Programación

Clase 7

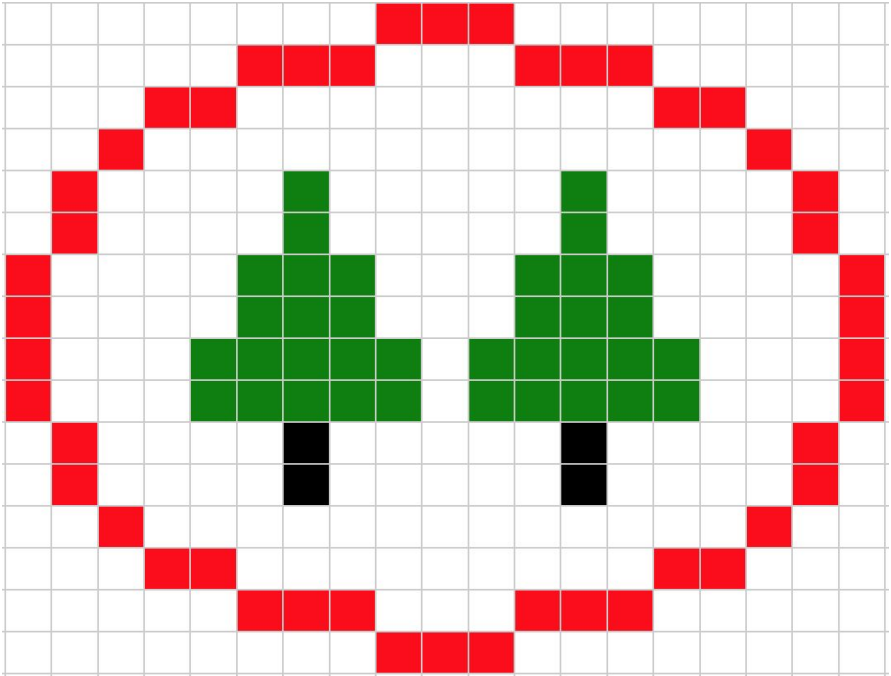
División en subtareas

Universidad Nacional de Quilmes

Ejercitamos un poco

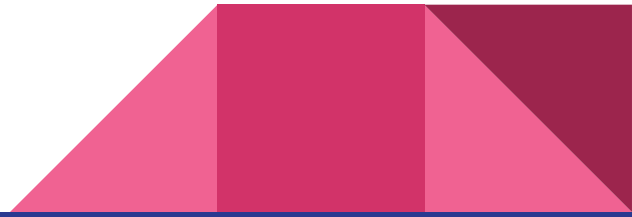
Ejercicio

Realicemos el siguiente dibujo en QDraw.



¿Por dónde arrancamos?

Recordatorio:
¡Programar es comunicar!



Intentando comunicar la solución

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

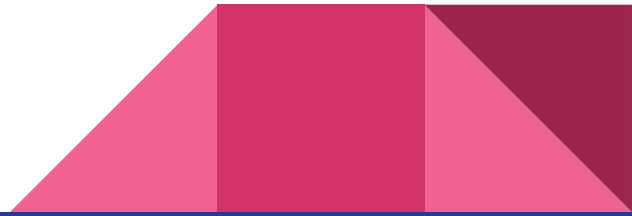
arriba, arriba, arriba, pinta, arriba, pinta... etc.

No es algo muy práctico que digamos.

- El código es confuso hasta para nosotros 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

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



Divide y vencerás

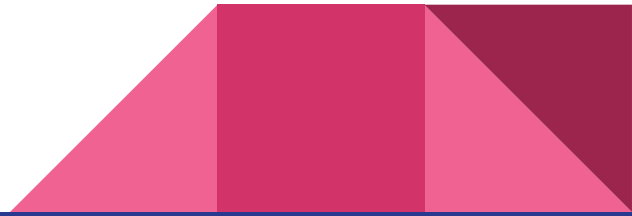
Divide y vencerás

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.

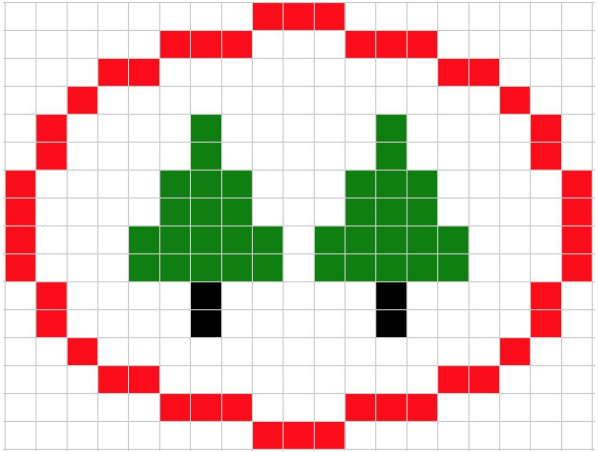
Divide y vencerás. Ejemplo 1

¿Qué pasos o acciones realizan cada vez que realizan la actividad de “cepillarse los dientes”?



Divide y vencerás. Ejemplo 2

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



Procedimientos

Procedimientos

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**.

Un procedimiento se define mediante la palabra "**procedimiento**", seguida de un **nombre** (El cual no puede contener espacios y comienza con mayúscula), paréntesis vacíos y un **bloque de código**.

Procedimientos

```
procedimiento DibujarTroncoDeArbol () {
    /*
    PROPÓSITO: Dibuja el tronco de un árbol de color negro, de dos
        celdas de alto, con la celda actual siendo la celda inferior del tronco.
    PRECONDICIÓN: Hay al menos una celda hacia arriba de la actual.
    */
    ▨↑▨↓
}
```

Instrucciones: Primitivas y Procedimientos

Acciones primitivas

Conjunto limitado de instrucciones:

- →
- ↑
- ←
- ↓
- ▨
- ▨
- ▨
- X

Procedimientos

- DibujarTroncoDeArbol ()

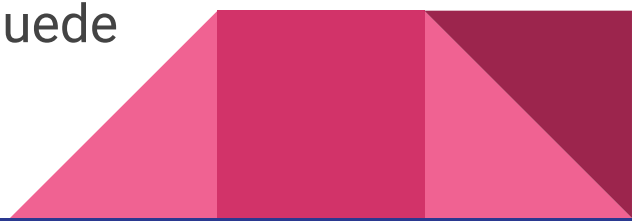
Llamar a procedimientos

Los **procedimientos definidos** pueden ser **llamados** luego en cualquier lugar del programa.

Al momento de **ejecutar el programa**, en los lugares en los que se llama a un procedimiento se ejecuta el **bloque de código con el cual fue definido**.

Un procedimiento puede ser **llamado en cualquier bloque de código,** ya sea el del programa o el de otro procedimiento.

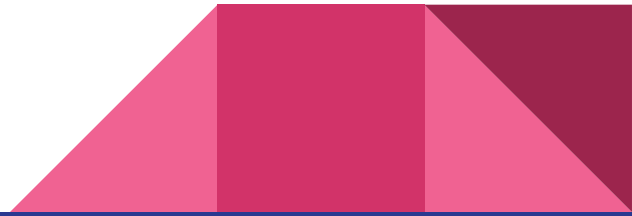
Un procedimiento **se define una sola vez**, pero se puede **llamar tantas veces como se desea**.



Llamar a Procedimientos

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

```
procedimiento DibujarTroncoDeArbol () {
    /* ... */
    ▨↑▨↓
}

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

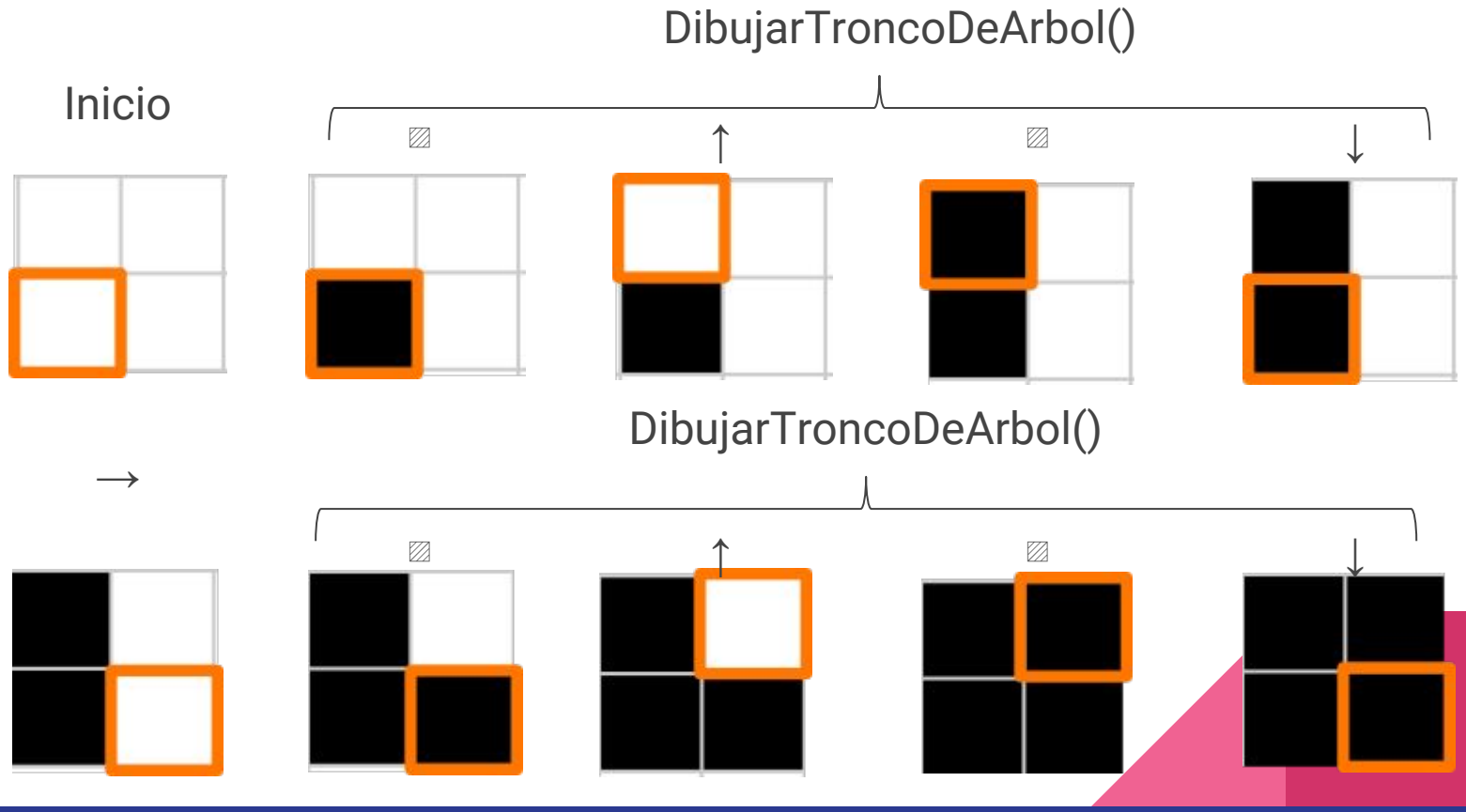


Llamar a Procedimientos

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

```
procedimiento DibujarTroncoDeArbol () {
    /* ... */
    ▨↑▨↓
}

programa {
    DibujarTroncoDeArbol()
    →
    DibujarTroncoDeArbol()
}
```

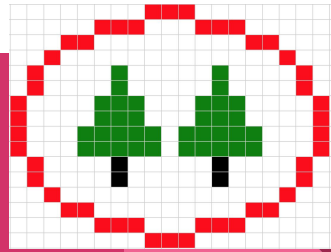
Ejecución del programa anterior



Llamar a Procedimientos en otro Procedimiento

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

procedimiento IrACopa()
        ←←↑↑
}

procedimiento Linea2Verde () {
        ▨↑▨↓
}

procedimiento Linea4Verde () {
        ▨↑▨↑▨↑▨↓↓↓
}
```

```
procedimiento DibujarTroncoDeArbol () {
        ▨↑▨↓
}

procedimiento DibujarCopaDelArbol () {
        Linea2Verde() → Linea4Verde() →
        Linea6Verde() → Linea4Verde() →
        Linea2Verde()
}

procedimiento Linea6Verde () {
        ▨↑▨↑▨↑▨↑▨↓↓↓↓↓
}
```



¡Cada procedimiento debería tener su propósito y precondición!

Dibujamos el árbol

```
programa{

 /* ... */

  DibujarArbol ()

}
```

Mi programa se resume a llamar al procedimiento principal, y solo eso.

Árbol de ejecución

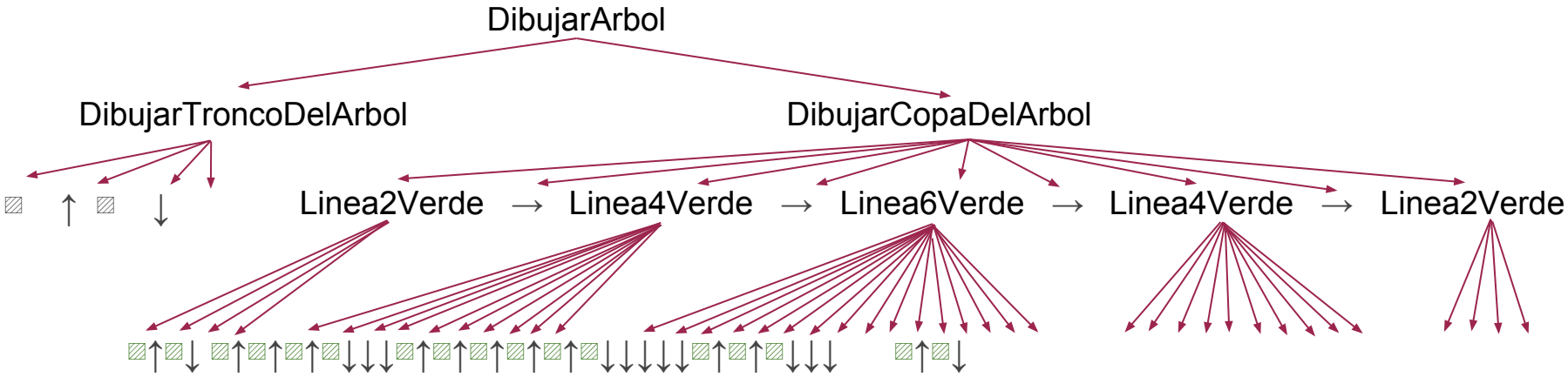
Árbol de ejecución

El **árbol de ejecución** de un programa es una construcción que permite **visualizar y comprender cuál es el orden en que el código final será ejecutado**, y por tanto que acciones se realizarán.

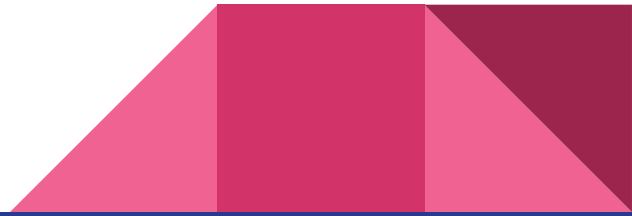
El árbol de ejecución permite ver esto al analizar las **hojas del árbol**, es decir, aquellos lugares en donde se arriba a una **instrucción primitiva**.

Cada elemento (**nodo**) tendrá como **hijos** las instrucciones que haya en el bloque que representa.

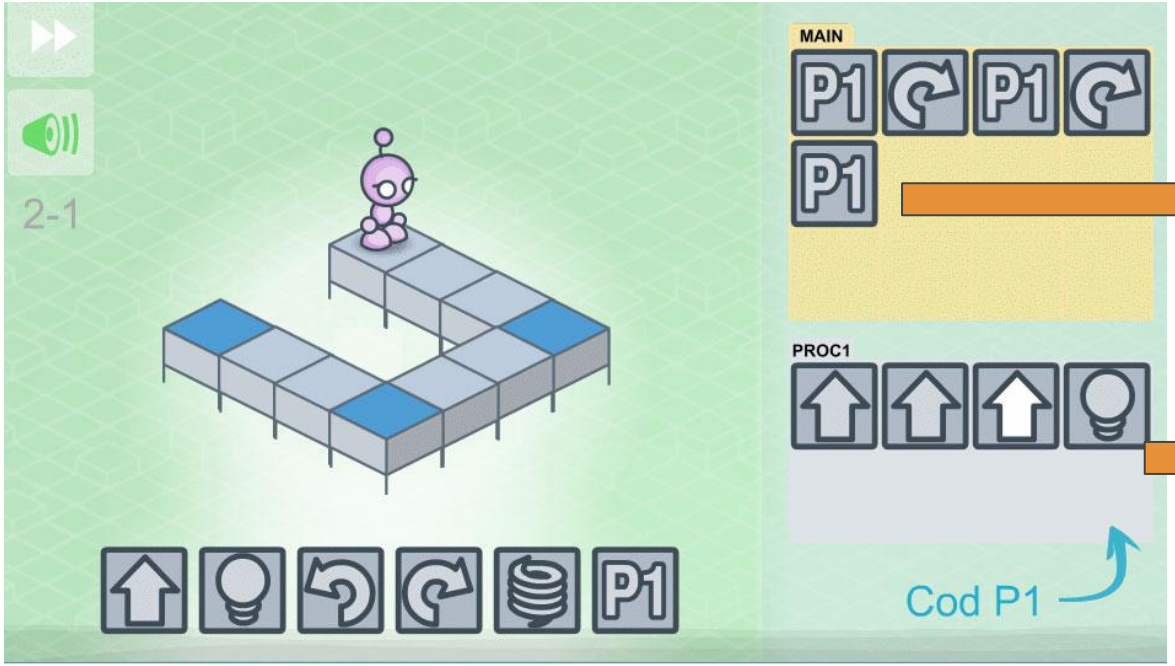
Árbol de ejecución de DibujarArbol



Ejercitamos un poco con Lightbot



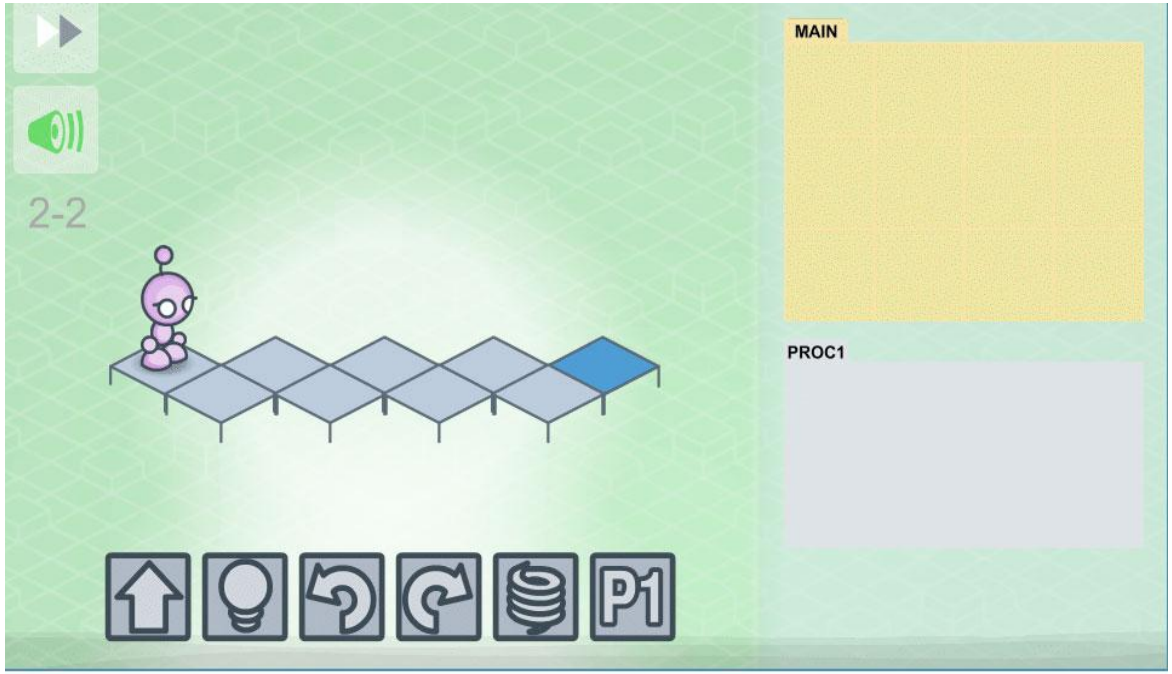
Nivel 1: Ejemplo



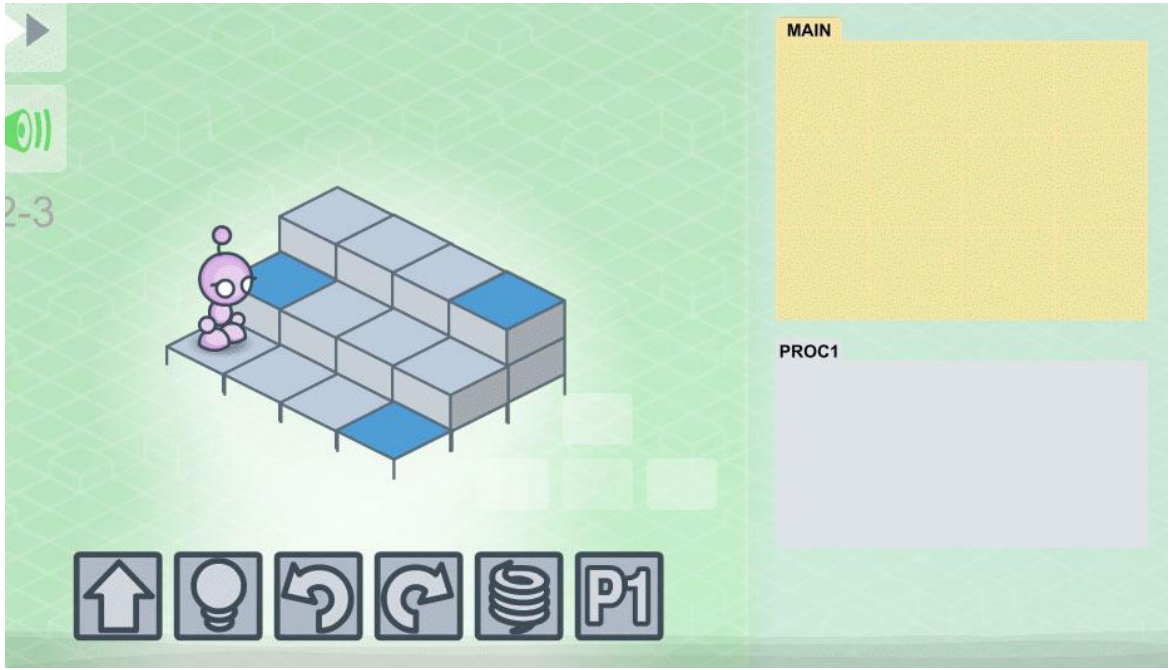
Llamada al procedimiento

Descripción del procedimiento

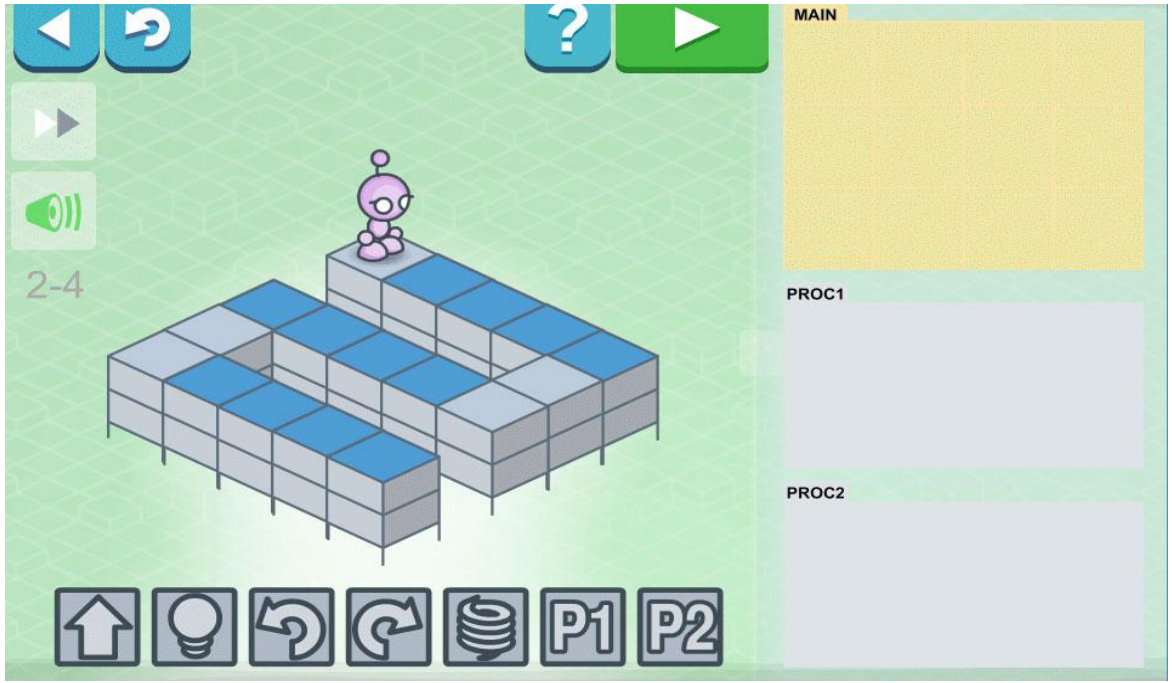
Nivel 2



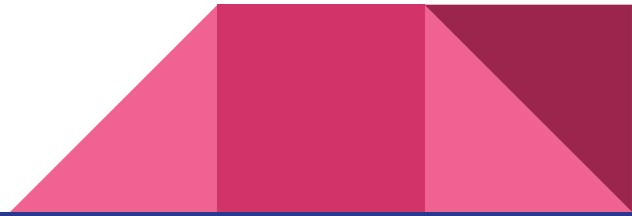
Nivel 3



Nivel 4

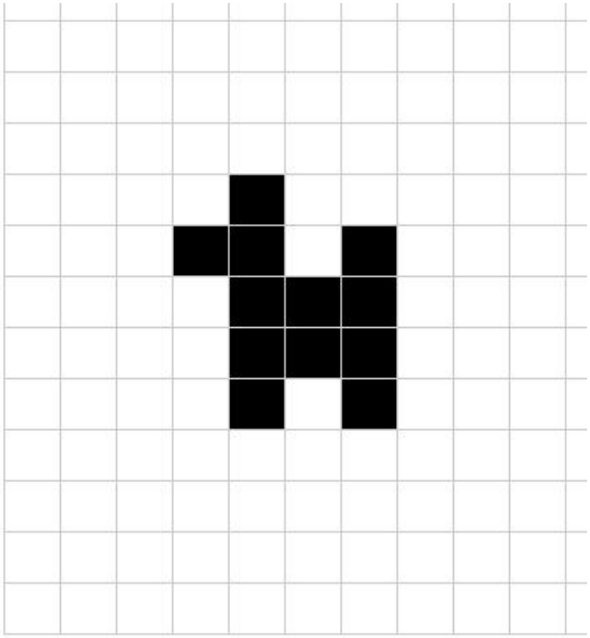


Ejercitamos un poco con QDraw



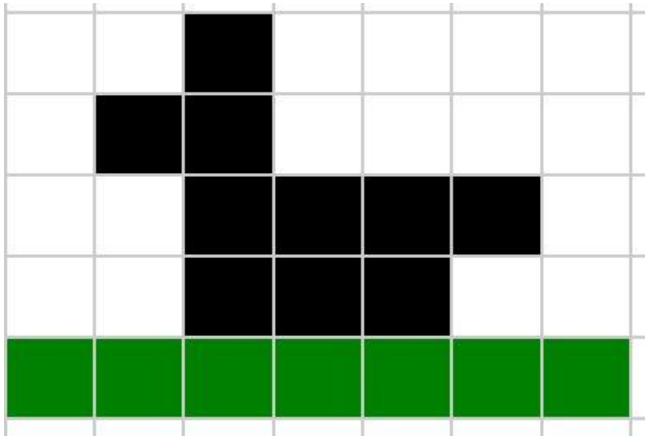
Actividad 1

Implemente un programa que realice el siguiente dibujo utilizando procedimientos.



Actividad 2

Implemente un programa que realice el siguiente dibujo utilizando procedimientos.



¿Encuentran algo parecido al dibujo anterior?

Programación

Clase 7

División en subtareas

Universidad Nacional de Quilmes