Introducción a la Programación

Clases teóricas

por Pablo E. “Fidel” Martínez López

4. Expresiones y tipos

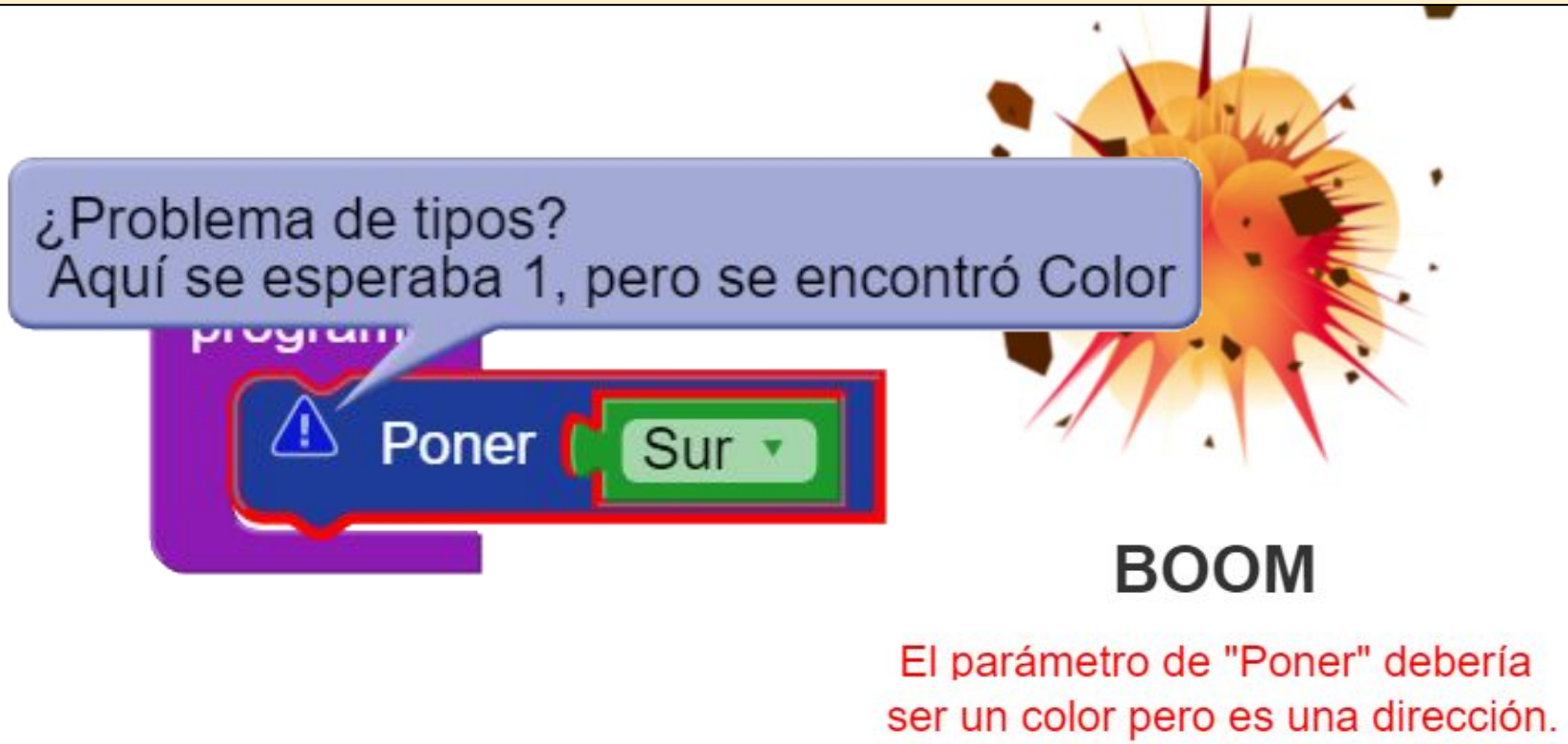
Repaso

- **Programar** es **comunicar** (con máquinas y personas)
- **Programas** (texto con diversos elementos)
  - **Comandos**: describen acciones
  - **Expresiones**: describen información
  - **Propósito, parámetros y precondiciones**
- **Procedimientos**
  - Para definir nuevos comandos
  - Permiten expresar **estrategia** y **representación de información**
  - Aportan legibilidad, reutilización, modificabilidad

- **Repetición simple**
  - herramienta para evitar la repetición de código
  - repite un número fijo de veces
  - deben considerarse condiciones de borde
- **Parámetros**
  - son agujeros en un procedimiento
  - por cada uno hay que poner un argumento
  - llevan un nombre y tienen un alcance
  - proveen generalidad y abstracción

Tipos de datos

- Los comandos primitivos no aceptan cualquier argumento
  - **`Poner`** espera un color, **`Mover`** una dirección
- ¿Qué pasa si les damos otra cosa?



- ¿Cómo saber si un comando va a hacer esto?
- Todas las expresiones tienen la misma forma
  - ¡pero no sirven para lo mismo!
- ¡Hace falta clasificar las expresiones!

Tipos de datos

- ¿Cuál criterio de clasificación utilizar?
  - Según el uso que le podemos dar
  - Eso da lugar a los ***tipos de datos***

Tipos de datos

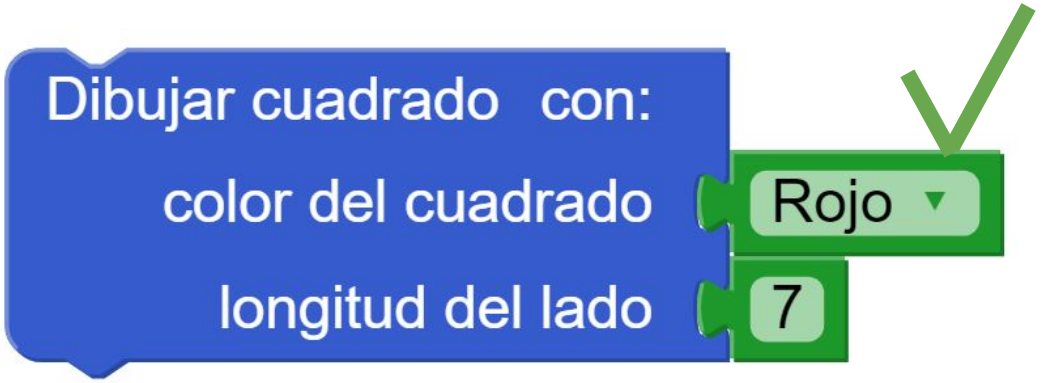
- Hasta ahora hemos encontrado 3 tipos de datos
  - Colores
  - Direcciones
  - Números

- El tipo sirve para saber qué dato poner en un argumento
- Entonces, los parámetros deben indicar cuál es ese tipo
  - Es parte del contrato de los procedimientos, en la parte de los parámetros

```
procedure DibujarCuadrado(colorDelCuadrado, longitudDelLado) {
  /* PROPÓSITO: dibuja un cuadrado con bolitas usando el color
                dado por colorDelCuadrado. El tamaño del cuadrado
                está dado por longitudDelLado
     PARÁMETROS:
       * colorDelCuadrado:
           un color que indica el color de las bolitas con las
           que dibujar el cuadrado
       * longitudDelLado:
           un número que indica la cantidad de celdas que tiene
           que tener de lado el cuadrado resultante
     PRECONDICIONES:
       * hay la cantidad indicada por longitudDelLado de celdas
         al Norte y al Este de la celda actual
  */
```

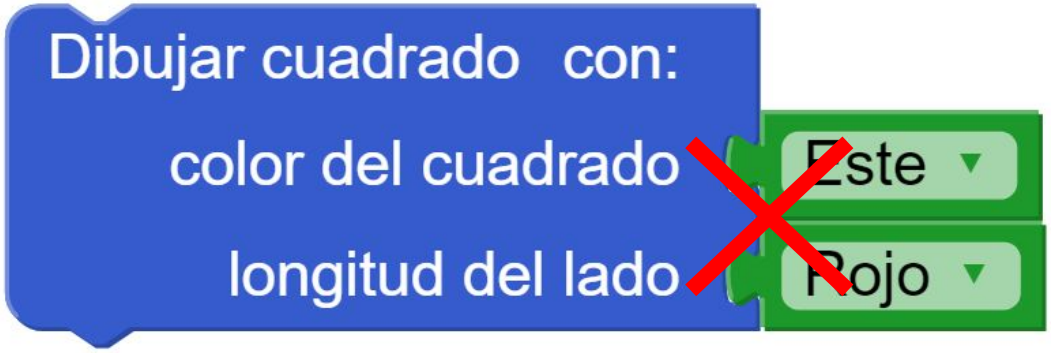
Tipos de datos

- Los parámetros son datos del tipo indicado
- ¡Debe respetarse el tipo al poner el argumento!



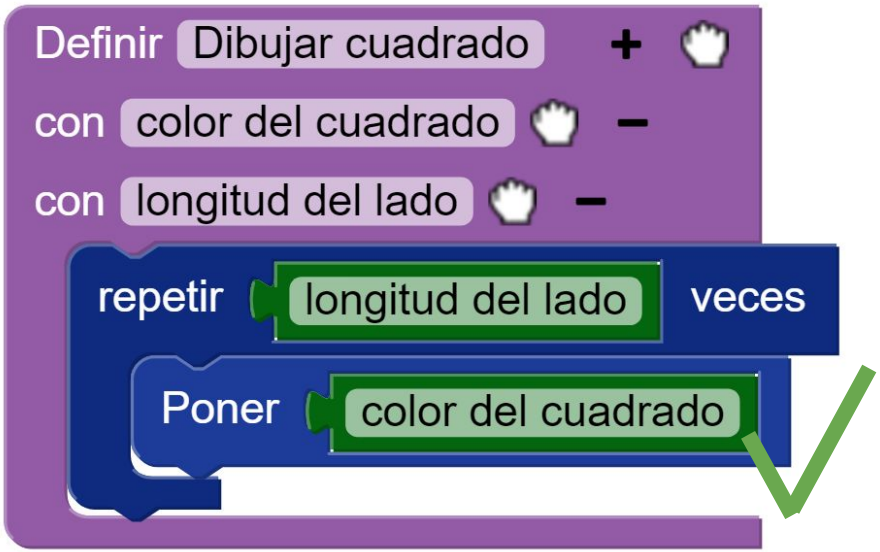
Ok, concuerdan

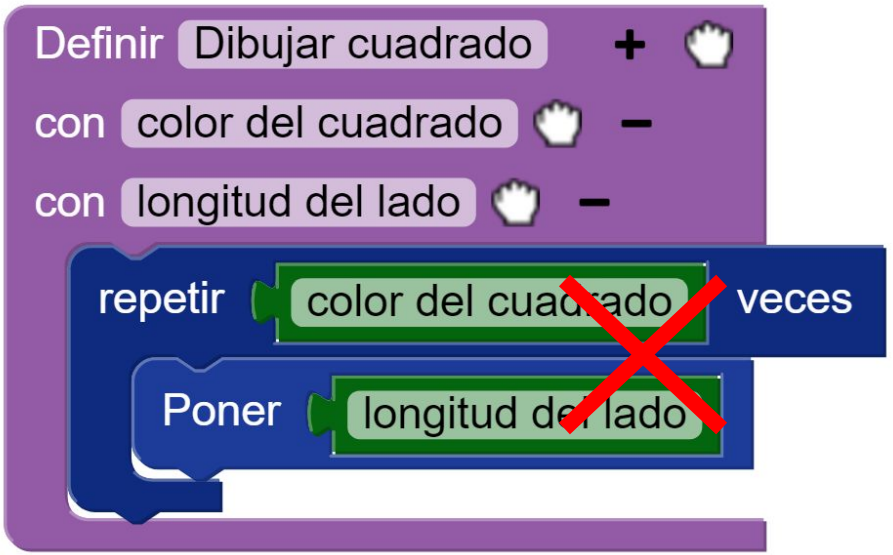
Mal, no concuerdan



- Los parámetros son datos del tipo indicado
- Y también debe respetarse donde se utiliza

Ok, concuerdan

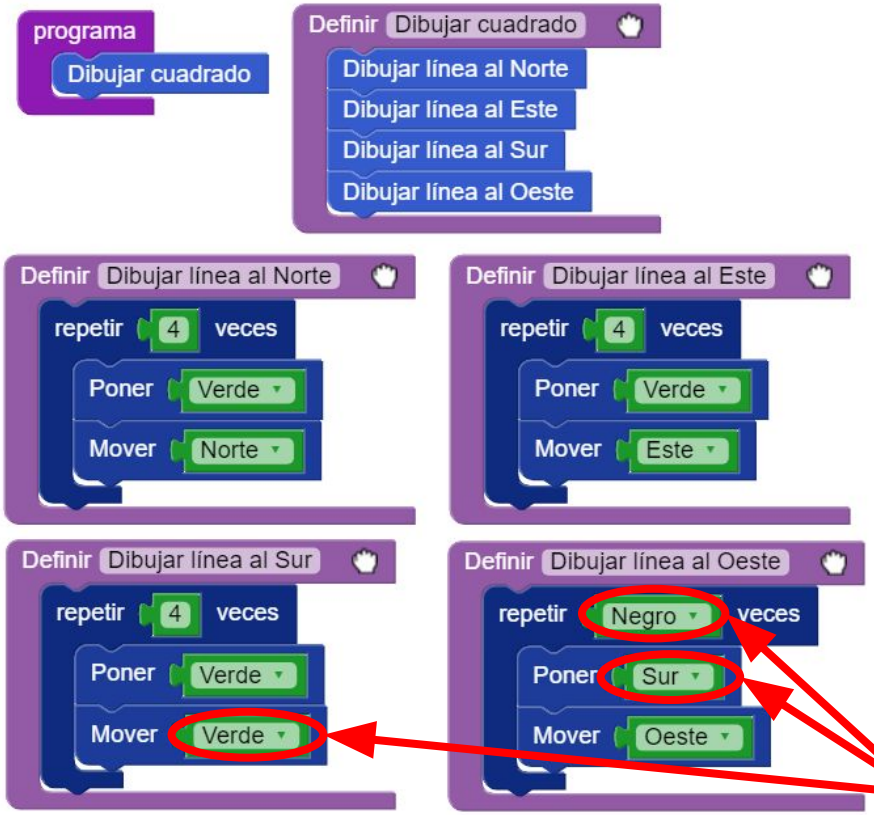


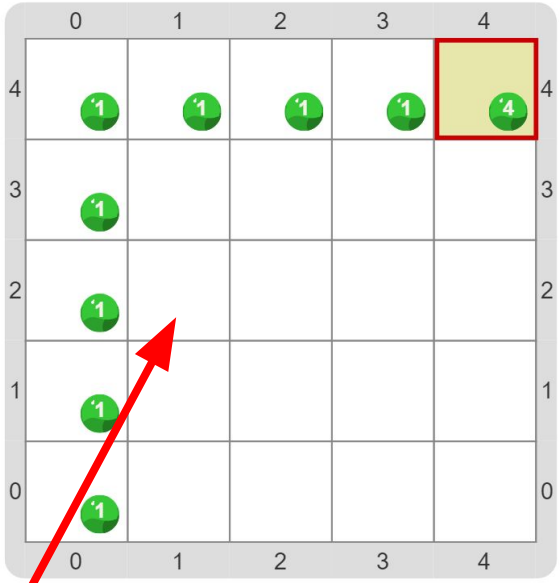


Mal, no concuerdan

Tipos de datos

- ¿Podría ser que los comandos primitivos no fallasen cuando reciben un argumento de tipo incorrecto?
- ¿Ayudaría o sería más complicado?

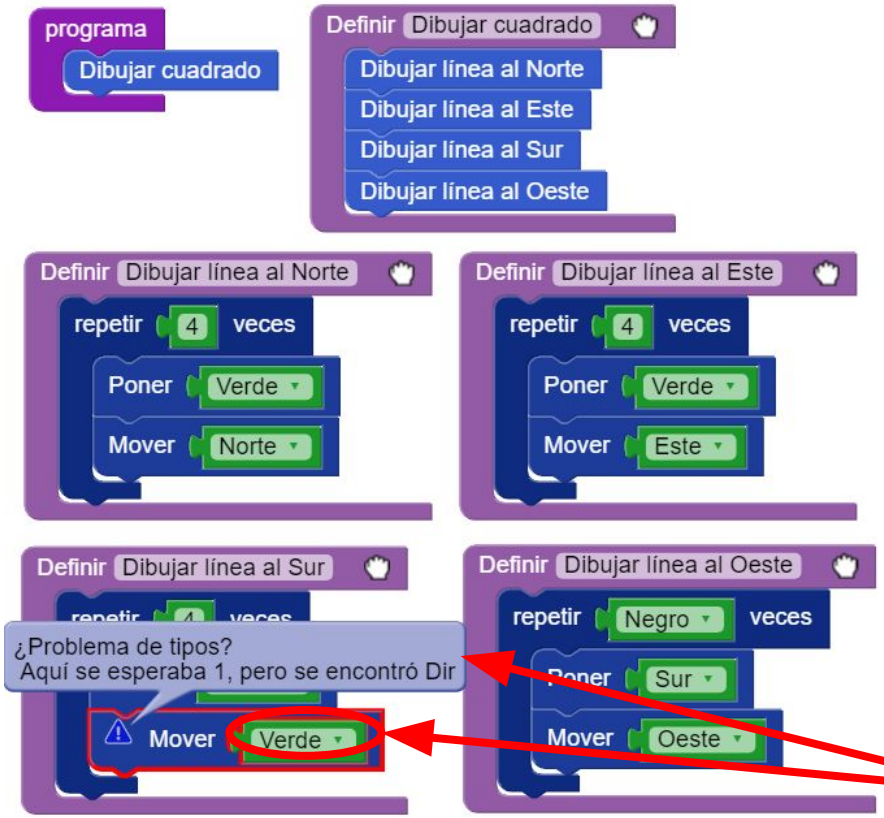




Este sería el resultado si las operaciones incorrectas no fallasen

Tipos de datos

- ¿Podría ser que los comandos primitivos no fallasen cuando reciben un argumento de tipo incorrecto?
- ¿Ayudaría o sería más complicado?



BOOM

El parámetro de "Mover" debería ser una dirección pero es un color.

¡Si falla, avisa en qué lugar!

Expresiones primitivas y operadores

Expresiones primitivas

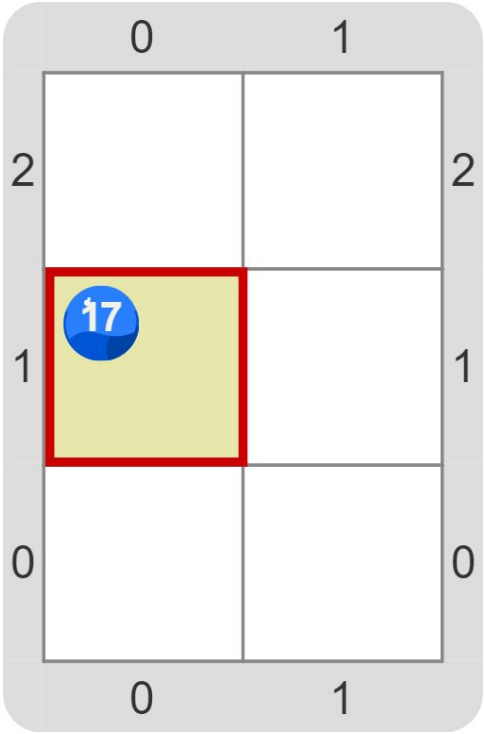
- ¿Cómo saber cuántas bolitas de cierto color hay en la celda actual? Se necesita una herramienta del lenguaje
- Las ***expresiones primitivas*** sirven para sensar el tablero
  - ejemplo: **`nroBolitas(`**<*color*>**`)`**


```
nroBolitas(Rojo)
```

- La expresión **`nroBolitas(`**<*color*>**`)`** de tipo número, describe el número de bolitas de ese color en la celda actual

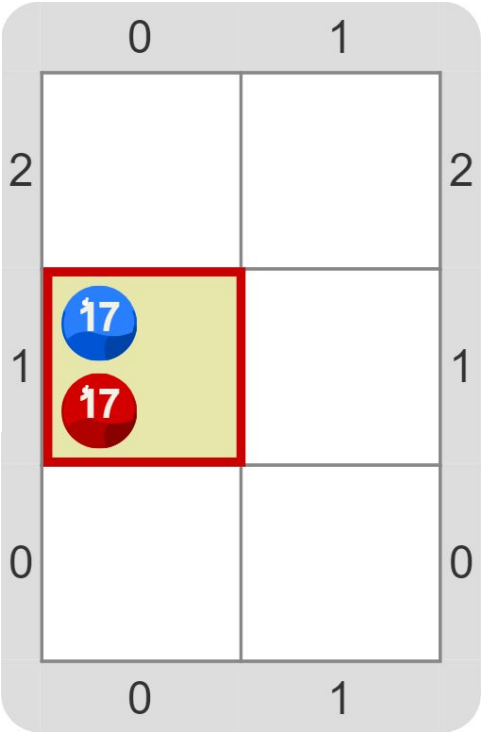
- Dado que **`nroBolitas(`**<*color*>**`)`** describe un número puede usarse en cualquier lugar en el que es necesario un número

¿Cuántas bolitas rojas se van a poner si se ejecuta en este tablero?



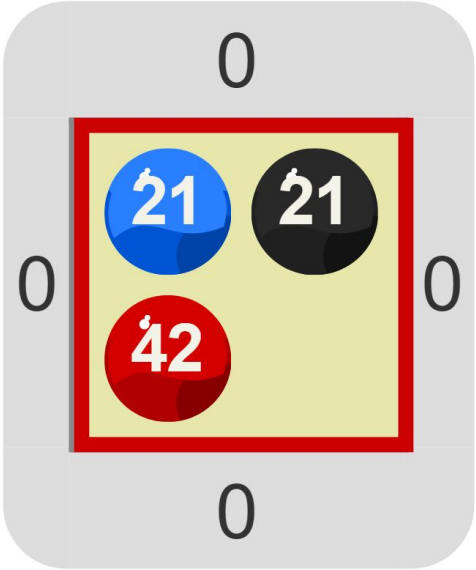
- Dado que **`nroBolitas(`**<*color*>**`)`** describe un número puede usarse en cualquier lugar en el que es necesario un número

¡La misma cantidad que haya de bolitas azules!

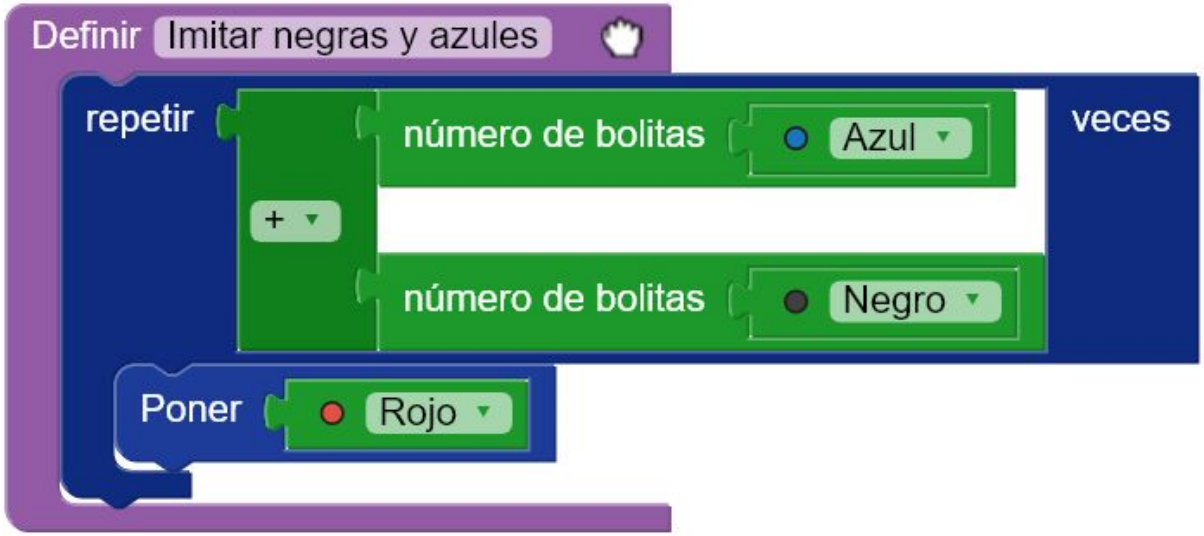


Operadores numéricos

- ¿Y si necesitamos sumar varios números de bolitas?
- Los ***operadores numéricos*** sirven para hacer cuentas con cualquier número (literal o tomado del tablero)

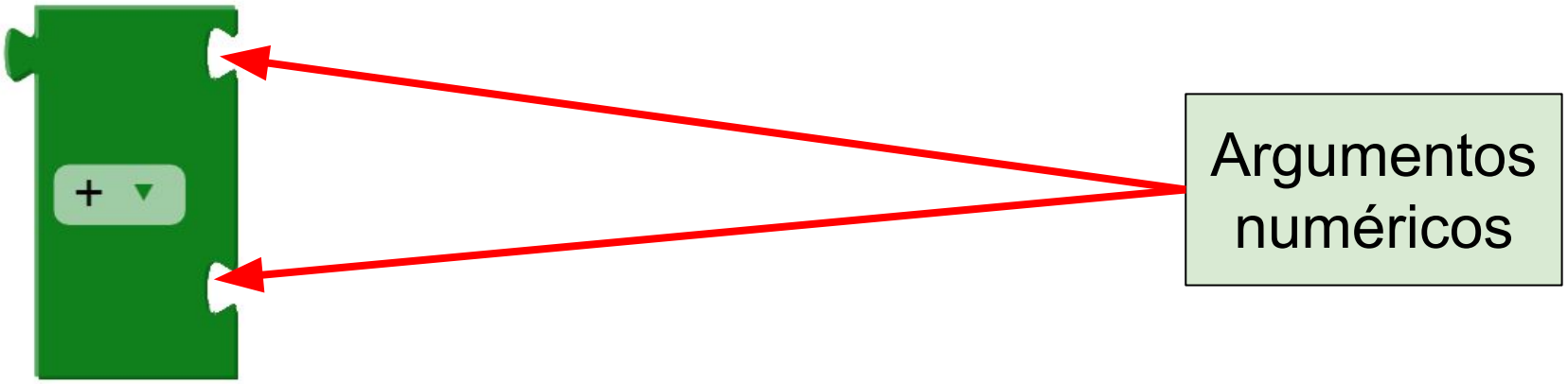


Aunque en el programa no dice el número resultante, ¡los números se suman para dar el resultado!



Operadores numéricos

- El operador de suma espera **dos argumentos**
  - Decimos que es un ***operador binario***
  - Esos argumentos deben ser de tipo número



Operadores numéricos

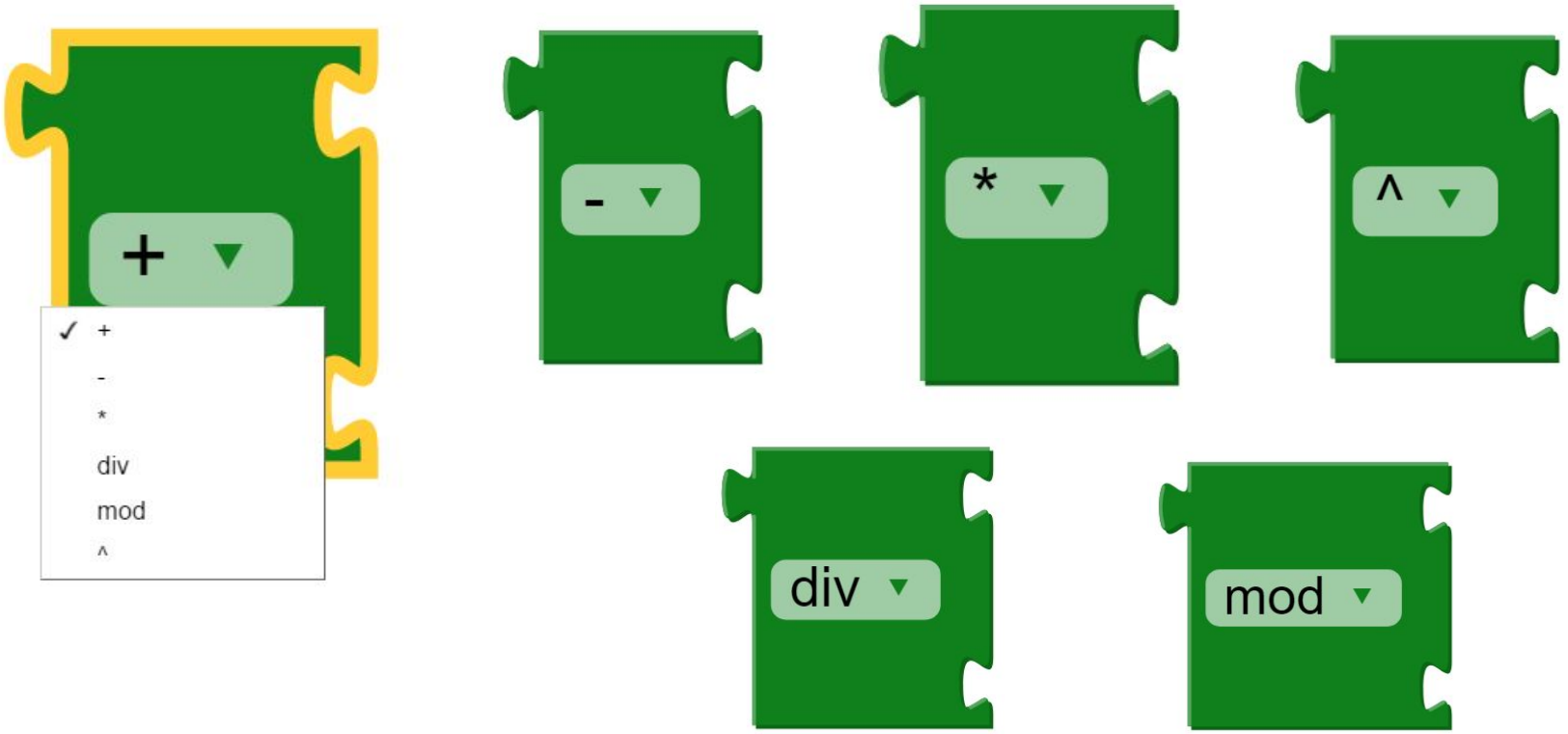
- El operador de suma espera **dos argumentos**
  - Decimos que es un ***operador binario***
  - Esos argumentos deben ser de tipo número


```
nroBolitas(Azul) + nroBolitas(Negro)
```

En texto se escribe en forma infija como es usual en matemáticas

Operadores numéricos

- Además del operador de suma hay más **operadores numéricos binarios**
  - Resta, multiplicación, potencia, división entera y resto



Operadores numéricos

- En texto, todos se escriben infijos
- Las cuentas son las que conocemos de matemáticas
  - Los operadores de **`div`** y **`mod`** son la división sin coma decimal
- Usando parámetros o expresiones primitivas se pueden obtener muchas cuentas conocidas

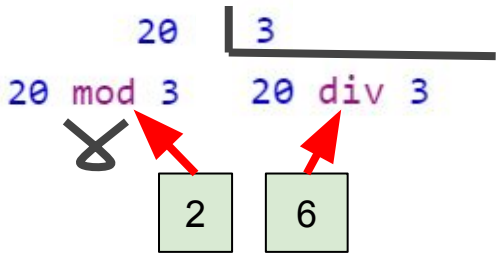
```
largoDelLado - 1
```

```
2 * nroBolitas(Azul)
```

```
radio ^ 2
```

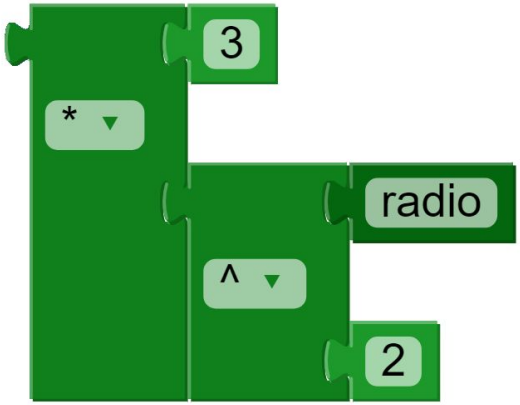
```
20 mod 3
```

```
20 div 3
```



Operadores numéricos

- Los **operadores numéricos** con argumentos…
  - ...describen un número (el resultado de la cuenta)
  - ¡Por lo tanto, se pueden usar en otras cuentas!
    - Decimos que los operadores están ***anidados***

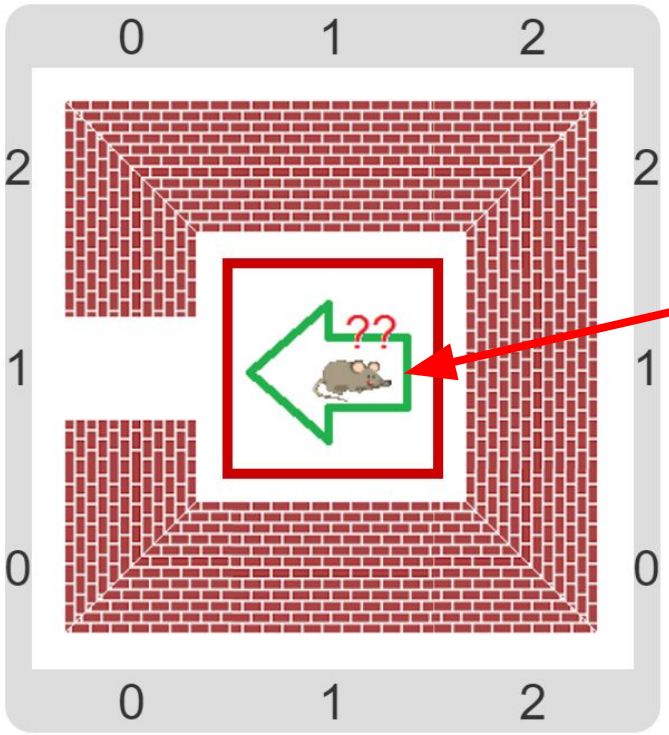



```
3 * (radio ^ 2)
```

En texto hacen falta paréntesis para que haga bien la cuenta

Funciones primitivas

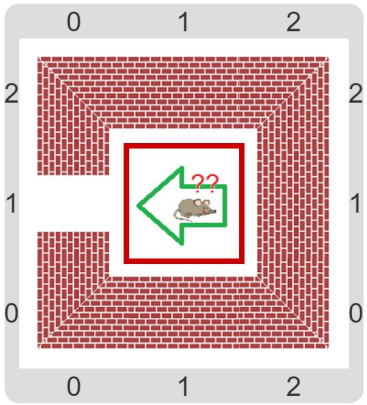
- Si en el tablero representamos elementos que no son bolitas, ¿cómo obtenemos información de ellos?
  - Las ***expresiones primitivas*** solo hablan de bolitas
  - Precisamos otra **herramienta del lenguaje**



¿Cómo saber en qué dirección apunta la flecha?
(¡Recordar que el tablero inicial puede ser otro!)

Funciones primitivas

- Las ***funciones primitivas***
  - son similares a las **expresiones primitivas**
  - pero las construye el que diseña la actividad



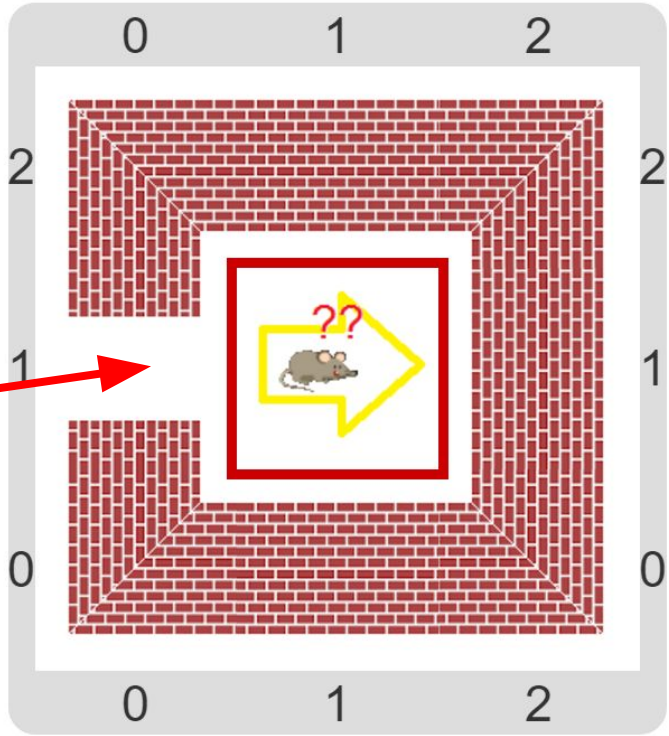
Más adelante aprenderemos a definir nuestras propias funciones

¡Es una expresión de tipo dirección!

Operadores de enumeración

- ¿Y si la flecha apunta para cualquier lado? ¿Cómo corregimos la dirección que nos informa?
  - Hay que modificar la dirección que obtuvimos
  - Precisamos otra **herramienta del lenguaje**

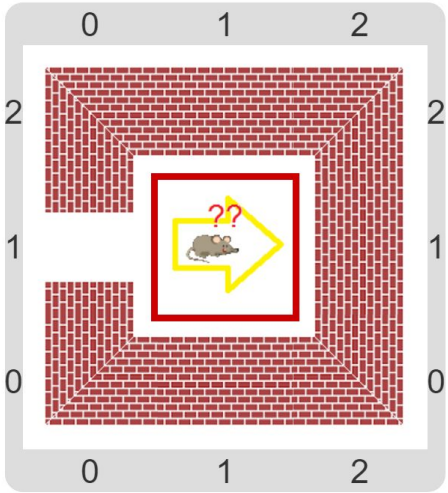
¡La dirección para salir es la ***opuesta*** de para donde indica la flecha!



Operadores de enumeración

- Los ***operadores de enumeración*** permiten modificar una dirección (y otros elementos también)
  - Son 3: **`siguiente`**, **`previo`** y **`opuesto`**

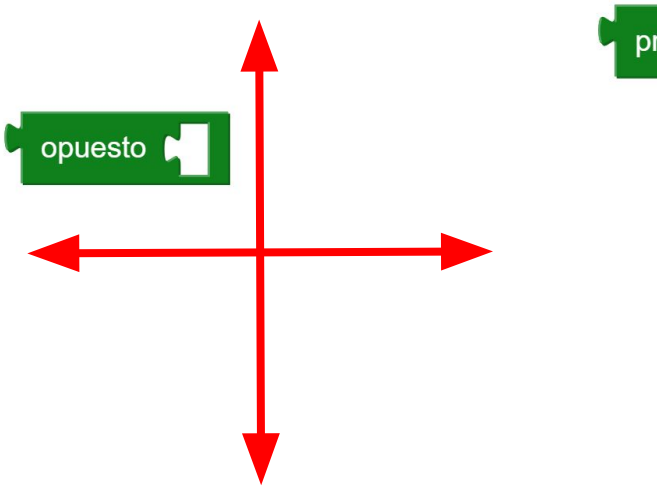
Espera un argumento de tipo dirección



Si la dirección es Este, el opuesto es Oeste

Operadores de enumeración

- **`siguiente`** avanza en el sentido de las agujas del reloj
  - **`siguiente(Norte)`** es **`Este`**, etc.
- **`previo`** avanza en el sentido contrario
  - **`previo(Norte)`** es **`Oeste`**, etc.



Hablamos de
operadores unarios

Cierre

- **Tipos de datos**
  - formas de clasificar las expresiones según la forma en que pueden utilizarse
  - los comandos primitivos, los procedimientos parametrizados y los operadores trabajan solo con cierto tipo de argumentos
  - si se proveen argumentos de tipo incorrecto, se produce un error
  - cada construcción con argumentos debe establecer los tipos de los mismos

- **Expresiones primitivas**
  - permiten sensar el tablero y obtener información
  - describen un dato que depende de la celda actual
- **Operadores**
  - los operadores numéricos permiten hacer cuentas con números (y describen un número, el resultado)
  - los operadores de enumeración permiten calcular nuevos valores en base a uno dado
- **Funciones primitivas**
  - son parecidas a las expresiones primitivas
  - pero las construye el que diseña la actividad