material proporcionado por [UNIFE](#)

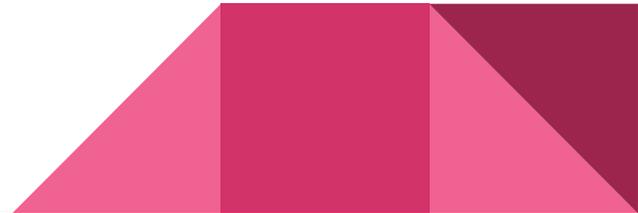
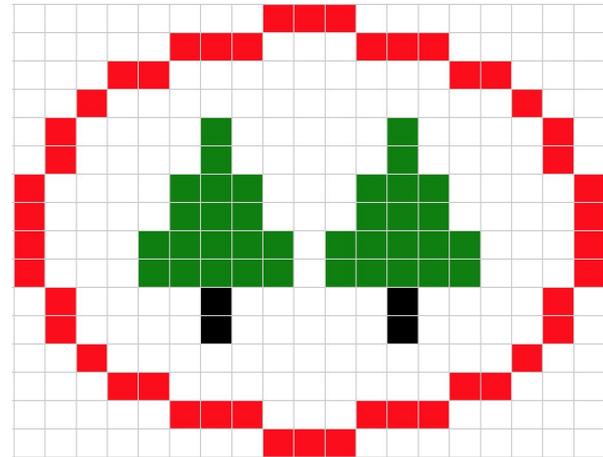
# Volviendo al ejercicio del dibujo del logo



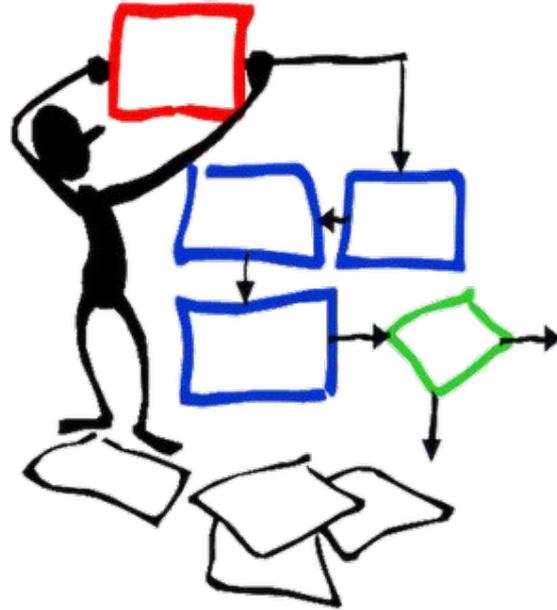
# Escribiendo las acciones de cada problema

Dibujar logo

- ↳ Dibujar óvalo rojo
  - ↳ Arriba, arriba, derecha, pintar de rojo ...
- ↳ Dibujar árbol izquierdo
  - ↳ Dibujar copa del árbol
    - ↳ Pintar de verde, arriba, pintar ....
  - ↳ Dibujar tronco del árbol
    - ↳ Pintar de negro, arriba, pintar ....
- ↳ Dibujar árbol derecho
  - ↳ Dibujar copa del árbol
    - ↳ Pintar de verde, arriba, pintar ....
  - ↳ Dibujar tronco del árbol
    - ↳ Pintar de negro, arriba, pintar ....



# Procedimientos



# Procedimientos

## ¿Qué son?

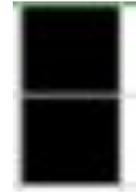
Los procedimientos son una **forma de estructurar el código** para reflejar estos esquemas mentales que hemos comentado.

Un **procedimiento** es una **nueva instrucción definida por el usuario** que representa la solución de una parte del problema.

## Sintaxis:

Un procedimiento se define mediante la palabra “**procedimiento**”, seguida de un **nombre** (no puede contener espacios y comienza con mayúscula), paréntesis vacíos, la sección de **documentación** (sintaxis de **comentarios**) y su correspondiente **bloque de código** entre llaves.

# Ejemplo



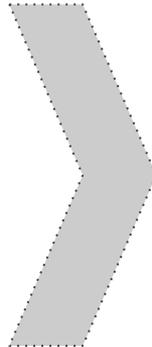
```
procedimiento DibujarTroncoDeArbol () {
```

```
/* PROPÓSITO: Dibujar el tronco de un árbol de color negro, de 2 celdas de alto. El cabezal  
inicia y finaliza en la celda inferior del tronco.
```

```
PRECONDICIÓN: Existe al menos 1 celda hacia arriba de la celda actual.
```

```
*/
```

```
    PintarNegro  
    MoverArriba  
    PintarNegro  
    MoverAbajo  
}
```



Recordar que consiste en **definir una nueva instrucción**, a partir de las instrucciones primitivas del lenguaje Qdraw, donde la secuencia de éstas acciones cumple siempre el mismo propósito

# Ahora tenemos Instrucciones y Procedimientos

**Acciones:** set limitado de instrucciones

- MoverDerecha
- MoverArriba
- MoverIzquierda
- MoverAbajo
- PintarNegro
- PintarRojo
- PintarVerde
- Limpiar

**Procedimientos:** instrucción definida por el usuario

- DibujarTroncoDeArbol ()

# Llamar (invocar) procedimientos

Cada **procedimiento** tiene dos etapas:

- Definición. Ya vimos como es la sintaxis para definir un procedimiento (diapo 15)
- Invocación / llamado. Cómo usamos a los procedimientos que definimos.

**Definición:** se define con un nombre, la sección de documentación y el bloque de código correspondiente al algoritmo que resuelve el problema.

**Invocación:** se llama (**invoca**) a través de su nombre, y desde cualquier bloque del programa o desde otro procedimiento.

**Ejecución:** al momento de **ejecutar el programa**, en el lugar desde donde se invoca al procedimiento, se ejecuta el **bloque de código** correspondiente en su definición.

IMPORTANTE: Un procedimiento **se define una sola vez**, pero se puede **invocar** tantas veces como sea necesario.



# Llamar a procedimientos - Sintaxis - Ejemplo 1

Para llamar a un procedimiento basta utilizar el nombre del mismo seguido de paréntesis.

## Invocación:

```
programa {  
    DibujarTroncoDeArbol()  
    ...  
}
```



## Definición:

```
procedimiento DibujarTroncoDeArbol () {  
    /* */  
    PintarNegro  
    MoverArriba  
    PintarNegro  
    MoverAbajo  
}
```



Estado inicial tablero

(antes de ejecutarse  
el procedimiento)



Estado final tablero

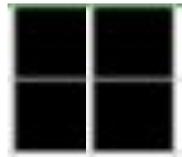
(después de ejecutarse  
el procedimiento)

# Llamar a procedimientos - Sintaxis - Ejemplo 2

Pueden llamarse más de una vez, aunque se define una sola vez.

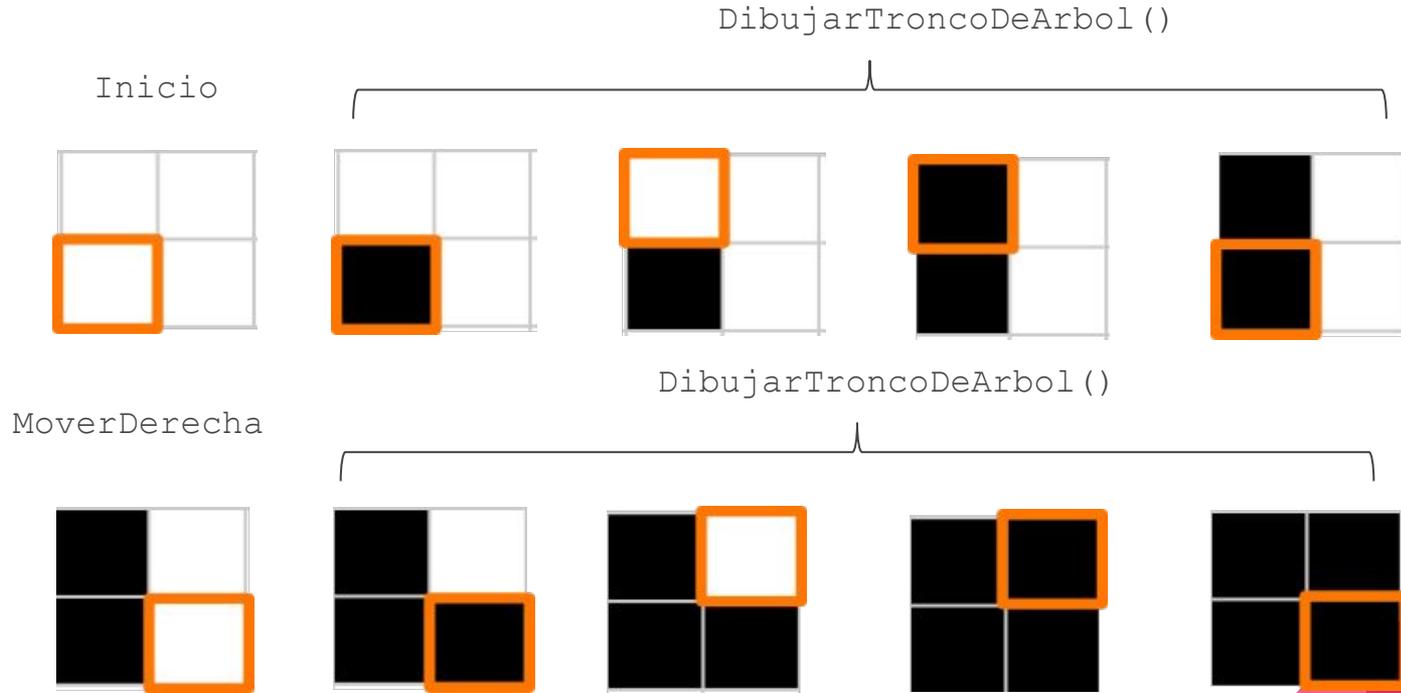
## Invocación desde el programa:

```
programa {  
    DibujarTroncoDeArbol()  
    MoverDerecha  
    DibujarTroncoDeArbol()  
}
```



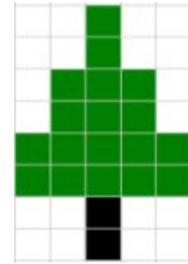
Estado final tablero

# Simulación de la ejecución del programa anterior



# Invocar a un procedimiento desde otro procedimiento

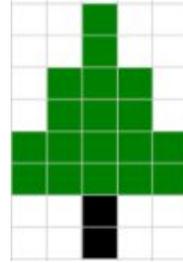
```
procedimiento DibujarArbol () {  
    DibujarTroncoDeArbol()  
    IrACopa()  
    DibujarCopaDelArbol()  
}
```



**Cada procedimiento debe tener su correspondiente documentación**

# Dibujamos el árbol

```
programa{  
    /* ... */  
    DibujarArbol ()  
}
```



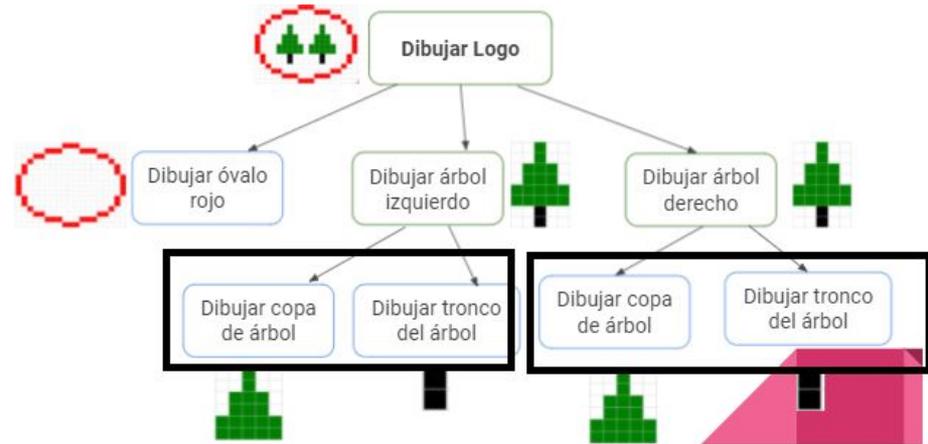
Estado final tablero

**Nota: este programa se resume en llamar al procedimiento principal, y nada más**

# Dibujamos el logo

Si volvemos nuevamente al dibujo del logo, podemos obtener la siguiente solución final del problema:

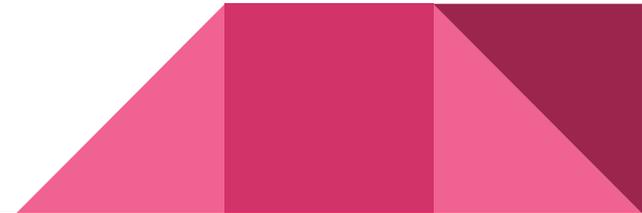
```
programa {  
    DibujarLogo()  
}  
procedimiento DibujarLogo(){  
    DibujarOvalo()  
    IrAlArbolIzquierdo()  
    DibujarArbol()  
    IrAlArbolDerecho()  
    DibujarArbol()  
    VolverAlInicio()  
}
```



Dibujar el árbol izquierdo y derecho son iguales. Podríamos tener una única tarea que nos dibuje el árbol

**Nota. Es necesario definir todos los procedimientos. Y no se olviden de la documentación!**

# Ejercitamos con Lightbot



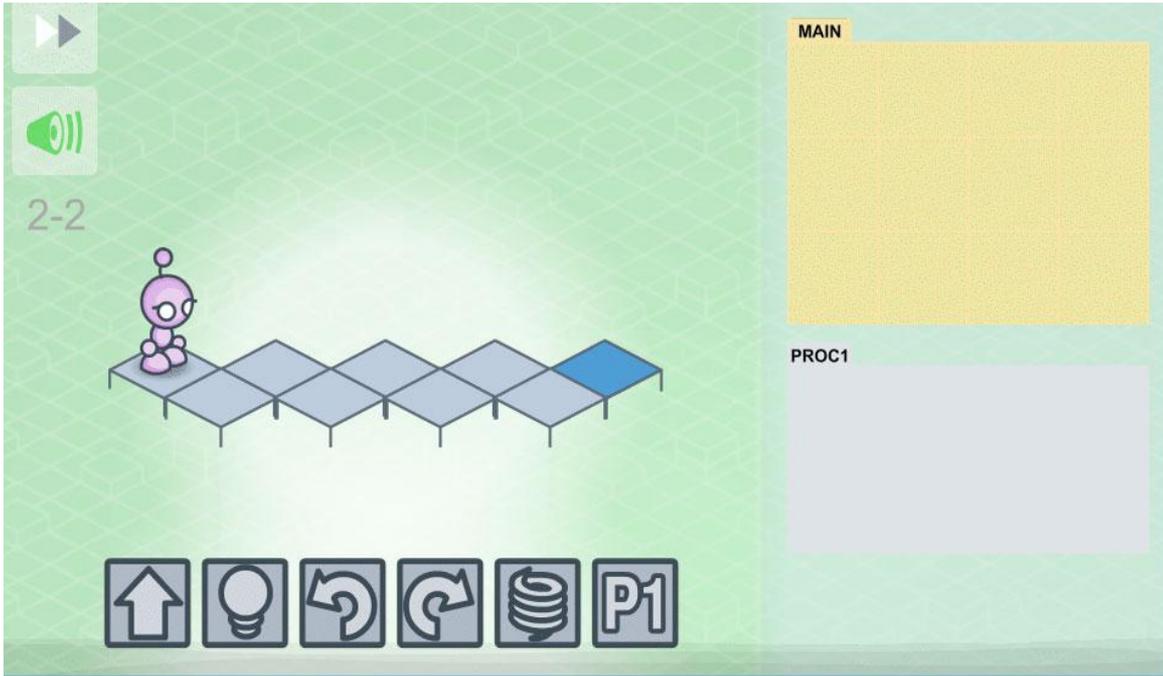
# Nivel 1: Ejemplo

The image shows a game level editor interface. On the left, a 3D maze is displayed with a pink robot character on a platform. The maze consists of grey blocks with blue blocks forming a path. In the top-left corner, there are icons for a play button and a speaker, along with the text '2-1'. At the bottom, a toolbar contains icons for an up arrow, a lightbulb, a return arrow, a circular arrow, a stack of coins, and a 'P1' label. On the right, a programming logic editor is shown with two sections: 'MAIN' and 'PROC1'. The 'MAIN' section contains three 'P1' labels and two circular arrows. The 'PROC1' section contains three up arrows and a lightbulb icon. Below the 'PROC1' section, the text 'Cod P1' is visible with a blue arrow pointing to it. A dashed arrow points from the 'MAIN' section to the text 'Llamada al procedimiento', and another dashed arrow points from the 'PROC1' section to the text 'Definición del procedimiento'.

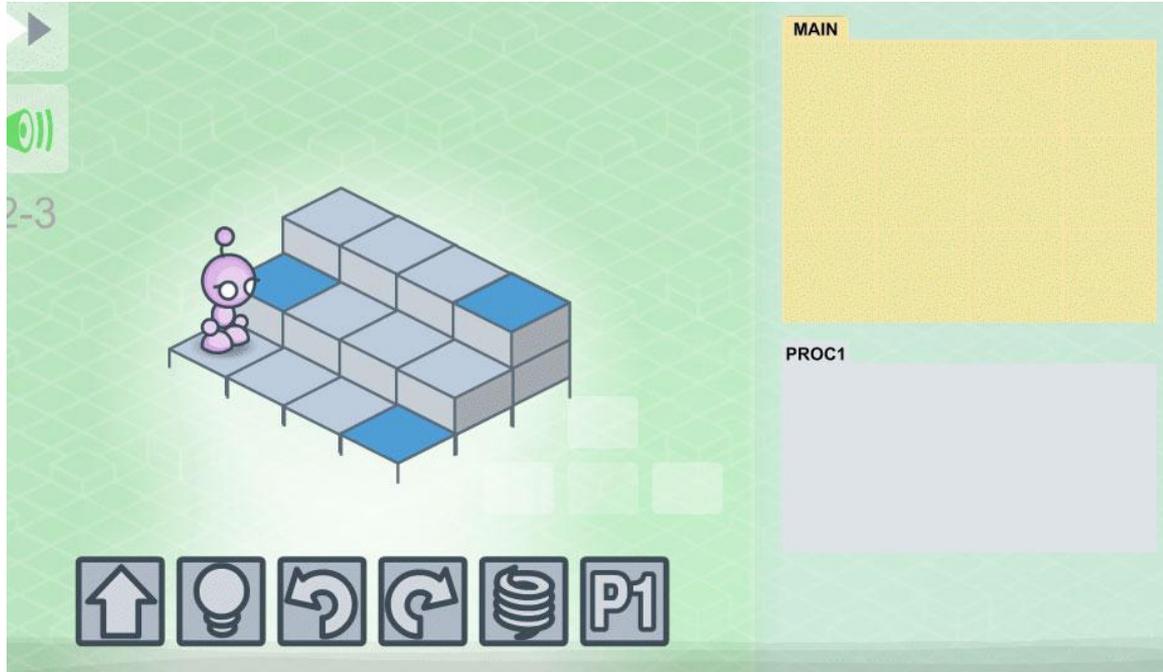
**Llamada al  
procedimiento**

**Definición del  
procedimiento**

# Nivel 2



# Nivel 3

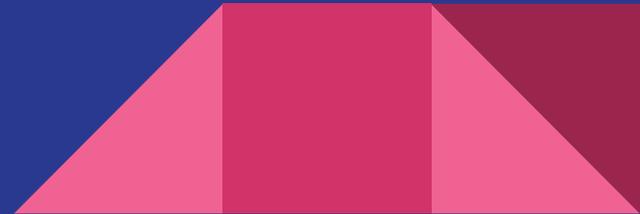
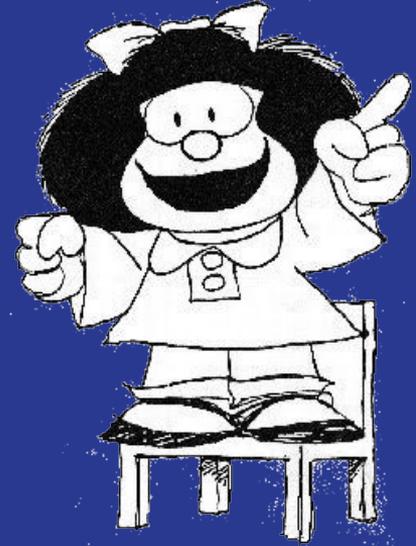


# Nivel 4

The image shows a game level editor interface. On the left, there are navigation and utility icons: a left arrow, a right arrow, a play button, a speaker icon, and the text '2-4'. The main area features a 3D isometric view of a level made of blue and grey blocks, with a small pink robot character standing on a raised platform. At the top, there are icons for undo, redo, a question mark, and a play button. On the right side, there are three panels: 'MAIN' (a yellow textured area), 'PROC1' (a grey rectangular area), and 'PROC2' (another grey rectangular area). At the bottom, there is a toolbar with icons for an upward arrow, a lightbulb, a left arrow, a right arrow, a stack of coins, and two buttons labeled 'P1' and 'P2'.

Para reflexionar...

"Me lo contaron y me lo  
olvidé, lo vi y lo entendí,  
lo hice y lo aprendí"





# Programación

Clase 3

División en subtareas

Universidad Nacional de Quilmes