



Programación

Repetición simple

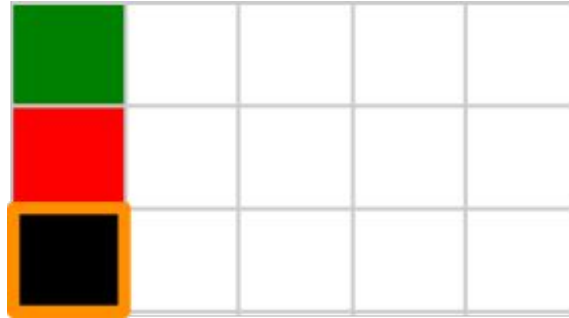
Universidad Nacional de Quilmes

Precalentando motores



Enunciado 1º parte

Realizar el siguiente dibujo con QDraw, teniendo en cuenta que el cabezal se encuentra ubicado en su base.



Primera solución

```
programa {
```

/ **PROPÓSITO:** Dibujar una columna de tres colores: negro en la base, rojo en el medio y verde en el tope.*

***PRECONDICIÓN:** El cabezal inicia en la base de la columna y debe haber al menos 2 celdas hacia arriba*/*

```
    PintarNegro
```

```
    MoverArriba
```

```
    PintarRojo
```

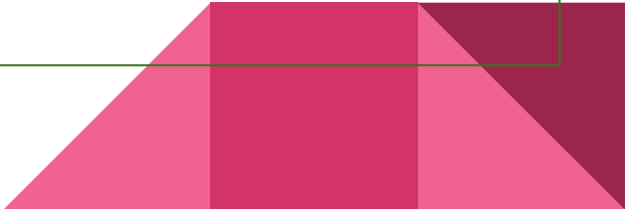
```
    MoverArriba
```

```
    PintarVerde
```

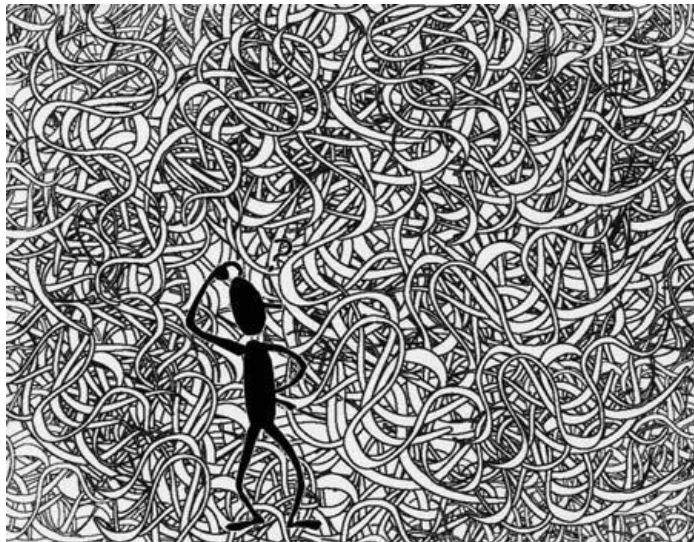
```
    MoverAbajo
```

```
    MoverAbajo
```

```
}
```



¿Y los procedimientos?



Solución mejorada con procedimientos

```
programa {
```

```
  /**/
```

```
    DibujarColumnaMulticolor()
```

```
  }
```

```
procedimiento DibujarColumnaMulticolor () {
```

```
  /* PROPÓSITO: Dibujar una columna de tres colores: negro en la base, rojo en el medio y verde en el tope.
```

```
  PRECONDICIÓN: El cabezal inicia en la base de la columna y debe haber al menos 2 celdas hacia arriba*/
```

```
    PintarNegro
```

```
    MoverArriba
```

```
    PintarRojo
```

```
    MoverArriba
```

```
    PintarVerde
```

```
    MoverAbajo
```

```
    MoverAbajo
```

```
}
```



2º parte: Extendemos el dibujo

Ahora nos pidieron realizar el siguiente dibujo:



Extendemos la solución del algoritmo anterior

```
programa {
```

```
  /* PROPÓSITO: Dibujar un rectángulo multicolor. El cabezal finaliza en la  
  esquina inferior derecha del rectángulo.
```

```
  PRECONDICIÓN: El cabezal inicia en la esquina inferior izquierda del rectángulo  
  y debe haber 2 celdas hacia arriba y 4 celdas hacia su derecha.*/
```

```
    DibujarColumnaMulticolor()
```

```
    MoverDerecha
```

```
    DibujarColumnaMulticolor()
```

```
    MoverDerecha
```

```
    DibujarColumnaMulticolor()
```

```
    MoverDerecha
```

```
    DibujarColumnaMulticolor()
```

```
    MoverDerecha
```

```
    DibujarColumnaMulticolor()
```

```
}
```



Si bien invocamos al
procedimiento varias
veces, notamos que
hay un patrón que se
repite

3° parte: Subimos la apuesta, extendiendo el dibujo

Ahora qué pasa si necesitamos extenderlo aún más, de la siguiente manera:



Extendemos la solución del algoritmo anterior

```
programa {
```

$$/* \dots */$$
[illegible]

}

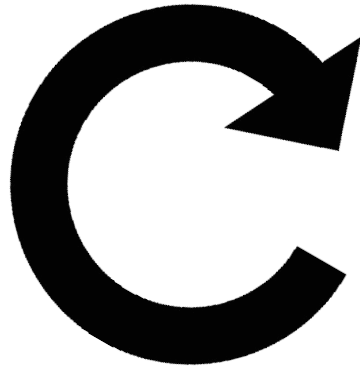
[illegible][illegible]

YA NO ES TAN LEGIBLE... ¿VERDAD?

¿Cómo lo resolvemos?



Repetición simple



Definición

La **repetición simple** es un **bloque de código** que se ejecutará de forma consecutiva una determinada cantidad de veces, es decir, se **repetirá un número fijo (N) de veces**.

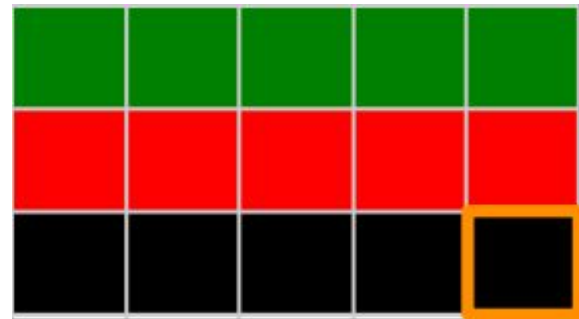
La sintaxis que utilizaremos, se compone de las siguientes palabras reservadas: “**repetir N veces**”, seguido de un **bloque de código** entre llaves, siendo N un **número natural**.



Resolviendo la legibilidad de los programas

```
programa {  
  /*...*/  
  repetir 5 veces {  
    DibujarColumnaMulticolor()  
    MoverDerecha  
  }  
}
```

N



Bloque de
código que se
repite 5 veces

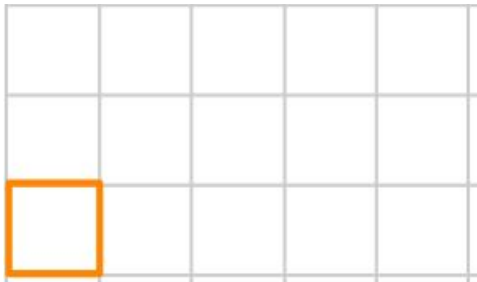
Pero....¿Ejecutaste el código?

¿Resuelve el enunciado solicitado?



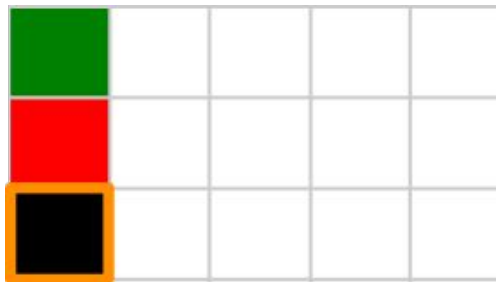
Simulación de la ejecución del código

Inicio



`DibujarColumnaMulticolor()`

`MoverDerecha`



1° ejecución
del bloque
de código
que se repite

`DibujarColumnaMulticolor()`

`MoverDerecha`



2° ejecución
de bloque de
código que
se repite



Simulación de la ejecución del código

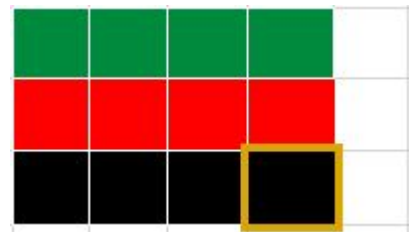
DibujarColumnaMulticolor()
MoverDerecha

3° ejecución
del bloque de
código

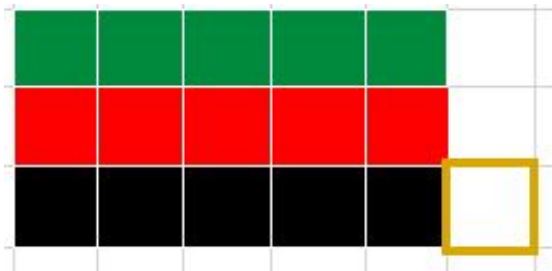


DibujarColumnaMulticolor()
MoverDerecha

4° ejecución
del bloque
de código



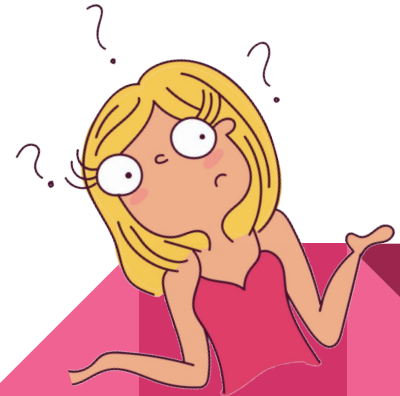
5° ejecución
del bloque
de código



El cabezal finaliza fuera de los límites establecidos del tablero (Error lógico)



¿Cómo lo corregimos?

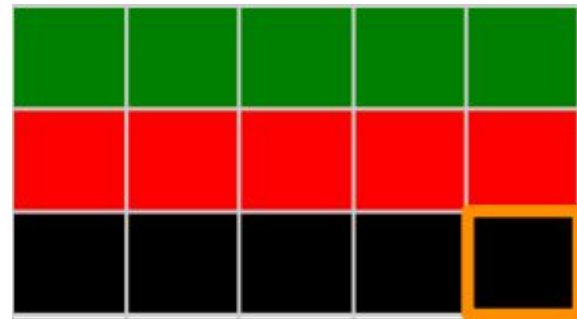


Resolviendo errores lógicos

```
programa {  
  /*...*/  
  repetir 4 veces {  
    DibujarColumnaMulticolor()  
    MoverDerecha  
  }  
  DibujarColumnaMulticolor()  
}
```

N

4



Ejecución de bloque de código que se repite 4 veces

Dibuja la última columna pero sin salir del tablero con la instrucción de movimiento.

Por lo tanto... ¿Cómo resolvemos la legibilidad del enunciado con 30 columnas y respetando los límites del tablero?

Pensalo vos antes de pasar a la siguiente diapositiva.....



Solución final utilizando repeticiones

¡Justo lo que necesitábamos! Aplicamos esta nueva herramienta como solución del ejercicio anterior.

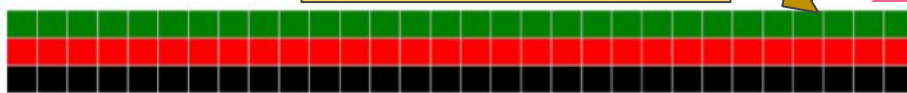
```
programa {
```

/ **PROPÓSITO:** Dibujar una guarda multicolor. El cabezal finaliza en la esquina inferior derecha de la guarda.*

***PRECONDICIÓN:** El cabezal inicia en la esquina inferior izquierda de la guarda y debe haber al menos 2 celdas hacia arriba y 29 celdas hacia su derecha.*/**

```
    repetir 29 veces {  
        ColumnaMulticolor()  
        MoverDerecha  
    }  
    ColumnaMulticolor()  
}
```

¡Analizar cuál es el patrón que se repite en el dibujo! Dicho patrón es el bloque de código que conformará la repetición



Responsabilidad profesional

A medida que necesitamos resolver problemas más complejos, nos encontramos ante la necesidad de ampliar el lenguaje. Para este caso necesitamos agregar una **estructura de control**. Estas estructuras permiten modificar y controlar la secuencia/flujo de las instrucciones planteadas en el **algoritmo**, pero debemos ser responsables en la manera en que las utilizamos. Como programadoras/es debemos **avaluar la calidad** de nuestros programas. Es por ello, que se cuenta con **buenas y malas prácticas de programación**.

Analicemos, mediante un ejemplo, las prácticas que debemos tener en cuenta al momento de utilizar la estructura **Repetir**.

```
programa{  
  /*...*/  
  repetir 2 veces {  
    repetir 2 veces {  
      PintarVerde  
      MoverArriba  
    }  
  }  
}
```

1. ¿Notas algo raro?
2. ¿Te resulta fácil entender cuál es el propósito?
3. ¿Te parece una buena o una mala práctica? ¿Por qué?
4. ¿Cómo lo solucionarías?

Errores comunes: analicemos el código de ejemplo

Respondamos las preguntas que quedaron pendientes:

- 1) ¿Qué notas raro?
- 2) ¿Te resulta fácil entender cuál es el propósito?
- 3) ¿Te parece una buena o mala práctica? ¿Por qué?
- 4) ¿Cómo lo solucionarías?

2) No. No sabemos cuál es. Y el código quizá no expresa lo que se necesita.



```
programa{  
  /*PROPÓSITO: ??*/  
  repetir 2 veces {  
    repetir 2 veces {  
      PintarVerde  
      MoverArriba  
    }  
  }  
}
```

1) Resaltado en rojo



3) MALA PRÁCTICA. Se llama **ANIDAR ESTRUCTURAS DE CONTROL**. No debemos anidar las repeticiones.



4) SOLUCIÓN

1. Primero identificar cuál es el propósito.
2. Luego, buscar el patrón que se repite en el código
3. Reescribir el código utilizando buenas prácticas que reflejen el patrón encontrado y su propósito.

Errores comunes: soluciones aplicando buenas prácticas (Ej1)

Veamos ejemplos de dibujos, con sus respectivas soluciones, que sí aplican buenas prácticas de programación.

Propósito: Se necesita dibujar una sola línea vertical de color verde de 4 de alto



```
programa{  
    repetir 2 veces {  
        repetir 2 veces {  
            PintarVerde  
            MoverArriba  
        }  
    }  
}
```



Unificar la cantidad de repeticiones en una sola.
No pintar de a 2 celdas, sino las 4 celdas en una única repetición.



El patrón: **PintarVerde y MoverArriba** se **repite 3 veces**. El último **PintarVerde** completa la 4ta celda. Se mueve 3 veces y pinta 4 veces.

```
programa{  
    /*PROPÓSITO: dibujar una línea vertical verde de 4 de alto. El cabezal termina en el extremo superior de la línea  
PRECONDICIÓN: El cabezal inicia en el extremo inferior de la línea y debe haber 3 celdas hacia arriba. */
```

```
        repetir 3 veces {  
            PintarVerde  
            MoverArriba  
        }  
        PintarVerde  
    }
```



Errores comunes: soluciones aplicando buenas prácticas (Ej2)

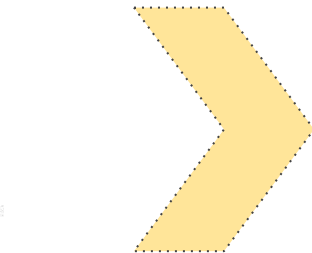


Escribir 2 veces consecutivas la instrucción **repetir** en el mismo bloque de código

```
procedimiento VolverAlInicio() {  
/*...*/  
    repetir 2 veces {  
        MoverAbajo  
    }  
    repetir 2 veces {  
        MoverIzquierda  
    }  
}
```



MALA PRÁCTICA: así como anidar, tampoco debemos colocar **2 estructuras de control consecutivas** en el mismo bloque de código



Solución: nuevamente, **Descomponer** en procedimientos

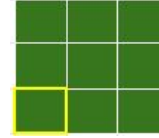
```
procedimiento VolverAlInicio(){  
/*...*/  
    Bajar2veces()  
    Volver2veces()  
}
```



```
procedimiento Bajar2veces{  
/*...*/  
    repetir 2 veces {  
        MoverAbajo  
    }  
}  
  
procedimiento Volver2veces{  
/*...*/  
    repetir 2 veces {  
        MoverIzquierda  
    }  
}
```

Errores comunes: soluciones aplicando buenas prácticas (Ej3)

Ejemplo que contiene las 2 situaciones de malas prácticas que vimos: **anidaciones** y **repetir consecutivos**



programa {

*/*PROPÓSITO: dibujar un cuadrado verde de 3x3. El cabezal finaliza en la esquina superior derecha del cuadrado.*

PRECONDICIÓN: El cabezal inicia en la esquina inferior izquierda y debe haber 2 celdas hacia arriba y 2 hacia su derecha./**

repetir 2 veces {

repetir 2 veces {
PintarVerde
MoverDerecha

}
PintarVerde
repetir 2 veces {
MoverIzquierda

}
MoverArriba

}
repetir 2 veces {
PintarVerde
MoverIzquierda

}
PintarVerde

}

Nuevamente... sí,
procedimientos

¡Código
repetido, poco
legible y difícil
de continuar!

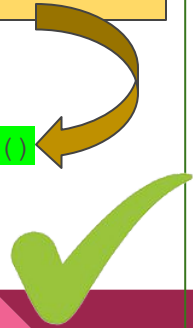
(Revisar los colores
para comprender la
mejora)

programa {
/**/

repetir 2 veces {
DibujarFilaVerde()
VolverAlInicioDeFila()
MoverArriba

}
DibujarFilaVerde()
}

Opción 2: este proc puede
ser parte de
DibujarFilaVerde().
En cuyo caso cambiaría el
estado final del cabezal



¡Repasemos un punto importante!



Retomemos el programa del ejemplo del [slide 24](#) para notar la siguiente diferencia:

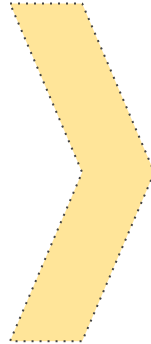
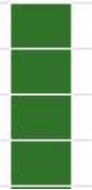


¡En el propósito menciona **4** celdas pero en la instrucción "repetir", el "N" es **3**! (y por lo tanto también la precond.)
¿Por qué?

```
programa{  
  /*PROPÓSITO: dibujar una línea vertical  
verde de 4 celdas de alto. El cabezal termina  
en el extremo superior de la línea.  
PRECONDICIÓN: El cabezal inicia en el  
extremo inferior y debe haber 3 celdas hacia  
arriba. */
```

```
  repetir 3 veces {  
    PintarVerde  
    MoverArriba  
  }  
  PintarVerde
```

```
}
```



**¡Prestar atención a los patrones
que se repiten!**

No confundir la cantidad de celdas
que se quieren pintar, con la
cantidad de veces que se repite el
patrón (bloque de código interno
del repetir)

Resumiendo ...

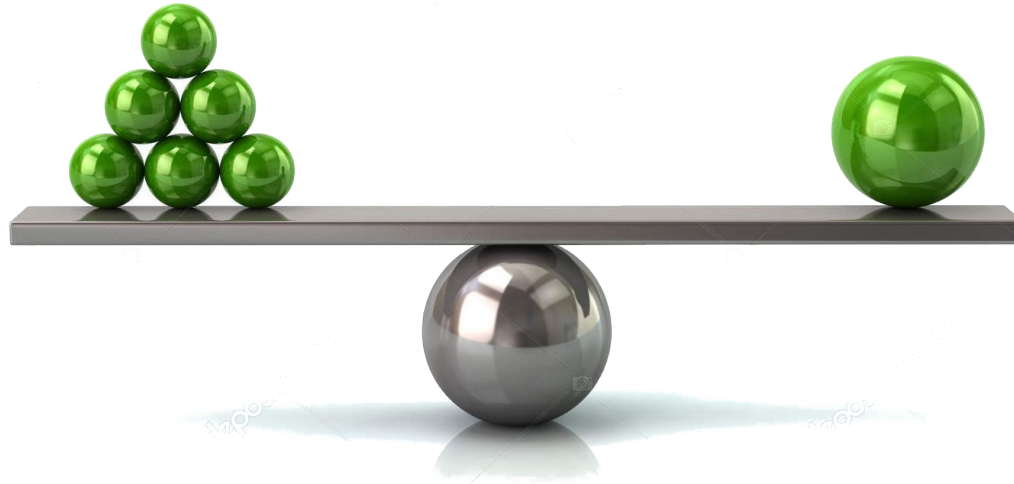
Tanto anidar, como escribir código repetido de manera consecutiva, genera los siguientes problemas:

- Es propenso a cometer errores
- Dificultad para leer el código, complicando así la comunicación, generando un código poco legible
- Dificultad para entender si el código cumple con el propósito
- Código más largo y poco eficiente en términos de la programación

Y por último: ¡Prestar atención a los límites del dibujo con respecto al tablero!



Equivalencias



Comparamos códigos equivalentes

A continuación se muestran ejemplos de programas que cumplen con el mismo propósito. Es decir que su código es equivalente, pero una versión refleja la manera inadecuada y la otra, utiliza las buenas prácticas.

Ejemplo 1:

```
programa {  
  HacerAlgo()  
  HacerAlgo()  
  HacerAlgo()  
}
```



**Ya contamos con una
instrucción que realiza
esta acción.
¡Aprovecharla entonces!**

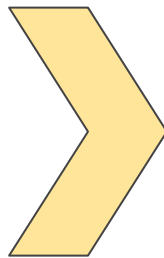
```
programa {  
  repetir 3 veces {  
    HacerAlgo()  
  }  
}
```



Comparamos códigos equivalentes

Ejemplo 2:

```
programa {  
  repetir 1 veces {  
    HacerAlgo()  
  }  
}
```

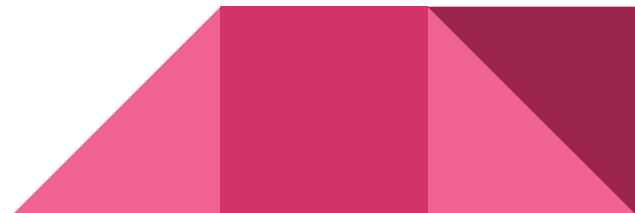


No hay repetición
si se ejecuta sólo
una vez

```
programa {  
  HacerAlgo()  
}
```



Ejercicio para precalentar



Recordar que....



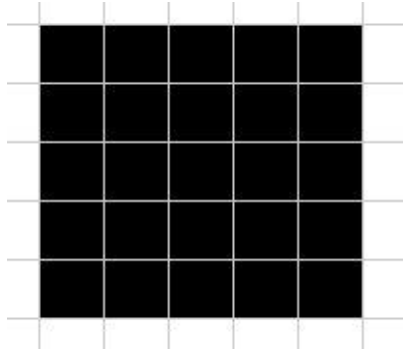
El truco está en encontrar el
patrón adecuado



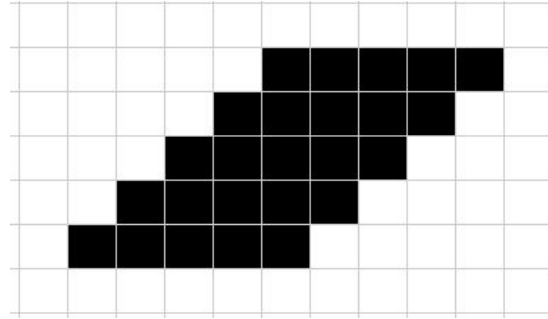
Actividad

Definir los siguientes procedimientos (con su respectiva documentación), que dibuje las siguientes figuras:

DibujarCuadrado

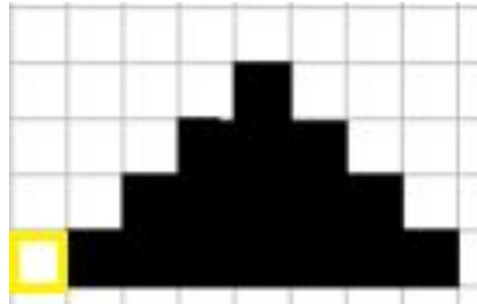


DibujarParalelogramo



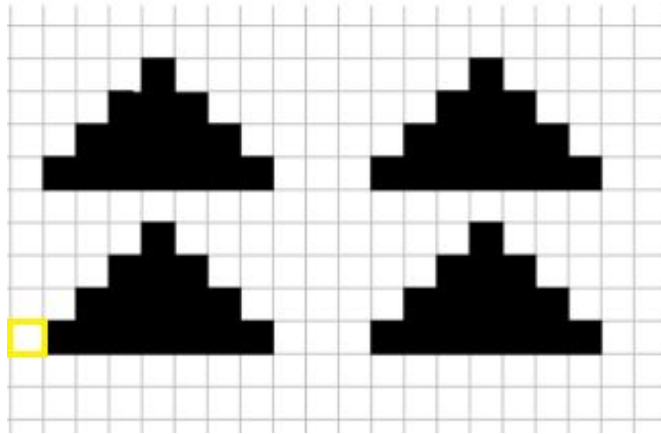
Actividad 2

Definir el procedimiento **DibujarPiramide** que realice el dibujo a continuación. El cabezal comienza a la izquierda de la esquina inferior izquierda de la pirámide.



Actividad 3

Definir el procedimiento `DibujarCuatroPiramides` que realice el dibujo a continuación. El cabezal comienza en la posición pintada de amarillo que se muestra en la imagen.



Para reflexionar...

"Si todo te da igual,
estás haciendo mal las
cuentas"

