Programación

Descomposición

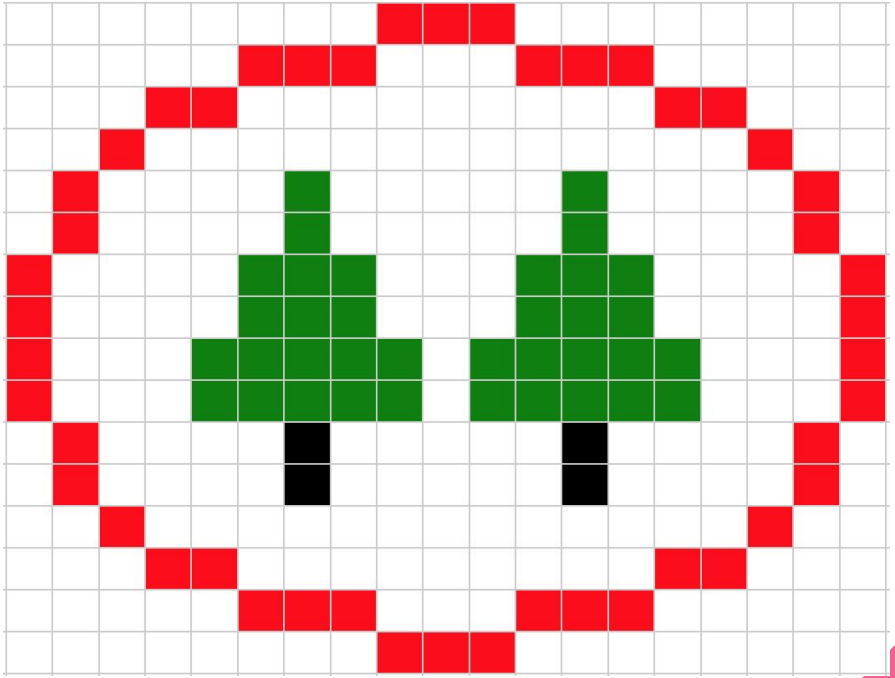
División en subtareas

Universidad Nacional de Quilmes

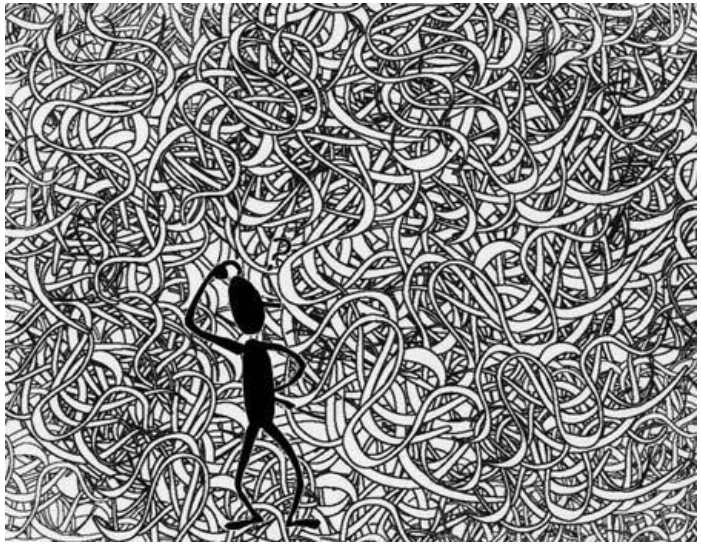
Precalentando motores

Ejemplo: Logo con árboles

Se desea realizar el siguiente dibujo en QDraw:



¿Por dónde arrancamos?



=

Recordemos siempre que **Programar es comunicar**

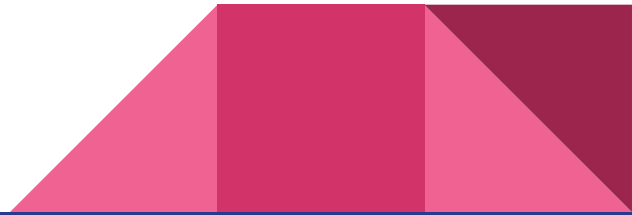
Buscando comunicar la solución

Si quisiéramos indicarle a alguien lo que hay que hacer, terminaríamos diciendo algo como:

mover arriba, mover arriba, mover arriba, pintar rojo, mover arriba, pintar rojo... etc.

Esta solución presenta varios problemas:

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

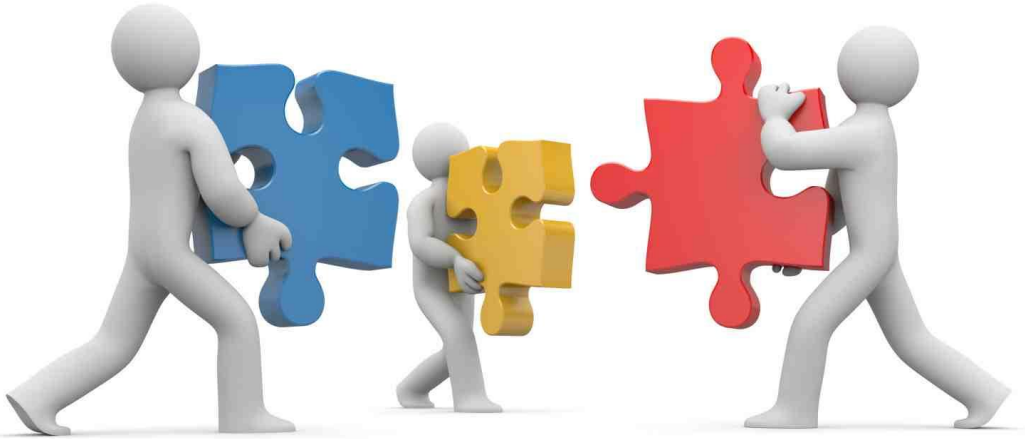


Identificando las partes

Una estrategia más sencilla implica identificar las partes que conforman el dibujo. Quedando algo por el estilo:

dibujar óvalo externo en rojo, dibujar árbol derecho, dibujar árbol izquierdo, volver a la posición inicial

Divide y vencerás



Divide y vencerás

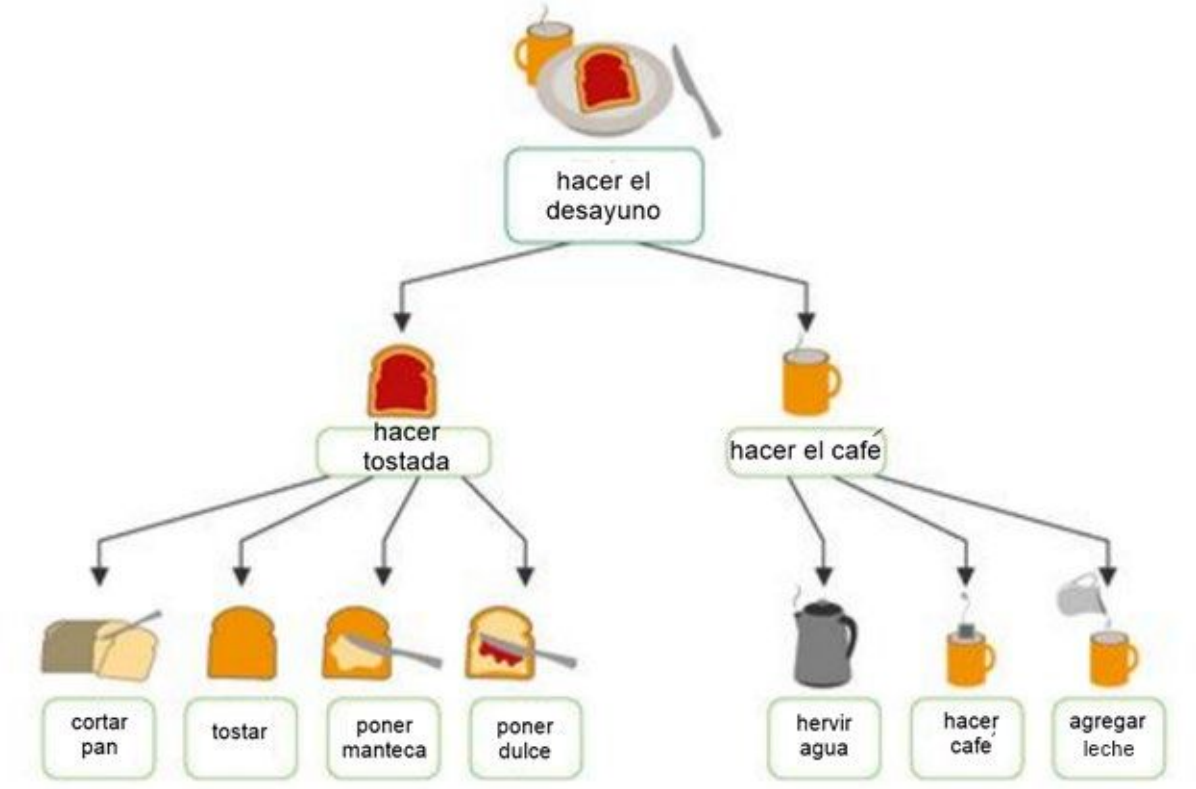
Al dividir el problema general en problemas más pequeños, podemos centrarnos en resolver cosas más sencillas, que requieren menos código y son más fáciles de razonar.

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

Técnica Top-Down

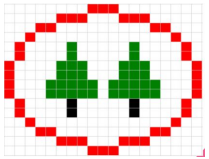
La técnica **Top-Down** es una herramienta gráfica de análisis que nos permite visualizar las distintas partes del problema. Crear un Top-Down consiste en graficar cada parte del problema en forma de árbol (cajitas), donde cada cajita representa una porción del problema, el cual, a su vez, se vuelve a dividir en partes aún más pequeñas, y así sucesivamente. Veamos un ejemplo muy sencillo.

Problema: hacer el desayuno



Fuente: Imágen perteneciente al material proporcionado por UNIPE

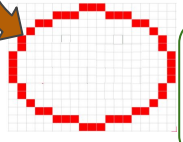
Dibujo Logo con árboles: analicemos sus partes



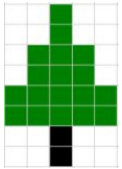
Dibujar Logo

Cada cajita resuelve una parte del problema

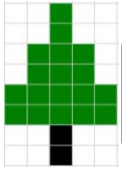
Resultado de ejecutar esta parte del problema



Dibujar óvalo rojo



Dibujar árbol izquierdo



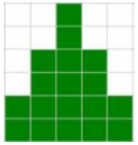
Dibujar árbol derecho

Este problema aún se puede seguir dividiendo

Este problema ya no se puede dividir más

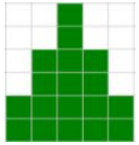
Las cajitas rojas se denominan “Hojas del árbol”. Y son las partes más atómicas (no se pueden dividir)

Dibujar copa de árbol



Dibujar tronco del árbol

Dibujar copa de árbol



Dibujar tronco del árbol

Análisis previo al programa

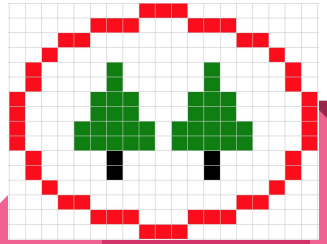
Para pasar del Top-Down al programa, es necesario entender que cada parte del problema (cajita) se deberá asociar a una porción del programa. Por lo que, a continuación, vamos a ver de una manera gráfica cómo pensar dicha traducción.
Nota: esto no es parte del top-down ni del programa, es sólo a modo explicativo.

Las partes grandes que se dividen, contienen instrucciones definidas por las/los programadoras/es, denominadas: PROCEDIMIENTOS.

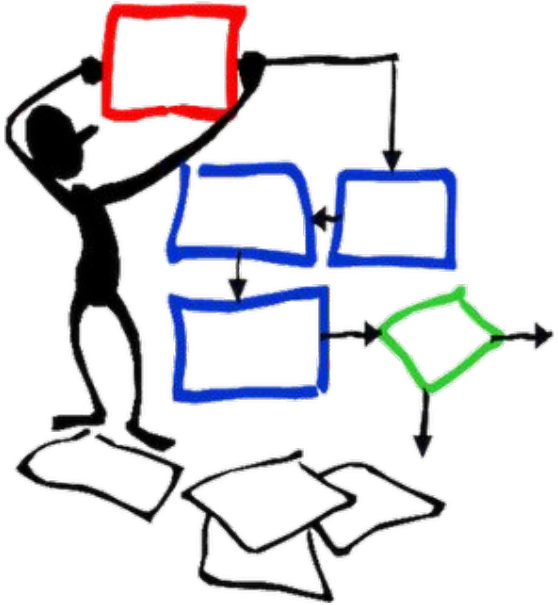
Dibujar logo

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

Las hojas del árbol son las que tendrán las instrucciones primitivas



Procedimientos



Procedimientos

¿Qué son?

Los procedimientos son una **forma de estructurar el código** de manera de reflejar los esquemas mentales que representan la realidad del problema.
Un **procedimiento** es una **nueva instrucción definida por quien programa** que representa la solución de una parte del problema.

Sintaxis:

Un procedimiento se define mediante la palabra reservada "**procedimiento**", seguida de un **nombre** que comienza en mayúscula y sin espacios (*), paréntesis vacíos, la sección de **documentación** (sintaxis de **comentarios**) y su correspondiente **bloque de código** entre llaves.

(*) Si tiene más de una palabra, cada una deberá comenzar con mayúscula.

Ejemplo sencillo

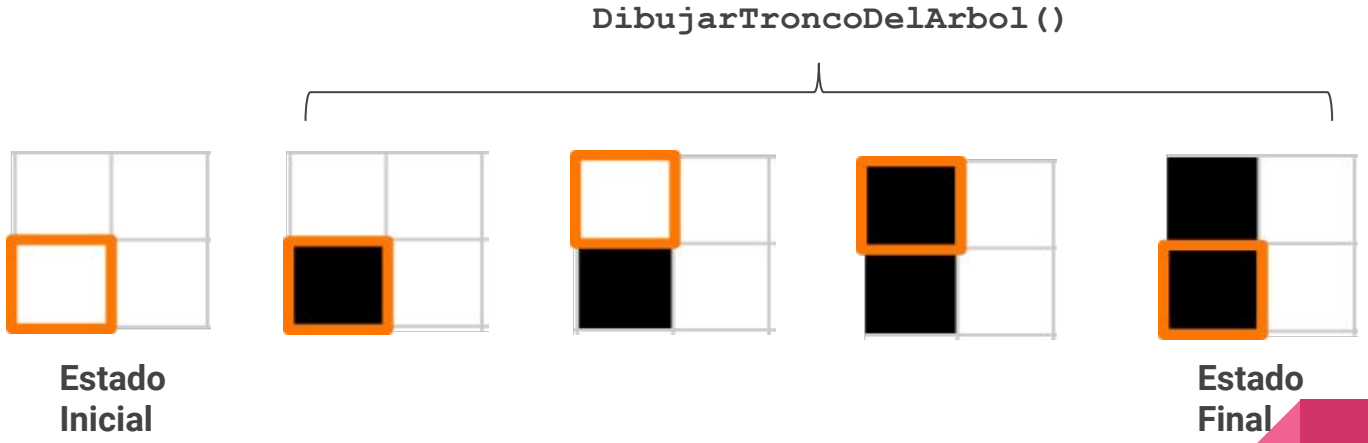
```
procedimiento DibujarTroncoDelArbol () {
/* PROPÓSITO: Dibujar el tronco de un árbol de color negro.
   PRECONDICIÓN: El cabezal inicia en la base del tronco y debe existir al menos 1 celda hacia arriba.*/

    PintarNegro
    MoverArriba
    PintarNegro
    MoverAbajo
}
```

Definimos esta nueva instrucción, cuyo bloque de código contiene las instrucciones primitivas del lenguaje Qdraw, las cuales cumplen con un propósito.
Por lo que es **necesario** que el nombre **comience con un verbo (es una orden).**

Simulación de la ejecución

Veamos la secuencia por la que pasa el cabezal y los distintos cambios de estado del tablero, al momento de ejecutar este procedimiento, como parte del dibujo, dentro de un programa.



Enriqueciendo el repertorio de instrucciones

Ahora nuestro set de instrucciones se compone de primitivas y de los procedimientos que definimos:

Primitivas de QDraw: set limitado de instrucciones

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

Procedimientos: instrucciones definidas por nosotres

- DibujarTroncoDeArbol ()

Invocar procedimientos (llamar)

Cada **procedimiento** pasa por dos etapas**:**

- Definición
- Invocación / llamado: nos permite ejecutar su bloque de código desde cualquier porción del programa que necesitemos.

Definición: se define con un nombre, la sección de documentación y el bloque de código correspondiente al algoritmo que resuelve ese problema (Ver diago 15).

Invocación: se invoca (llama) 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.*

Invocar procedimientos - Sintaxis - Ejemplo

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

Invocación:

```
procedimiento DibujarArbol() {
/* */
        DibujarTroncoDeArbol()
        ...
}
```

Definición:

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

Al invocarlo se ejecuta el bloque de código de su definición.
En este caso se invoca desde un procedimiento.

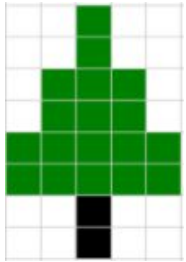
Estado inicial tablero

(Antes de ejecutarse el procedimiento)

Estado final tablero

(Después de ejecutarse el procedimiento)

Ejemplo con 2 procedimientos



```
procedimiento DibujarArbol () {
/*PROPÓSITO: dibujar un árbol tipo pino. El cabezal finaliza en la punta de la copa del árbol.
PRECONDICIÓN: El cabezal inicia en la base del tronco del árbol y debe haber 7 celdas hacia arriba.*/
        DibujarTroncoDeArbol()
        IrACopa()
        DibujarCopaDelArbol()
}
```

Cada procedimiento debe tener su correspondiente documentación

Se invocan a 2 procedimientos distintos. Las 2 partes del árbol según nuestro Top-Down

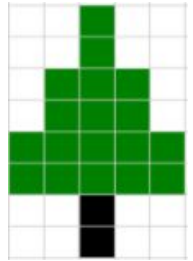
Dibujamos el árbol: Ejecución del programa

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

El programa se resume en invocar al procedimiento principal. No necesita más código

Se invoca desde el programa, en lugar de otro procedimiento

Resultado producto de la ejecución del programa.



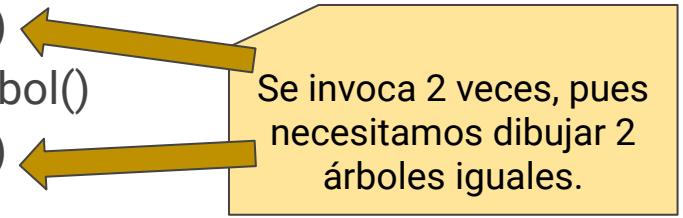
Estado final tablero

¡No ejecutamos procedimientos individuales. Sólo programas!
Los procedimientos se van ejecutando al invocarse dentro del programa.

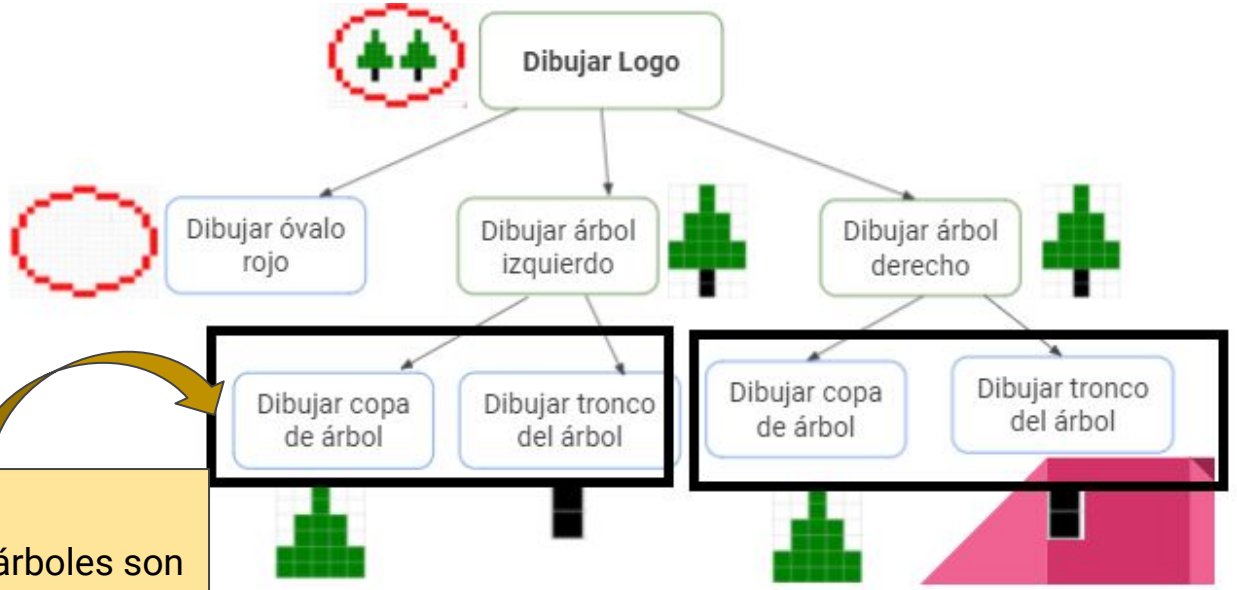
Dibujo con 2 árboles

Muchas veces necesitamos ejecutar más de una vez una parte del problema, y no es muy eficiente, en términos de programación, duplicar su código.
Pero esto se puede evitar mediante la definición de procedimientos.
Lo definimos una sóla vez y lo invocamos tantas veces como sea necesario.

```
programa {
/*PROPÓSITO: dibujar 2 árboles tipo pino. El cabezal finaliza en la punta de la copa
de 2do árbol.
PRECONDICIÓN: El cabezal inicia en la base del tronco del 1er árbol y
debe haber 7 celdas hacia arriba y 6 celdas hacia su derecha.*/
    DibujarArbol()
    IrAProximoArbol()
    DibujarArbol()
}
```



Analizando el Top-Down



Viendo que ambos árboles son iguales (tienen las mismas partes), podemos definir un único procedimiento, e invocarlo 2 veces donde sea necesario.

Programamos el dibujo del logo

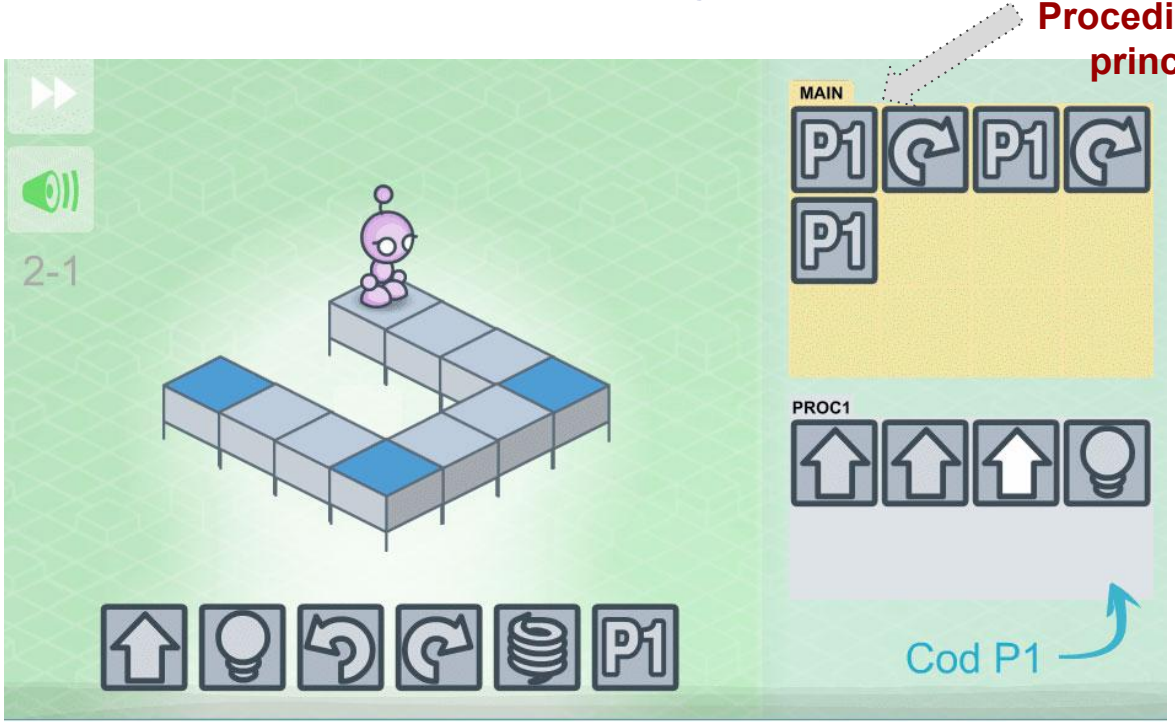
Volviendo al dibujo original del logo con árboles, y según el TOP-DOWN realizado, podemos obtener la siguiente solución final del problema:

```
programa {
/**/
      DibujarLogo()
}
procedimiento DibujarLogo(){
/*PROPÓSITO: dibujar un logo con 2 árboles tipo pino.
PRECONDICIÓN: El cabezal inicia en el centro del extremo superior del óvalo y debe haber 9 celdas hacia su derecha, 9 celdas hacia su izquierda y 15 celdas hacia abajo.*/
      DibujarOvalo()
      IrAlArbolIzquierdo()
      DibujarArbol()
      IrAlArbolDerecho()
      DibujarArbol()
      VolverAlIncio()
}
```

Siendo que el programa sólo ejecuta un procedimiento, su documentación es la misma. De manera que no es necesario agregarla en el programa, pero sí en la definición del procedimiento.

Analogía con Lightbot

Nivel 1: Ejemplo

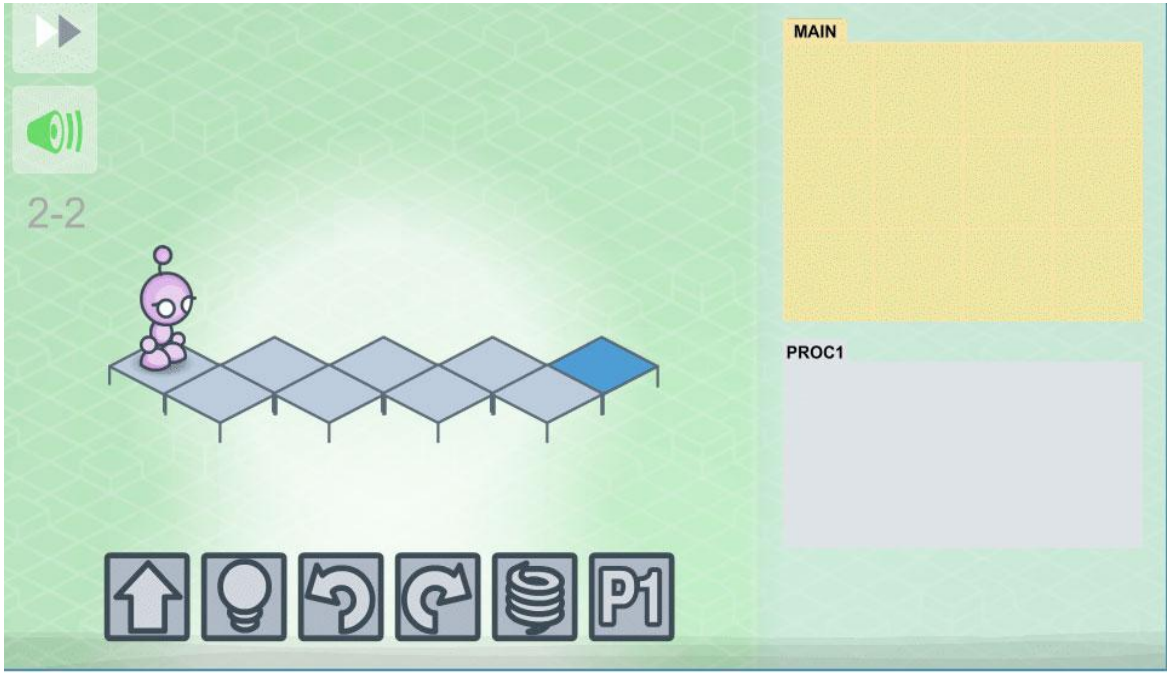


Procedimiento principal

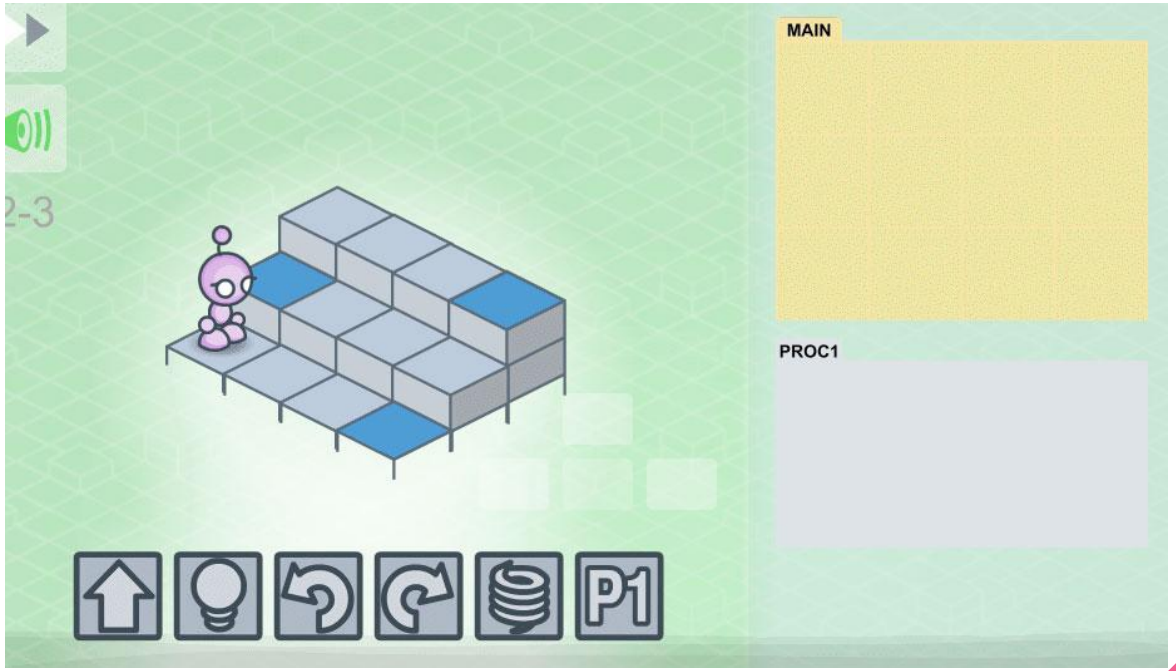
Invocación al procedimiento P1

Definición del procedimiento P1

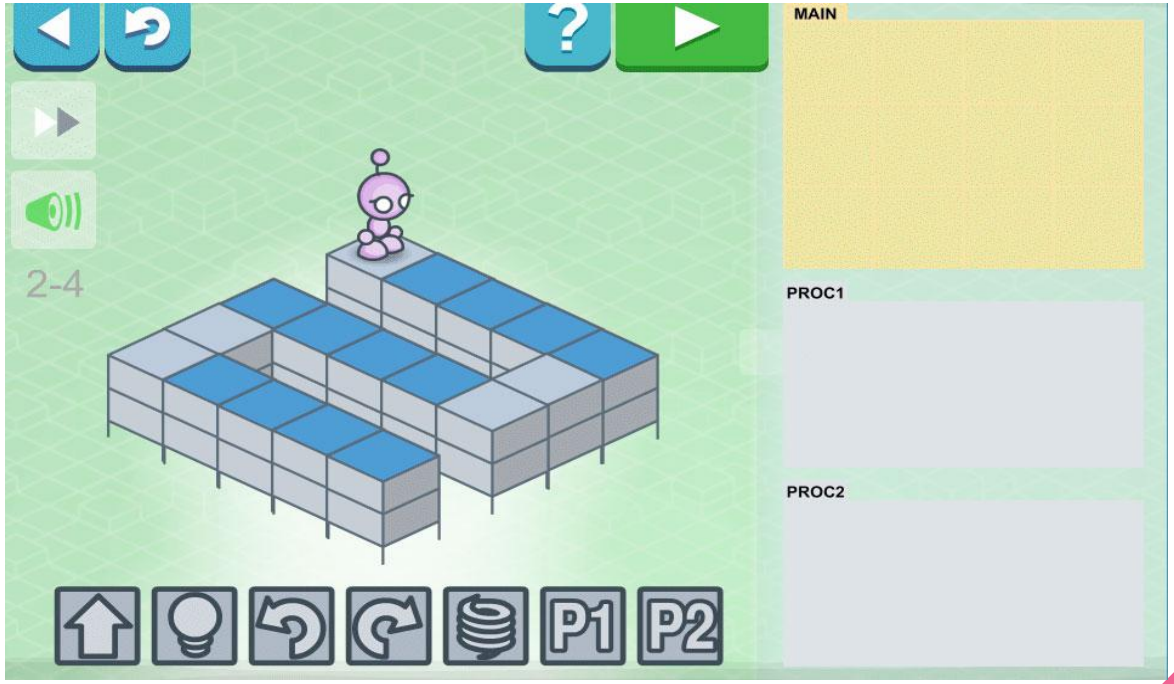
Nivel 2



Nivel 3

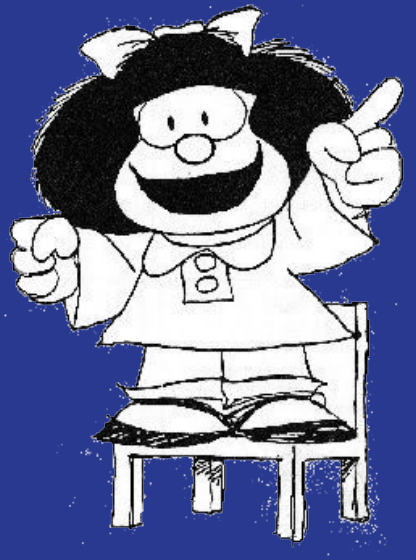


Nivel 4



Para reflexionar...

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



Programación

División en subtareas

Universidad Nacional de Quilmes