



Introducción a la Programación

Clases teóricas

por Pablo E. “Fidel” Martínez López

3. Parámetros y repetición





Repaso



- **Programar es comunicar** (con máquinas y personas)
- Lenguaje de programación (Gobstones)
 - **Comandos**: describen acciones
 - **Expresiones**: describen información
- Programas
 - **Describen transformaciones de estado**
 - Hay infinitos programas **equivalentes**
 - Deben **documentarse** e **indentarse**
 - **Propósito y precondiciones**



- **Procedimientos**

- Definición de nuevos comandos
 - Brindan ***abstracción*** para los comandos
- Permiten **expresar** diversas cosas
 - ***Representación*** de información y primitivas del dominio del problema a solucionar
 - ***Estrategia*** de solución y subtarear
- Aportan legibilidad, claridad y modificabilidad
- Pueden ser reutilizados muchas veces



Repetición simple



- Para hacer una tarea muchas veces,
 - podemos poner muchas veces el mismo comando, o
 - podemos usar algunos procedimientos
 - PERO esto es incómodo y poco generalizable

```
procedure Poner13Rojas() {  
    /* ... */  
    Poner10Rojas()  
    Poner(Rojo)  
    Poner(Rojo)  
    Poner(Rojo)  
}
```

```
procedure Poner16Rojas() {  
    /* ... */  
    Poner10Rojas()  
    Poner5Rojas()  
    Poner(Rojo)  
}
```

¡Resulta difícil generalizar!

```
procedure Poner1500Rojas() {  
    /* ... */  
    // ¿¿¿Cómo hacer???
```



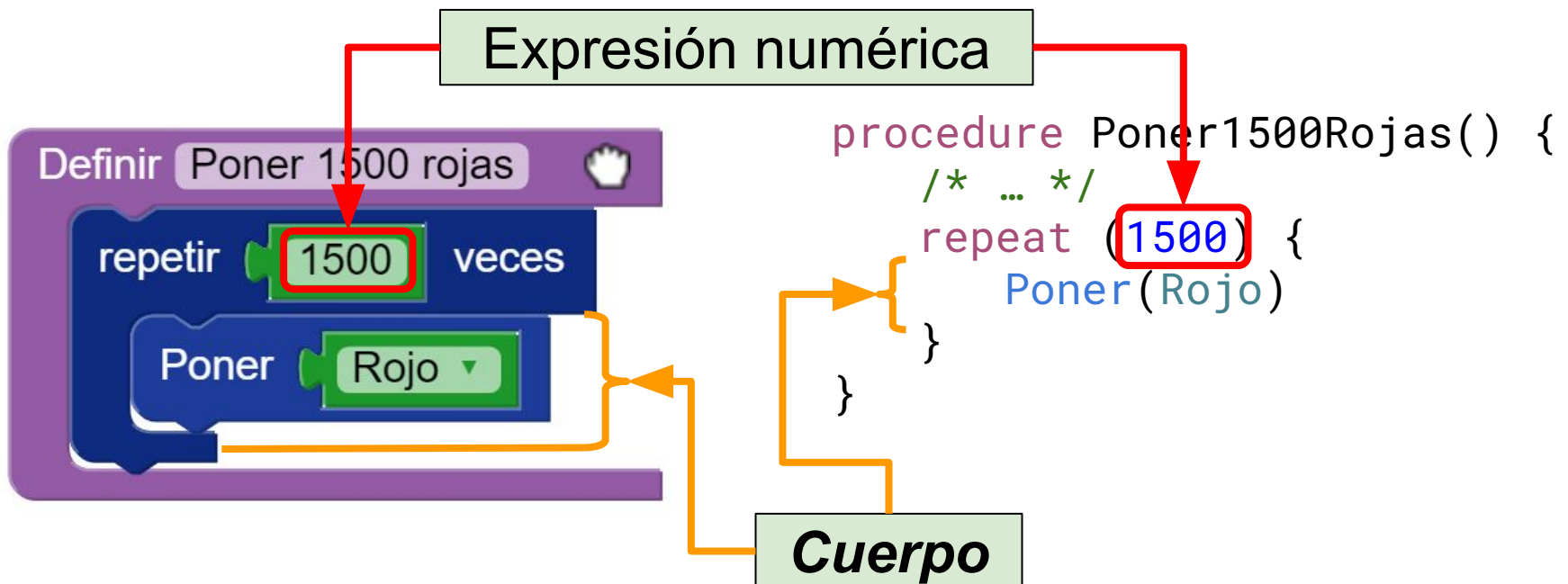
- ¿Cómo mejorar esta situación?
- Precisamos una ***herramienta del lenguaje***
 - ***Repetición simple***
 - Permite repetir un grupo de comandos una cantidad fija de veces



```
procedure Poner1500Rojas() {  
    /* ... */  
    repeat (1500) {  
        Poner(Rojo)  
    }  
}
```

En bloques y texto son parecidas
pero levemente diferentes

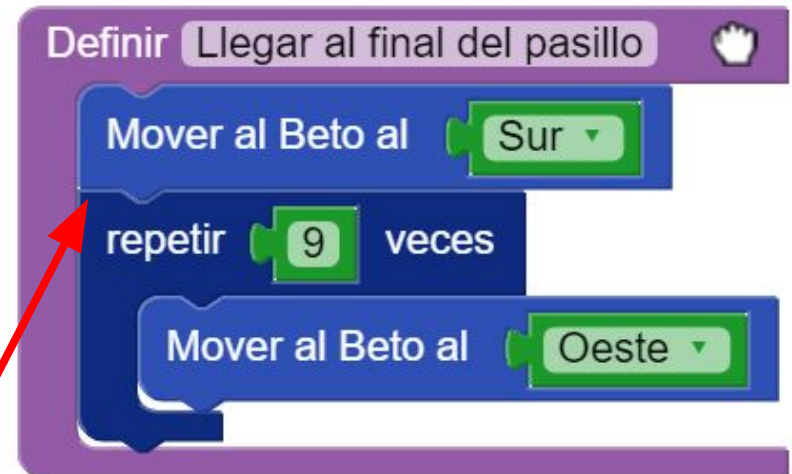
- ¿Cómo se define la repetición simple?
 - En bloques, con el bloque “**repetir _ veces**”
 - En texto, con la palabra clave **repeat**
 - Lleva una expresión numérica (entre paréntesis)
 - Tiene un **cuerpo** (entre llaves)





- Una repetición simple
 - Arma un comando compuesto
 - Observar la forma que tiene el bloque...
 - Por ello se puede usar como otros comandos

```
procedure LlegarAlFinalDelPasillo() {  
    /* ... */  
    MoverAlBetoAl_(Sur)  
    repeat (9) {  
        MoverAlBetoAl_(Oeste)  
    }  
}
```



Secuencia de
comandos

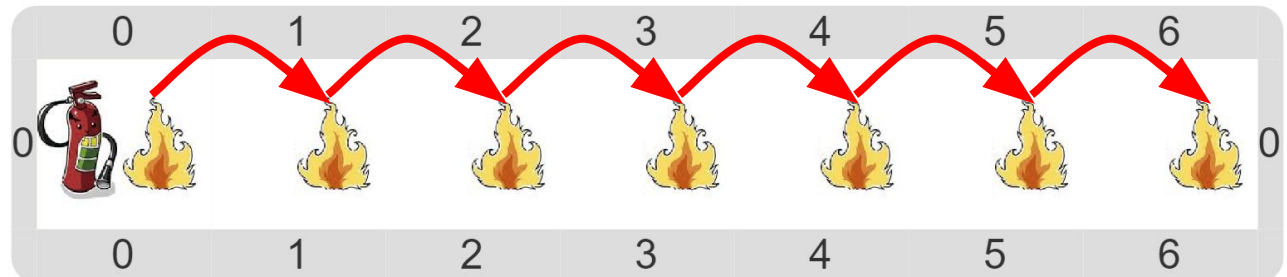


- Al usar una repetición
 - Hay que tener cuidado de los “casos de borde”
 - En los “bordes” a veces hay que hacer algo diferente

```
procedure ApagarIncendio() {  
  /* ... */  
  repeat (6) {  
    UsarMatafuego()  
    IrAlSiguienteFocoAlEste()  
  }  
  UsarMatafuego()  
}
```

Hay 7 focos de incendio,
¡pero solo hay que
moverse 6 veces!

Caso de borde





- Una regla general para mejorar código
 - *No usar una repetición dentro de otra*
 - Si bien se puede hacer, es difícil de entender
 - Recordar que la **legibilidad** importa

```
procedure EntrenarAlBeto() {  
    /* ... */  
    repeat (4) {  
        repeat (6) { MoverAlBetoAl_(Este) }  
        MoverElPieDelBeto()  
        repeat (3) { MoverLaPelotaAl_(Este) }  
        repeat (3) { Mover(Oeste) }  
        repeat (6) { MoverAlBetoAl_(Oeste) }  
        MoverAlBetoAl_(Norte)  
    }  
    // ¡Caso de borde!  
    repeat (6) { MoverAlBetoAl_(Este) }  
    MoverElPieDelBeto()  
    repeat (3) { MoverLaPelotaAl_(Este) }  
    repeat (3) { Mover(Oeste) }  
    repeat (6) { MoverAlBetoAl_(Oeste) }  
}
```

Una repetición
dentro de otra

¡MUY FEO!
¡¡FEÍSIMO!!



- ¡No usar una repetición dentro de otra!
 - Si hay 2 repeticiones, es porque hay una subtarea que se repite. ¡Mejor definir procedimientos!

```
procedure EntrenarAlBeto() {  
    /* ... */  
    repeat (4) {  
        PatearYVolver()  
        IrALaSiguienteFilaAlNorte()  
    }  
    PatearYVolver()  
}
```

```
procedure PatearYVolver() {  
    /* ... */  
    IrHastaLaPelota()  
    PatearLaPelota()  
    Volver()  
}
```

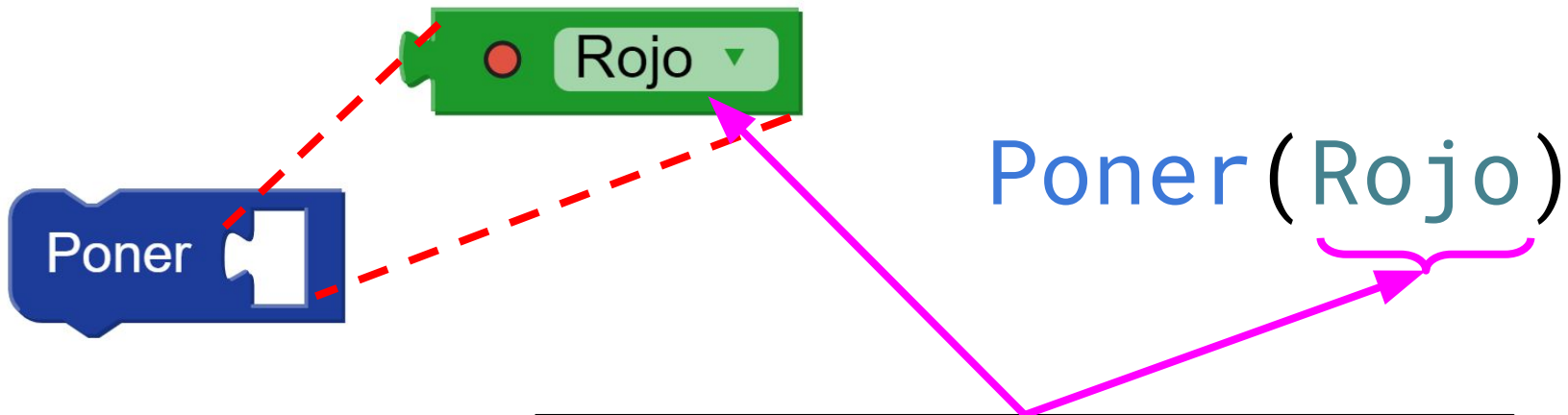
```
procedure IrHastaLaPelota() {  
    /* ... */  
    repeat (6) {  
        MoverAlBetoAl_(Este)  
    }  
}
```

¡Así es más fácil
de entender!



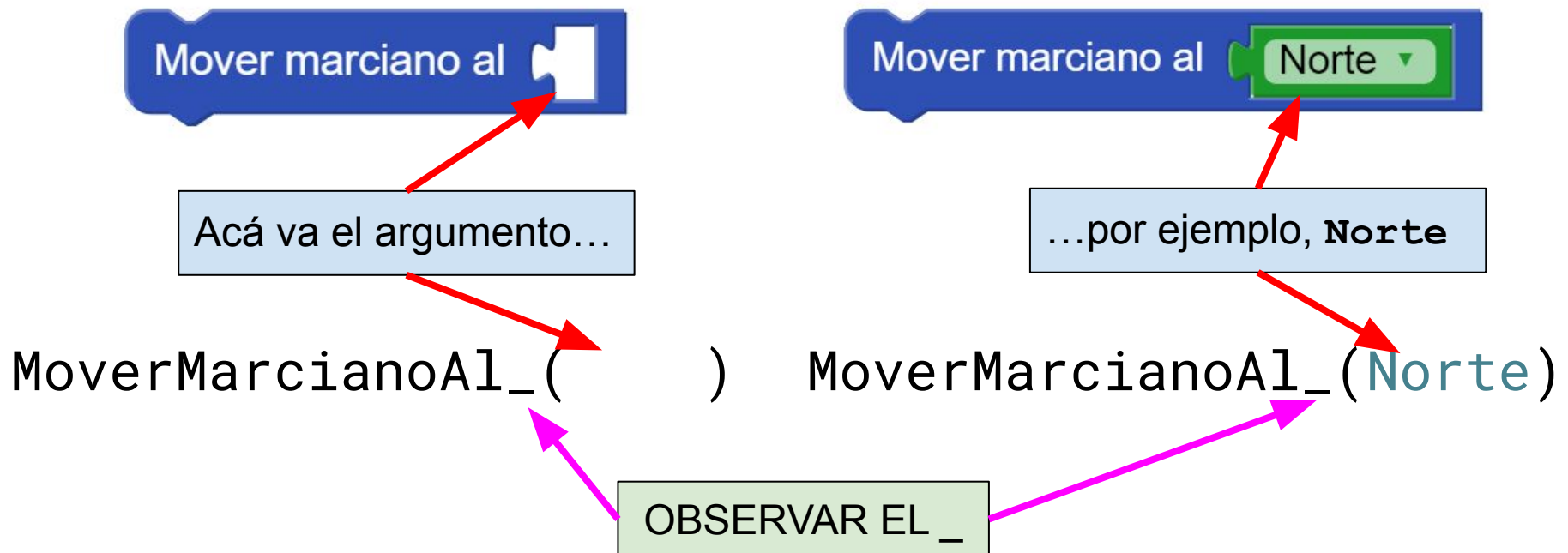
Parámetros

- Vimos que los comandos primitivos llevan **argumentos**
 - Es un dato que le da información al comando
 - En GobstonesJr se arma como un rompecabezas
 - En GobstonesSr se usan paréntesis después del nombre del comando



La expresión **Rojo** es el **argumento** del comando **Poner**

- Los procedimientos primitivos también pueden llevar **argumentos**
 - Sirven para lo mismo, y se escriben igual
 - En GobstonesWeb, los *procedimientos primitivos* que esperan argumentos tienen un `_` en el nombre en su forma textual





- Los procedimientos definidos por nosotros, por el momento, no pueden llevar argumentos
 - Los llamamos ***procedimientos simples***
 - Van seguidos de (), para indicar que no los tienen
 - ¿Cómo hacer para que esperen argumentos?

Buscar todo el hierro

Definir Buscar todo el hierro



Buscar un hierro

No espera argumentos

BuscarTodoElHierro()

```
procedure BuscarTodoElHierro() {  
  /* ... */  
  BuscarUnHierro()  
}
```




- Un procedimiento puede tener **parámetros** en su definición
 - Hablamos de **procedimientos con parámetros**
 - Por cada parámetro definido, el comando definido esperará un argumento (misma cantidad y orden)
 - ¿Cómo se definen los parámetros?

Dibujar cuadrado con:
color del cuadrado

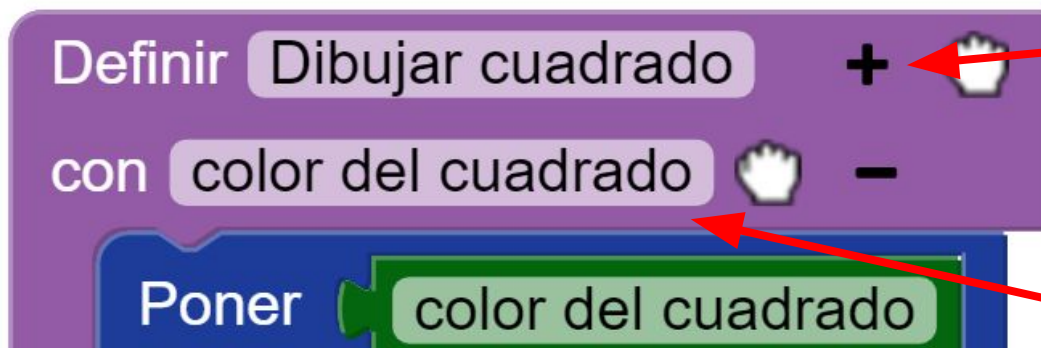
Rojo ▼

¡Quiero definir un
procedimiento que
tenga esta posibilidad!

DibujarCuadrado_(Rojo)



- ¿Cómo se definen los parámetros?
 - En bloques, con el **+** dentro del bloque de definición
 - En texto, con un nombre entre los paréntesis al definir el procedimiento



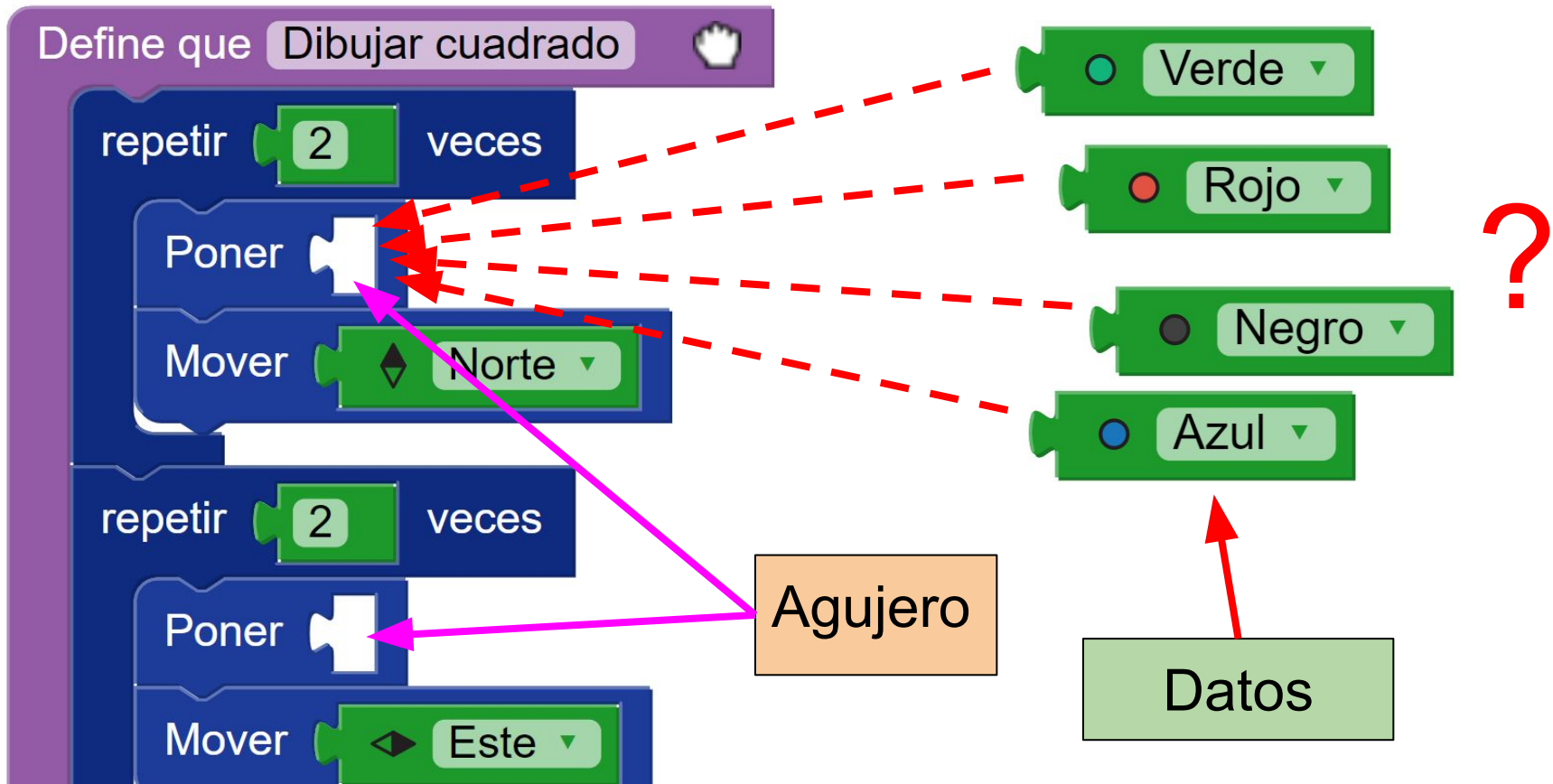
Cada click acá
agrega un parámetro

El nombre debe describir
qué argumento se espera
en ese lugar

```
procedure DibujarCuadrado_(colorDelCuadrado) {  
    /* ... */  
    Poner(colorDelCuadrado)
```



- ¿Qué es un parámetro?
 - Un **agujero** en un procedimiento
 - Un dato que **FALTA**, y debe proveerse al usarlo







- Un procedimiento parametrizado
 - Representa a muchos otros procedimientos simples
 - Permite solucionar muchos problemas parecidos de una sola vez







- ¿Cómo hacer para saber poner un parámetro? (1)
 - En varios procedimientos parecidos, determinar las diferencias (por ejemplo, recuadrarlas)...

```
procedure DibujarCuadradoRojo() {  
  /*  */  
  repeat (2) { Poner(Rojo) Mover(Norte) }  
  repeat (2) { Poner(Rojo) Mover(Este) }  
  repeat (2) { Poner(Rojo) Mover(Sur) }  
  repeat (2) { Poner(Rojo) Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoAzul() {  
  /*  */  
  repeat (2) { Poner(Azul) Mover(Norte) }  
  repeat (2) { Poner(Azul) Mover(Este) }  
  repeat (2) { Poner(Azul) Mover(Sur) }  
  repeat (2) { Poner(Azul) Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoVerde() {  
  /*  */  
  repeat (2) { Poner(Verde) Mover(Norte) }  
  repeat (2) { Poner(Verde) Mover(Este) }  
  repeat (2) { Poner(Verde) Mover(Sur) }  
  repeat (2) { Poner(Verde) Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoNegro() {  
  /*  */  
  repeat (2) { Poner(Negro) Mover(Norte) }  
  repeat (2) { Poner(Negro) Mover(Este) }  
  repeat (2) { Poner(Negro) Mover(Sur) }  
  repeat (2) { Poner(Negro) Mover(Oeste) }  
}
```




- ¿Cómo hacer para saber poner un parámetro? (1)
 - En varios procedimientos parecidos, determinar las diferencias (por ejemplo, recuadrarlas)...

```
procedure DibujarCuadradoRojo () {  
    /* ... */  
    repeat (2) { Poner(Rojo)  Mover(Norte) }  
    repeat (2) { Poner(Rojo)  Mover(Este)  }  
    repeat (2) { Poner(Rojo)  Mover(Sur)   }  
    repeat (2) { Poner(Rojo)  Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoAzul () {  
    /* ... */  
    repeat (2) { Poner(Azul)   Mover(Norte) }  
    repeat (2) { Poner(Azul)   Mover(Este)  }  
    repeat (2) { Poner(Azul)   Mover(Sur)   }  
    repeat (2) { Poner(Azul)   Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoVerde() {  
    /* ... */  
    repeat (2) { Poner(Verde) Mover(Norte) }  
    repeat (2) { Poner(Verde) Mover(Este)  }  
    repeat (2) { Poner(Verde) Mover(Sur)   }  
    repeat (2) { Poner(Verde) Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoNegro() {  
    /* ... */  
    repeat (2) { Poner(Negro) Mover(Norte) }  
    repeat (2) { Poner(Negro) Mover(Este)  }  
    repeat (2) { Poner(Negro) Mover(Sur)   }  
    repeat (2) { Poner(Negro) Mover(Oeste) }  
}
```



- ¿Cómo hacer para saber poner un parámetro? (1)
 - En varios procedimientos parecidos, determinar las diferencias (por ejemplo, recuadrarlas)...

```
procedure DibujarCuadradoRojo () {  
  /* ... */  
  repeat (2) { Poner(Rojo) Mover(Norte) }  
  repeat (2) { Poner(Rojo) Mover(Este) }  
  repeat (2) { Poner(Rojo) Mover(Sur) }  
  repeat (2) { Poner(Rojo) Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoAzul () {  
  /* ... */  
  repeat (2) { Poner(Azul) Mover(Norte) }  
  repeat (2) { Poner(Azul) Mover(Este) }  
  repeat (2) { Poner(Azul) Mover(Sur) }  
  repeat (2) { Poner(Azul) Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoVerde() {  
  /* ... */  
  repeat (2) { Poner(Verde) Mover(Norte) }  
  repeat (2) { Poner(Verde) Mover(Este) }  
  repeat (2) { Poner(Verde) Mover(Sur) }  
  repeat (2) { Poner(Verde) Mover(Oeste) }  
}
```

```
procedure DibujarCuadradoNegro() {  
  /* ... */  
  repeat (2) { Poner(Negro) Mover(Norte) }  
  repeat (2) { Poner(Negro) Mover(Este) }  
  repeat (2) { Poner(Negro) Mover(Sur) }  
  repeat (2) { Poner(Negro) Mover(Oeste) }  
}
```



- ¿Cómo hacer para saber poner un parámetro? (1)
 - En varios procedimientos parecidos, determinar las diferencias (por ejemplo, recuadrarlas)...

<pre>procedure DibujarCuadradoRojo () { /* ... */ repeat (2) { Poner(Rojo) Mover(Norte) } repeat (2) { Poner(Rojo) Mover(Este) } repeat (2) { Poner(Rojo) Mover(Sur) } repeat (2) { Poner(Rojo) Mover(Oeste) } }</pre>	<pre>procedure DibujarCuadradoAzul () { /* ... */ repeat (2) { Poner(Azul) Mover(Norte) } repeat (2) { Poner(Azul) Mover(Este) } repeat (2) { Poner(Azul) Mover(Sur) } repeat (2) { Poner(Azul) Mover(Oeste) } }</pre>
<pre>procedure DibujarCuadradoVerde () { /* ... */ repeat (2) { Poner(Verde) Mover(Norte) } repeat (2) { Poner(Verde) Mover(Este) } repeat (2) { Poner(Verde) Mover(Sur) } repeat (2) { Poner(Verde) Mover(Oeste) } }</pre>	<pre>procedure DibujarCuadradoNegro () { /* ... */ repeat (2) { Poner(Negro) Mover(Norte) } repeat (2) { Poner(Negro) Mover(Este) } repeat (2) { Poner(Negro) Mover(Sur) } repeat (2) { Poner(Negro) Mover(Oeste) } }</pre>



- ¿Cómo hacer para saber poner un parámetro? (1)
 - En varios procedimientos parecidos, determinar las diferencias (por ejemplo, recuadrarlas)...

<pre>procedure DibujarCuadradoRojo() /* ... */ repeat (2) { Poner(Rojo) Mover(Norte) } repeat (2) { Poner(Rojo) Mover(Este) } repeat (2) { Poner(Rojo) Mover(Sur) } repeat (2) { Poner(Rojo) Mover(Oeste) } }</pre>	<pre>procedure DibujarCuadradoAzul () { /* ... */ repeat (2) { Poner(Azul) Mover(Norte) } repeat (2) { Poner(Azul) Mover(Este) } repeat (2) { Poner(Azul) Mover(Sur) } repeat (2) { Poner(Azul) Mover(Oeste) } }</pre>
---	--



- ¿Cómo hacer para saber poner un parámetro? (1)
 - En varios procedimientos parecidos, determinar las diferencias (por ejemplo, recuadrarlas)...

```
procedure DibujarCuadradoNegro() {  
  /* ... */  
  repeat (2) { Poner(Negro) Mover(Norte) }  
  repeat (2) { Poner(Negro) Mover(Este) }  
  repeat (2) { Poner(Negro) Mover(Sur) }  
  repeat (2) { Poner(Negro) Mover(Oeste) }  
}
```








- ¿Cómo hacer para saber poner un parámetro? (1)
 - En varios procedimientos parecidos, determinar las diferencias (por ejemplo, recuadrarlas)...

```
procedure DibujarCuadradoNorte() {  
    /* ... */  
    repeat (2) { Poner(Norte) Mover(Norte) }  
    repeat (2) { Poner(Norte) Mover(Este) }  
    repeat (2) { Poner(Norte) Mover(Sur) }  
    repeat (2) { Poner(Norte) Mover(Oeste) }  
}
```



- ¿Cómo hacer para saber poner un parámetro? (2)
 - ...y recortar el contenido de los recuadros para obtener el agujero. ¡Quedan todos iguales!

```
procedure DibujarCuadrado  () {  
    /* ... */  
    repeat (2) { Poner  Mover(Norte) }  
    repeat (2) { Poner  Mover(Este) }  
    repeat (2) { Poner  Mover(Sur) }  
    repeat (2) { Poner  Mover(Oeste) }  
}
```



- ¿Cómo hacer para saber poner un parámetro? (2)
 - ...y recortar el contenido de los recuadros para obtener el agujero. ¡Quedan todos iguales!

```
procedure DibujarCuadrado i() { Azul
/* ... */
repeat (2) { Poner( ) Mover(Norte) }
repeat (2) { Poner( ) Mover(Este) }
repeat (2) { Poner( ) Mover(Sur) }
repeat (2) { Poner( ) Mover(Oeste) }
}
```

```
procedure DibujarCuadrado i() { Verde
{ Poner( ) Mover(Norte) }
{ Poner( ) Mover(Este) }
{ Poner( ) Mover(Sur) }
{ Poner( ) Mover(Oeste) }
```

```
procedure DibujarCuadrado i() { Negro
/* ... */
repeat (2) { Poner( ) Mover(Norte) }
repeat (2) { Poner( ) Mover(Este) }
repeat (2) { Poner( ) Mover(Sur) }
repeat (2) { Poner( ) Mover(Oeste) }
}
```

```
procedure DibujarCuadrado i() { Rojo
{ Poner( ) Mover(Norte) }
{ Poner( ) Mover(Este) }
{ Poner( ) Mover(Sur) }
{ Poner( ) Mover(Oeste) }
```



- ¿Cómo hacer para saber poner un parámetro? (3)
 - El procedimiento con agujero es uno solo
 - Pero falta *algo* para que esté completo

Azul

Verde

Negro

Rojos

```
procedure DibujarCuadrado() {  
    /* ... */  
    repeat (2) { Poner( ) Mover(Norte) }  
    repeat (2) { Poner( ) Mover(Este) }  
    repeat (2) { Poner( ) Mover(Sur) }  
    repeat (2) { Poner( ) Mover(Oeste) }  
}
```



- ¿Cómo hacer para saber poner un parámetro? (4)
 - Le ponemos **nombre** al agujero...
 - ...y ahora el procedimiento está completo

```
procedure DibujarCuadrado_(colorDelCuadrado) {  
    /* ... */  
    repeat (2) { Poner(colorDelCuadrado) Mover(Norte) }  
    repeat (2) { Poner(colorDelCuadrado) Mover(Este) }  
    repeat (2) { Poner(colorDelCuadrado) Mover(Sur) }  
    repeat (2) { Poner(colorDelCuadrado) Mover(Oeste) }  
}
```

DibujarCuadrado_(Rojo)

DibujarCuadrado_(Azul)

DibujarCuadrado_(Negro)

DibujarCuadrado_(Verde)



- El parámetro tiene un ***nombre***
 - Que representa al valor del argumento
- El parámetro *solamente puede usarse* en el procedimiento que lo define (y en **ningún** otro lado)

Nombre del parámetro

```
procedure DibujarCuadrado_(colorDelCuadrado) {  
  /* ... */  
  repeat (2) { Poner(colorDelCuadrado) Mover(Norte) }  
  repeat (2) { Poner(colorDelCuadrado) Mover(Este) }  
  repeat (2) { Poner(colorDelCuadrado) Mover(Sur) }  
  repeat (2) { Poner(colorDelCuadrado) Mover(Oeste) }  
}
```

Usos del parámetro



- El nombre de un parámetro
 - Debe ser un sustantivo (pues describe un dato)
 - En Gobstones, debe empezar con minúscula
 - También usamos camelCase para escribirlo

color 2 p1 colorDelCuadrado

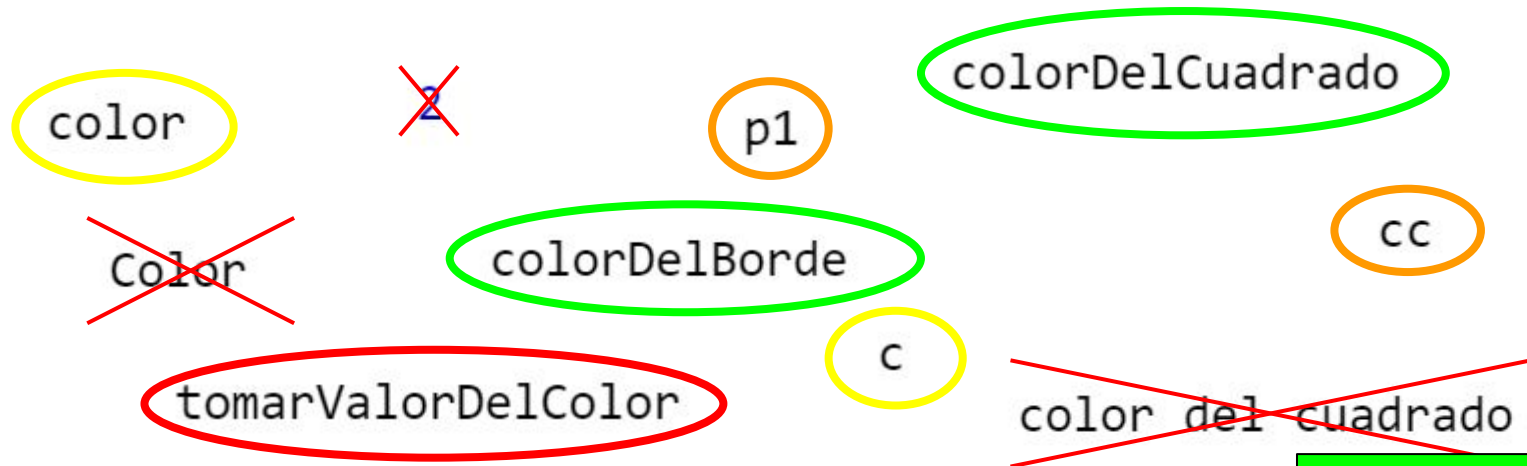
Color colorDelBorde cc

tomarValorDelColor c color del cuadrado

¿Cuales de éstos son nombres válidos para el parámetro anterior? ¿Y cuáles son adecuados? ¿Por qué?



- El nombre de un parámetro
 - Debe ser un sustantivo (pues describe un dato)
 - En Gobstones, debe empezar con minúscula
 - También usamos camelCase para escribirlo



**¡Este sí en
GobstonesJr!**

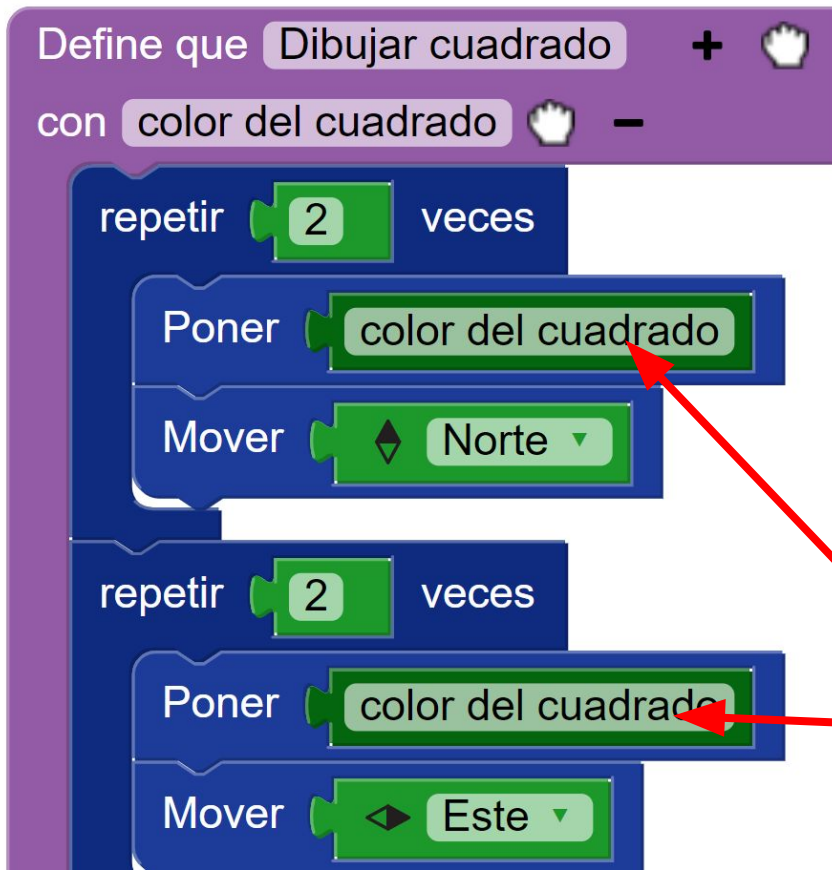
¿Cuales de éstos son nombres válidos para el parámetro anterior? ¿Y cuáles son adecuados? ¿Por qué?



Los parámetros representan valores



- Como un parámetro representa a un valor
 - puede usarse como argumento en otros comandos
 - puede combinarse con otros valores en expresiones



Representa al valor del argumento
(e.g. Rojo, Azul, etc.)
¡Por eso su forma!



Usos del
parámetro



- ¿Qué pasa si el parámetro de un procedimiento es un número?
 - ¡Se puede usar en repeticiones!
 - Permite repetir cantidades distintas cada vez



Dibujar línea Roja al Norte con:
longitud de la línea al Norte

4

Dibujar línea Roja al Norte con:
longitud de la línea al Norte

2

Representa
a un número



- Los parámetros deben aparecer en el contrato
 - ¿Cuál es su propósito? ¿Qué datos pueden llenar ese agujero? O sea,
 - ¿Qué información va a describir cada parámetro?

```
procedure DibujarCuadrado_DeLado_(colorDelCuadrado, longitudDelLado) {  
  /* PROPÓSITO: Dibujar un cuadrado de color **colorDelCuadrado**  
    de lado **longitudDelLado**.  
  PARÁMETROS:  
    * colorDelCuadrado: Un color que describe el color del  
      cuadrado a dibujar.  
    * longitudDelLado: Un número que describe el largo de  
      cada lado del cuadrado a dibujar.  
  PRECONDICIONES:  
    * Hay al menos **longitudDelLado** celdas al Norte y  
      al Este de la celda actual.  
  */
```



- Restricciones en el uso de parámetros (1)
 - Solamente tienen validez dentro del procedimiento que los define (hablamos del ***alcance*** del parámetro)




¡No sirve afuera de DibujarCuadrado!




Alcance de color del cuadrado

- Restricciones en el uso de parámetros (2)
 - En bloques se valida el alcance al armar
 - En texto, no; da error al ejecutar

```
procedure DibujarCuadrado_(colorDelCuadrado) {  
  /**/  
  DibujarLínea_DeLargo2Al_(colorDelCuadrado, Norte)  
  DibujarLínea_DeLargo2Al_(colorDelCuadrado, Este)  
  DibujarLínea_DeLargo2Al_(colorDelCuadrado, Sur)  
  DibujarLínea_DeLargo2Al_(colorDelCuadrado, Oeste)  
}
```

Alcance de
colorDelCuadrado

```
procedure DibujarLínea_DeLargo2Al_(colorDeLaLínea, direcció  
  /**/  
  repeat (2)  
  { Poner(colorDelCuadrado) Mover(dirección) }  
}
```

¡No sirve afuera de
DibujarCuadrado!



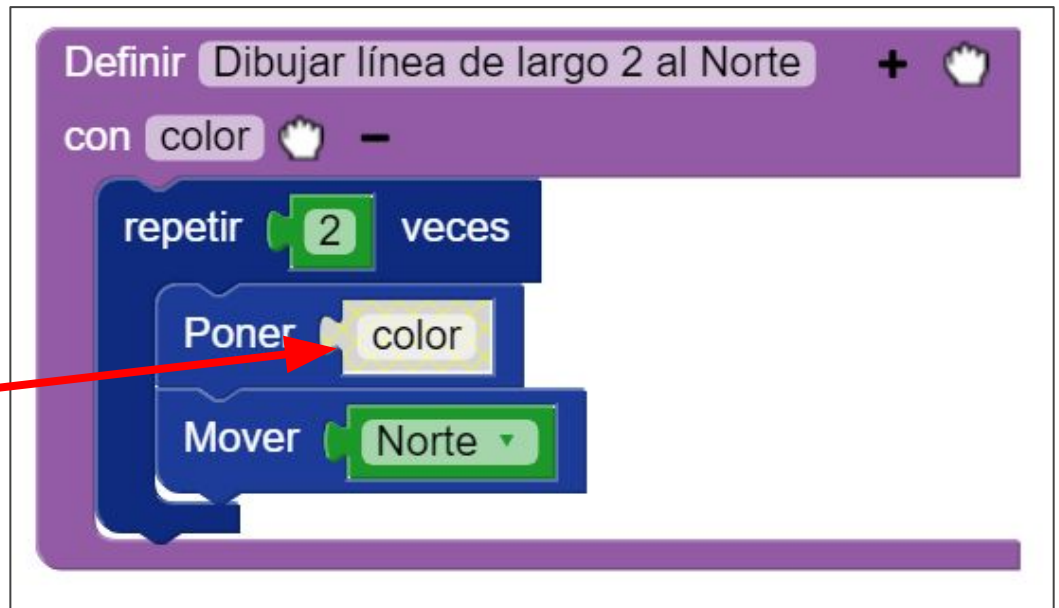
BOOM

La variable "colorDelCuadrado"
no está definida.



- Restricciones en el uso de parámetros (3)
 - No es bueno repetir nombres de parámetros entre diferentes procedimientos porque genera confusión (al menos al principio)

¿Por qué no anda, si el nombre parece estar bien?



- Restricciones en el uso de parámetros (3)
 - No es bueno repetir nombres de parámetros entre diferentes procedimientos porque genera confusión (al menos al principio)

Definir Dibujar cuadrado +

con color -

Dibujar línea de largo 2 al Norte con: color color

Dibujar línea de largo 2 al Este con: color color

Dibujar línea de largo 2 al Sur con: color color

Dibujar línea de largo 2 al Oeste con: color color

Definir Dibujar línea de largo 2 al Norte +

con color -

repetir 2 veces

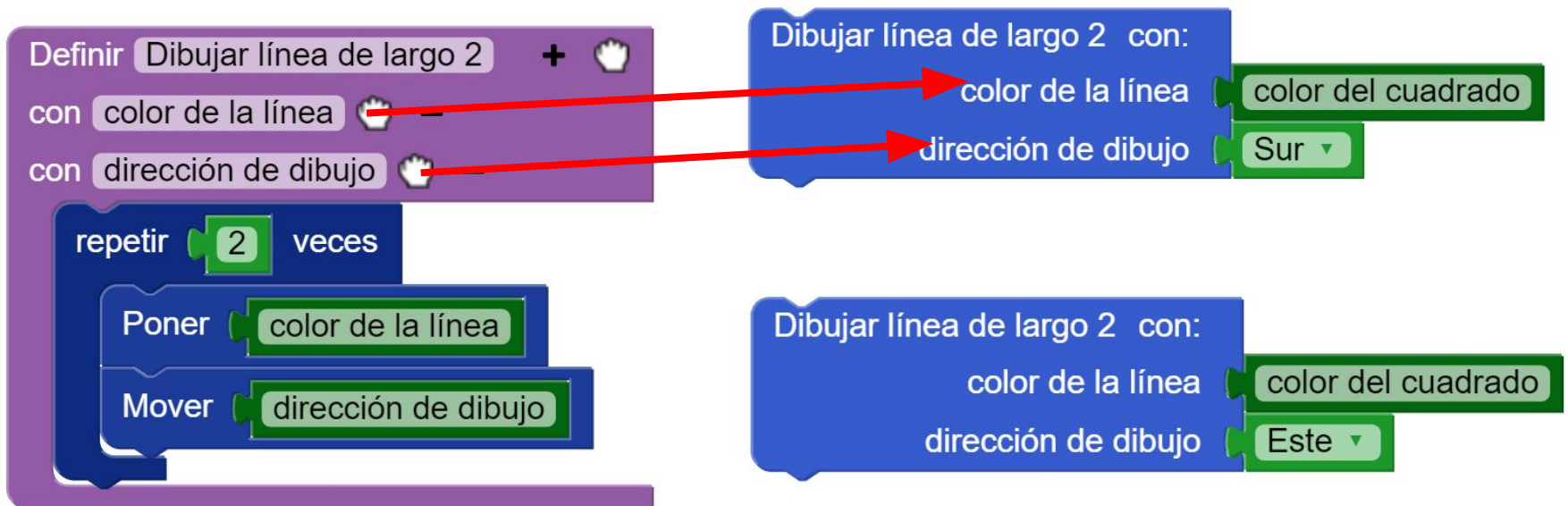
Poner color color

Mover Norte


¡Porque es el parámetro de otro procedimiento!



- Puede haber más de un parámetro (1)
 - En ese caso, se debe respetar la cantidad y el orden
 - En bloques, es fácil, por las formas



- Puede haber más de un parámetro (2)
 - En texto, se separan con comas y hay que recordar el orden y la cantidad cada vez que se usa

```
procedure DibujarLínea_DeLargo2Al_(colorDeLaLínea, direcciónDeDibujo) {  
  /*  */  
  repeat (2)  
    { Poner(colorDeLaLínea) Mover(direcciónDeDibujo) }  
}
```

DibujarLínea_DeLargo2Al_(colorDelCuadrado, Este)

DibujarLínea_DeLargo2Al_(colorDelCuadrado, Sur)




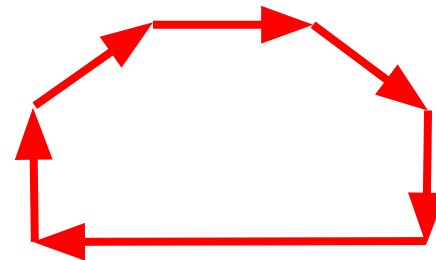
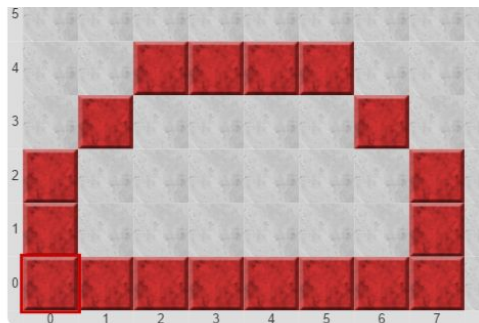
- Usando parámetros puedo hacer subtarefas poderosas
 - Líneas de cualquier longitud, dirección y color
 - Incluso líneas en diagonal (¡Ojo a las precondiciones!)

```
procedure DibujarLínea_DeLargo_EnDiagonal__(colorDeLaLínea, largoDeLaLínea
                                           , dirección1, dirección2) {
  /* PROPÓSITO:
    * Dibujar una línea en diagonal del largo y color dado hacia las
      dos direcciones dadas.
    * Ubicar el cabezal **largoDeLaLínea** celdas al **dirección1** y
      al **dirección2**.
  PARÁMETROS:
    * colorDeLaLínea: El color con el que dibujar la línea.
    * largoDeLaLínea: La cantidad de celdas que debe ocupar la línea.
    * dirección1 y dirección2: Las direcciones hacia donde dibujar.
  PRECONDICIONES:
    * **dirección1** y **dirección2** no son iguales ni opuestas.
    * Hay al menos **largoDeLaLínea** celdas al **dirección1** y al
      **dirección2** de la celda actual.
  */
  repeat (largoDeLaLínea) {
    Poner(colorDeLaLínea) Mover(dirección1) Mover(dirección2)
  }
}
```



- Usando parámetros puedo hacer subtarefas poderosas
 - Y dibujar figuras complejas con ellas

```
procedure DibujarFiguraRoja() {  
  /*  */  
  DibujarLínea_DeLargo_Al_(Rojo, 2, Norte)  
  DibujarLínea_DeLargo_EnDiagonal__(Rojo, 2, Norte, Este)  
  DibujarLínea_DeLargo_Al_(Rojo, 3, Este)  
  DibujarLínea_DeLargo_EnDiagonal__(Rojo, 2, Sur, Este)  
  DibujarLínea_DeLargo_Al_(Rojo, 2, Sur)  
  DibujarLínea_DeLargo_Al_(Rojo, 7, Oeste)  
}
```





- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone la representación de una rosa con su tallo y su maceta
 - **PonerAsDeEspadas()**, que pone la representación de la carta 1 de Espadas

Representación	Azul	Negro	Rojo	Verde
Rosa	0	4	5	3
As de Espadas	4	301	0	0

¿Cuántos comandos **Poner** tenemos que escribir?
¿Y cuántas repeticiones simples?



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure PonerRosa() {  
  /* ... */  
  repeat (5) { Poner(Rojo) }  
  repeat (3) { Poner(Verde) }  
  repeat (4) { Poner(Negro) }  
}
```

```
procedure PonerAsDeEspadas() {  
  /* ... */  
  repeat (4) { Poner(Azul) }  
  repeat (301) { Poner(Negro) }  
}
```

Representación	Azul	Negro	Rojo	Verde
Rosa	0	4	5	3
As de Espadas	4	301	0	0

¡Hay mucho
código parecido!
(5 veces repetir+Poner)

¡Definir subtarea!!



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure PonerRosa() {  
  /* ... */  
  repeat (5) { Poner(Rojo) }  
  repeat (3) { Poner(Verde) }  
  repeat (4) { Poner(Negro) }  
}
```

Representación	Azul	Negro	Rojo	Verde
Rosa	0	4	5	3
As de Espadas	4	301	0	0

```
procedure PonerAsDeEspadas() {  
  /* ... */  
  repeat (4) { Poner(Azul) }  
  repeat (301) { Poner(Negro) }  
}
```

¡Hay mucho
código parecido!
(5 veces repetir+Poner)

¡Definir subtarea!!



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure PonerRosa() {  
  /* ... */  
  repeat ([ ]) { Poner([ ]) }  
  repeat ([ ]) { Poner([ ]) }  
  repeat ([ ]) { Poner([ ]) }  
}
```

Representación	Azul	Negro	Rojo	Verde
Rosa	0	4	5	3
As de Espadas	4	301	0	0

```
procedure PonerAsDeEspadas() {  
  /* ... */  
  repeat ([ ]) { Poner([ ]) }  
  repeat ([ ]) { Poner([ ]) }  
}
```

¡Hay mucho
código parecido!
(5 veces repetir+Poner)

¡Definir subtarea!!



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure Poner_DeColor_(                    ,                    ) {  
    /* ... */  
    repeat (                    ) {  
        Poner(                    )  
    }  
}
```

La subtarea es para poner muchas bolitas juntas

¡Falta determinar el
nombre de los
parámetros!



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure Poner_DeColor_(5, Rojo) {  
    /* ... */  
    repeat (5) {  
        Poner(Rojo)  
    }  
}
```



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure Poner_DeColor_(  
    /* ... */  
    repeat (  
        Poner(  
    )  
}  
) {  
    }  
}
```

Diagram illustrating the structure of the `Poner_DeColor_` procedure with annotations:

- `cantidadAPoner` (purple) points to the `5` in the `repeat` loop.
- `colorAPoner` (green) points to the `Rojo` in the `Poner` call.
- `cantidadAPoner` (purple) points to the `5` in the `Poner_DeColor_` call.
- `colorAPoner` (green) points to the `Rojo` in the `Poner_DeColor_` call.



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure Poner_DeColor_(cantidadAPoner,colorAPoner) {  
    /* ... */  
    repeat (cantidadAPoner) {  
        Poner(colorAPoner)  
    }  
}
```

Buenos nombres
para los parámetros



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure Poner_DeColor_(cantidadAPoner,colorAPoner) {  
    /* ... */  
    repeat (cantidadAPoner) {  
        Poner(colorAPoner)  
    }  
}
```

Buenos nombres
para los parámetros



- ¿Cómo construir los siguientes procedimientos?
 - **PonerRosa()**, que pone una rosa
 - **PonerAsDeEspadas()**, que pone la carta 1 de Espadas

```
procedure PonerRosa() {  
  /* ... */  
  Poner_DeColor_(5, Rojo)  
  Poner_DeColor_(3, Verde)  
  Poner_DeColor_(4, Negro)  
}
```

Representación	Azul	Negro	Rojo	Verde
Rosa	0	4	5	3
As de Espadas	4	301	0	0

```
procedure PonerAsDeEspadas() {  
  /* ... */  
  Poner_DeColor_(4, Azul)  
  Poner_DeColor_(301, Negro)  
}
```

Ahora con la
subtarea nueva

- Escribir un procedimiento `PonerDominó__(númeroIzquierdo, númeroDerecho)` que ponga un dominó horizontal en la celda actual
- ¡No olvidar escribir su contrato (propósito, parámetros y **precondiciones**) y aplicar *todos* los conceptos vistos!

Representación

- Una bolita azul indica que hay un dominó horizontal
- Las bolitas rojas indican el número de la izquierda
- Las bolitas verdes indican el número de la derecha

`PonerDominó__(2,3)`



`PonerDominó__(6,6)`





Cierre



Cierre

- **Repetición simple**

- una herramienta del lenguaje para repetir acciones
- se arma con una expresión numérica y un cuerpo
- la cantidad de repeticiones es fija
- arma un comando, por lo que se puede usar junto con otros comandos en procedimientos
- deben tenerse en cuenta condiciones “de borde”
- es mejor usar una única repetición por procedimiento



- **Parámetros**

- una herramienta del lenguaje para hacer procedimientos más generales
- se define junto con un procedimiento y representa un *agujero* en el mismo que debe completarse
- se completa con un **argumento** al momento de usar el procedimiento como comando
- tiene un **nombre** que debe seguir reglas
 - empezar con minúscula
 - empezar con un sustantivo (porque describe a un dato)
 - describir para qué se va a usar

● Parámetros

- un parámetro solamente sirve en el procedimiento que lo define (**alcance** = el cuerpo de ese procedimiento)
- puede haber varios parámetros en un mismo procedimiento (se separan con comas)
- la cantidad y el orden importa al usar el procedimiento como comando
- permiten definir procedimientos muy poderosos
 - proveen generalidad
 - proveen abstracción