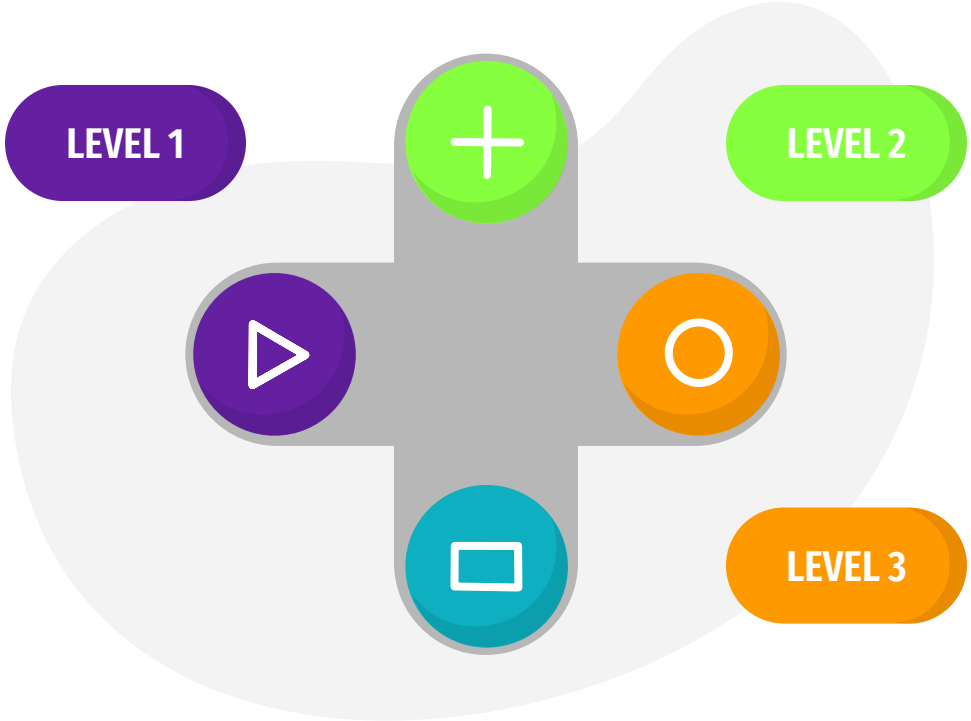
Fundamentos de Programación en Videojuegos I

Clase 2

Contenido de la clase

El entorno de PilasBloques
Primeros programas: soluciones y no soluciones
Punto de entrada, comandos y secuenciación.
Problemas computacionales: Transformaciones de estado
Infinidad de soluciones y equivalencia de programas



Procedimientos como forma de expresar la estrategia
Definición de nuevos comandos y su uso
Importancia de la comunicación

EJERCITARIO 1 - PRIMEROS PROGRAMAS: PRIMITIVAS Y PROCEDIMIENTOS

El entorno de PilasBloques
Primeros programas: soluciones y no soluciones
Punto de entrada, comandos y secuenciación.
Problemas computacionales: Transformaciones de estado
Infinidad de soluciones y equivalencia de programas

Pilas Bloques

- Pilas Bloques es un entorno de programación didáctico, en el cual se encastran bloques como si fuera un rompecabezas para crear programas.
- Pilas Bloques adopta el paradigma de programación estructurado, para un inicio sencillo a la disciplina.
- En lugar de programar una computadora (lo cual requiere comprender varios aspectos abstractos y una buena base matemática), Pilas Bloques se concentra en simulaciones. En esas simulaciones vamos a encontrar un personaje (un robot, un gato, etc.) que actúa como nuestra "computadora" y que será el encargado de llevar adelante las acciones que pidamos.
- Pilas Bloques fue creado como parte del proyecto Program.AR por la Fundación Sadosky y su uso es libre y gratuito.
- https://pilasbloques.program.ar/online

ACTIVIDAD Y TAREA DE LA CLASE 1

RESOLVER LOS EJERCICIOS DEL NIVEL PRINCIPIANTE

PROGRAMANDO EN LA COMPUTADORA:

Dieta a base de churrascos

- Desafío 1
- Desafío 2
- Desafío 3

REPETICIÓN:

Más churrascos para Duba

- Desafío 1
- Desafío 2
- Desafío 3

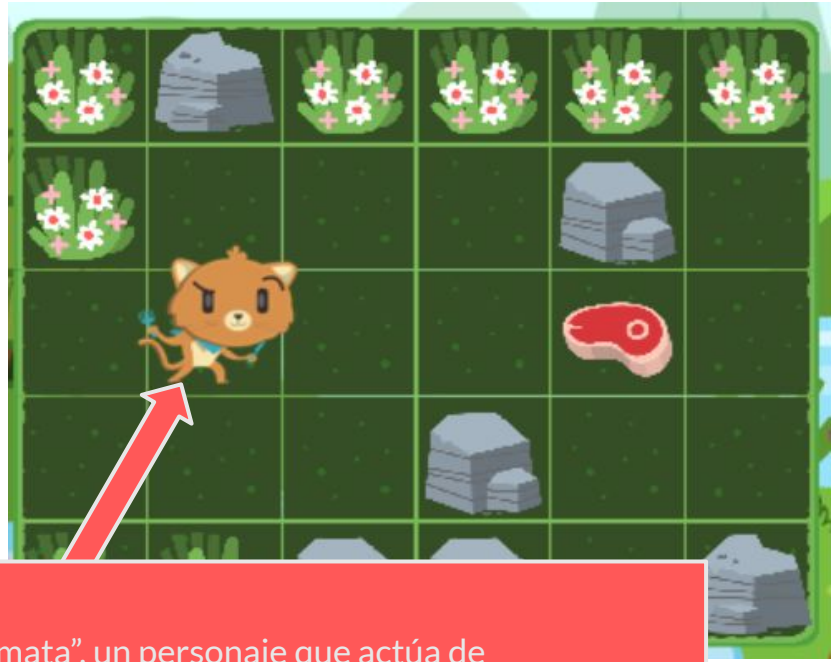
Aprendizaje por indagación

- La forma en la que vamos a trabajar está centrada en lo que llamamos “aprendizaje por indagación”.
- La propuesta es encarar actividades, intentar solucionarlas, y luego hacemos una puesta en común de lo trabajado.
- Allí vamos a ver los conceptos teóricos que pusimos en acción en la actividad realizada.
- Luego realizamos más actividades, cada vez con mayor nivel de dificultad, aplicando eso que aprendimos.
- Volvemos a repetir el mismo esquema para cada nuevo tema.

Desafío 1: ¿Qué aprendimos?

- Los bloques azules de la barra de herramientas encastran uno debajo de otro.
- Además, encastran dentro del bloque verde “al empezar a ejecutar”.
- Si los bloques los dejamos sueltos, al apretar el botón “Ejecutar” no pasa nada. Su efecto sólo se manifiesta cuando son encastrados dentro del bloque verde.
- Las acciones que lleva adelante “Duba” están dadas por los bloques que pusimos dentro de “al empezar a ejecutar”.
- Las acciones se llevan a cabo en el mismo orden que las escribimos. En secuencia.
- Cada bloque azul tiene una semántica asociada, y “Duba” realiza una acción determinada cuando en la solución figura un bloque, y no otra.

Desafío 1: ¿Qué aprendimos?



Autómata

Los problemas de PilasBloques se basan en un “Autómata”, un personaje que actúa de computadora, y a quien le daremos órdenes para que realice diversas acciones.
En estas actividades es Duba.
En un entorno profesional, el autómata es la computadora en sí misma.

Desafío 1: ¿Qué aprendimos?

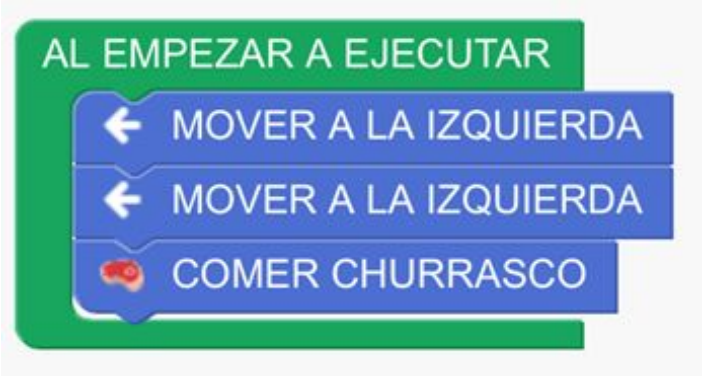
- Ante un problema…

Desafío 1: ¿Qué aprendimos?

- Hay programas que solucionan el mismo.

- Ante un problema…

Desafío 1: ¿Qué aprendimos?



- Hay programas que no solucionan el problema planteado.

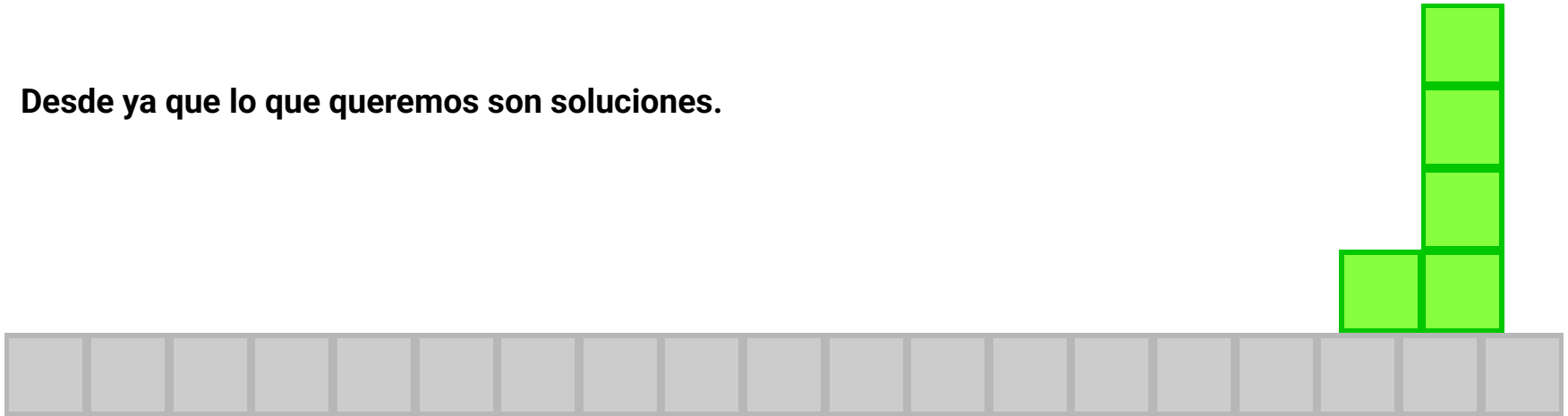
- Ante un problema…

Soluciones y No soluciones

- **Cuando un programa soluciona el problema planteado, decimos que es una solución al problema, o decimos que es un programa correcto para el problema dado.**

- **Cuando no lo hace, simplemente no es un programa correcto (es incorrecto) o también podemos decir que no es una solución al problema.**

Desde ya que lo que queremos son soluciones.



Punto de entrada

Punto de entrada

El “bloque de programa” es un bloque especial, que va a aparecer siempre, y es único en toda solución.
Es el bloque que se ejecuta al momento de presionar “ejecutar”.
En programación se conoce a esto como “punto de entrada de un programa”.
Todos los lenguajes suelen incluir algún punto de entrada, para indicarle a la computadora cómo debe ser leída la solución (es decir, por dónde se arranca a leer).

Comandos

Bloques de comandos

Un comando es la descripción de una acción. Cada bloque azul, refleja entonces una acción que Duba (o nuestra computadora en general, sea quien sea el personaje u objetivo del comando) deberá llevar adelante cuando presionemos “Ejecutar”.
Cada comando tiene una semántica asociada, la cual es única y no ambigua.

Secuenciación

Secuenciación

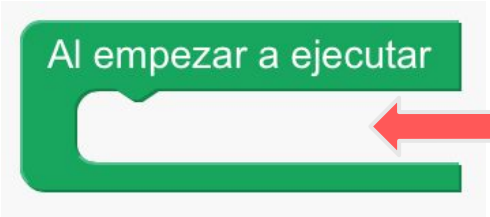
La forma más simple de disponer los comandos es mediante la secuenciación, es decir, uno después de otro.

En este entorno, esa secuenciación está dada de arriba a abajo.

Eso quiere decir que hay bloques que están “primero” y bloques que están “después” en el orden de ejecución.

Cuando presionemos el botón “Ejecutar”, la computadora llevará adelante las acciones asociadas a cada comando en orden.

Cuerpo



Cuerpo
Un cuerpo es un “agujero” dentro de los bloques, que debe completarse con comandos. El “punto de entrada” tiene un cuerpo, y los comandos de ese cuerpo son los que se ejecutan al momento de presionar “Ejecutar”.

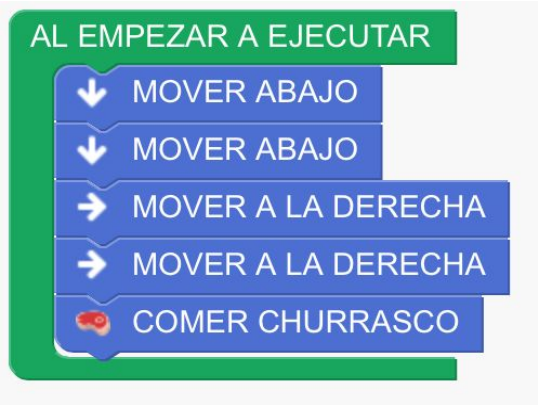
Problemas computacionales: Transformación de estado

- **Acá no hay transformación de información, sino de estado (que es información en la memoria de una computadora).**

- **Cambiamos el estado del escenario, entre “antes de ejecutar” llamado “estado inicial” a aquel en el que termina “después de ejecutar” el programa, llamado “estado final”.**

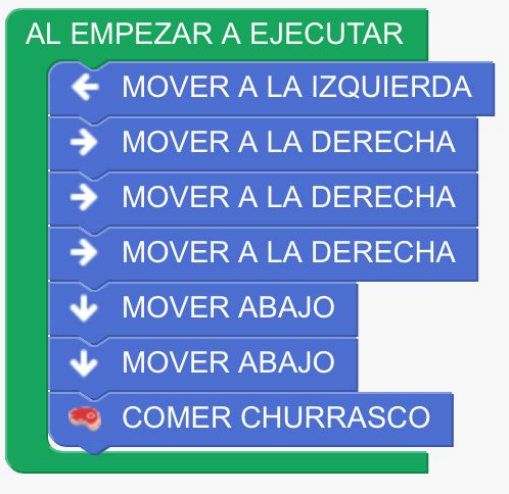
- **Cada comando realiza un pequeño cambio de estado, y el programa pasa por varios estados intermedios antes de llegar al estado final.**

Desafío 5: ¿Qué aprendimos?



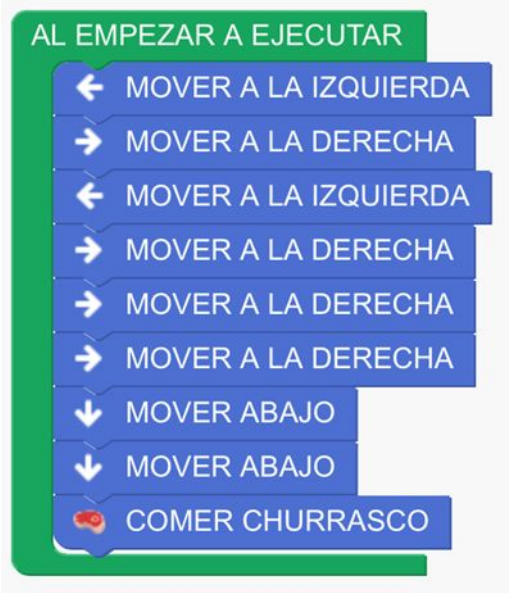
¿Cuál es la solución correcta?

Desafío 5: ¿Qué aprendimos?



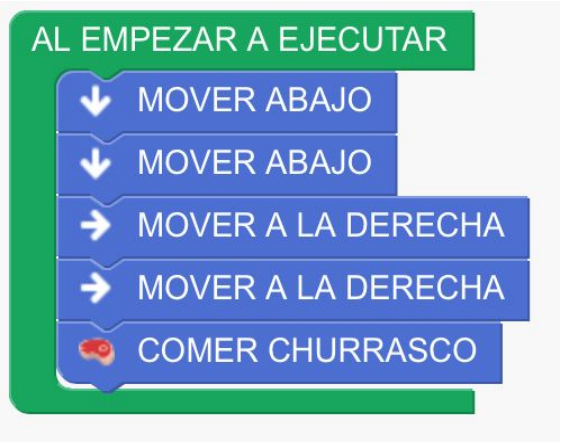
¿Y esta?

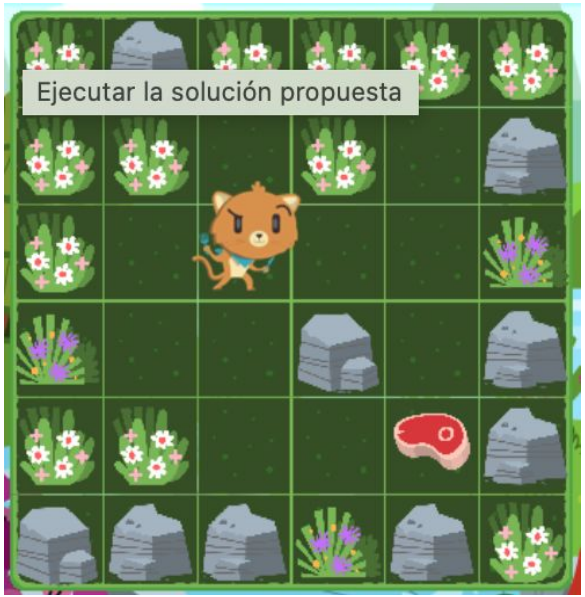
Desafío 5: ¿Qué aprendimos?

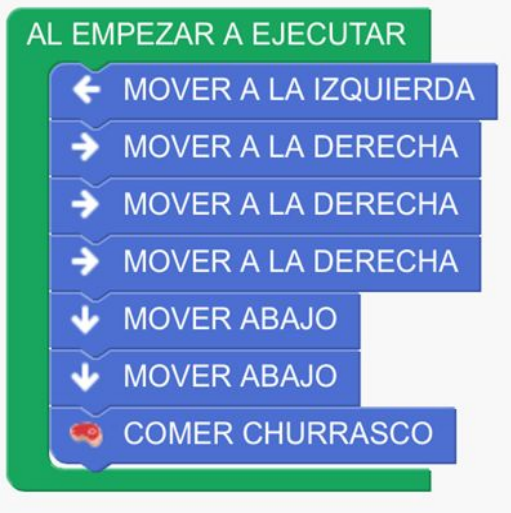


¿Y esta?

Desafío 5: ¿Qué aprendimos?







¿Cuál es la solución adecuada?

Programas equivalentes

- **Un problema admite en principio infinitas soluciones.**
- **A veces las soluciones posibles están limitadas por las herramientas posibles, pero con todas las herramientas a disposición, hay infinitas.**
- **Si dos programas solucionan el mismo problema de formas diferentes, decimos que son programas equivalentes.**
- **Aunque todas resuelvan el problema, hay soluciones que son adecuadas (en términos de la materia), y otras que no lo son.**

Fallos

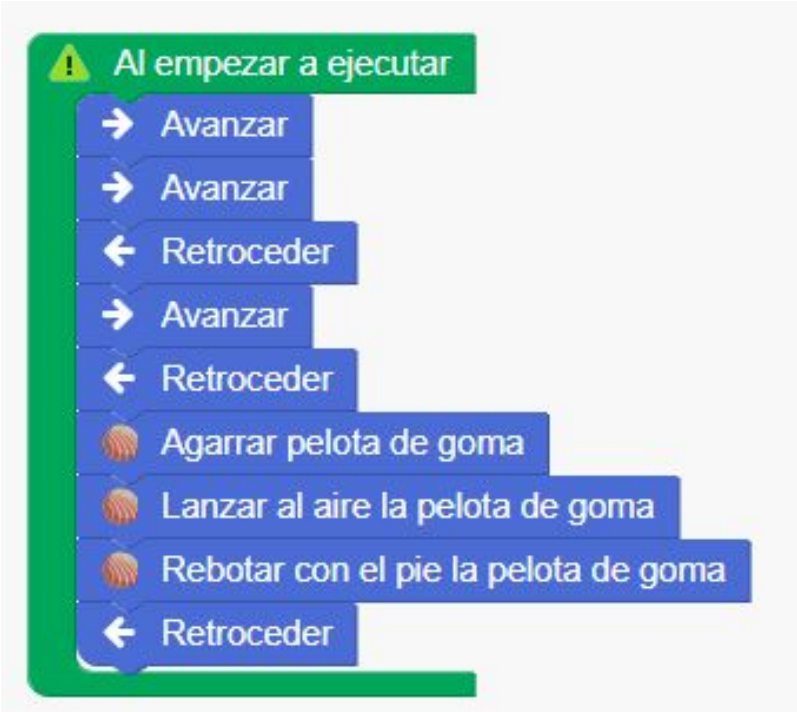
- **Un programa puede fallar por varios motivos.**
- **Uno de los motivos más comunes (y el único por ahora) es pedirle a la computadora que ejecute una acción que no puede realizar (por ej. pedirle a Duba que se mueva a la derecha cuando no hay más lugar a la derecha)**
- **Entonces, no cualquier secuencia de comandos funciona. Hay que saber qué acciones se pueden realizar, en qué contexto.**

PROCEDIMIENTOS COMO FORMA DE EXPRESAR LA ESTRATEGIA

A trabajar

Realizamos del Nivel Intermedio Autómatas, comandos, procedimientos y repetición “Chuy hace jueguito”.

“Chuy hace jueguito”: ¿Qué aprendimos?



Primera aproximación

“Chuy hace jueguito”: ¿Qué aprendimos?

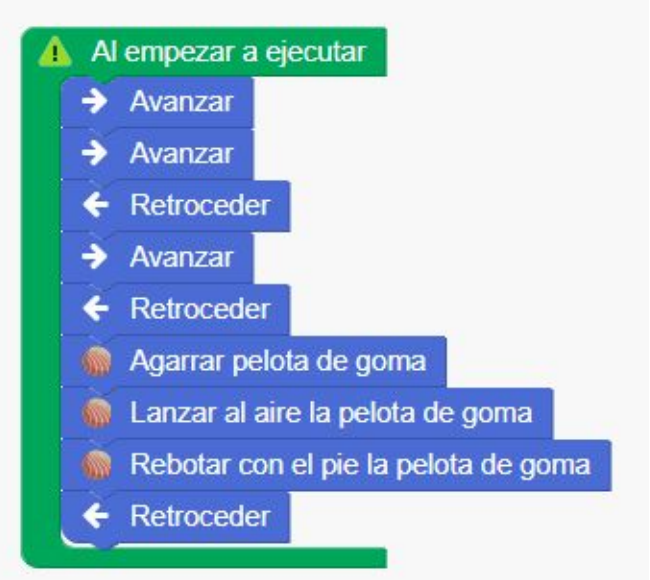
Volvamos a leer el enunciado:

Hacé que Chuy avance un paso, entre en calor (avance y retroceda dos veces), recoja la pelota de goma, haga jueguito lanzando al aire la pelota y rebotándola con el pie y, por último, vuelva a su lugar.

Claramente hay 5 elementos en el problema, 5 cosas que Chuy debe realizar.

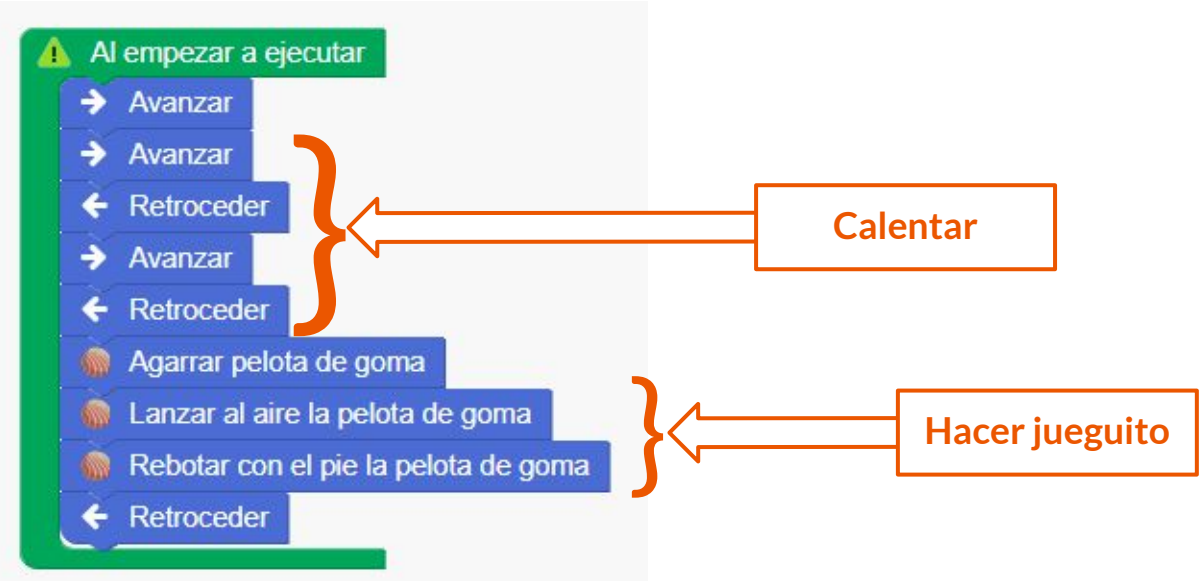
“Chuy hace jueguito”: ¿Qué aprendimos?

Pero sí el problema implica hacer 5 cosas, ¿Por qué nuestra solución tiene 9 elementos?



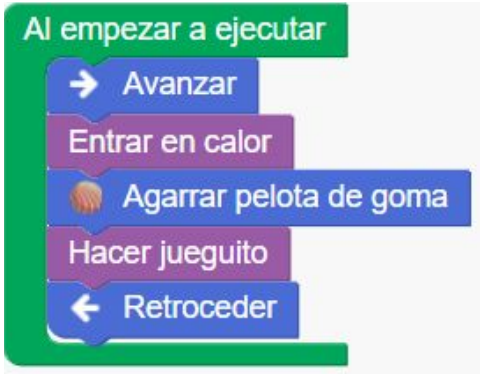
“Chuy hace jueguito”: ¿Qué aprendimos?

Podemos pensar que hay grupos de comandos que corresponden a una misma cosa a nivel conceptual:



“Chuy hace jueguito”: ¿Qué aprendimos?

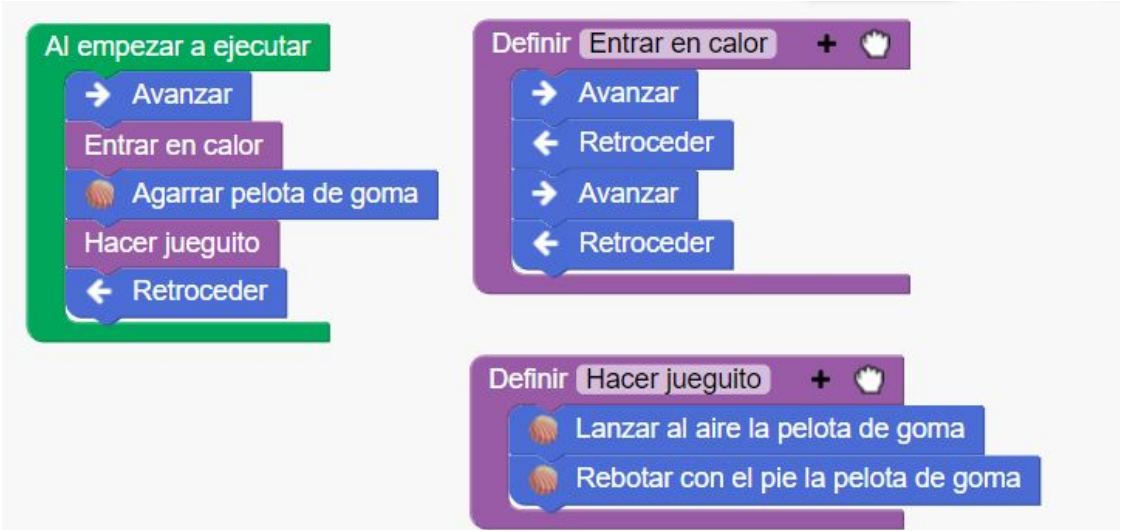
Podemos pensar que hay grupos de comandos que corresponden a una misma cosa a nivel conceptual:



Lo que queremos

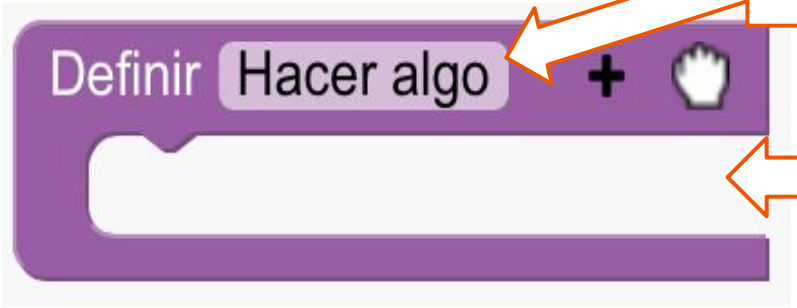
“Chuy hace jueguito”: ¿Qué aprendimos?

Podemos pensar que hay grupos de comandos que corresponden a una misma cosa a nivel conceptual:



Podemos lograr esto gracias a los procedimientos

Procedimientos



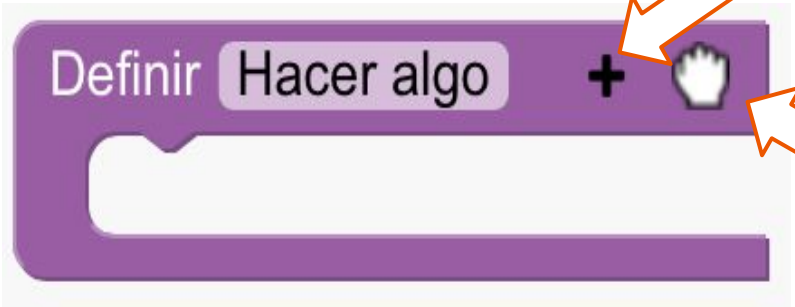
Nombre

Notar que se puede escribir en ese espacio para elegir un nombre apropiado.

Cuerpo

Recordemos que un cuerpo se completa con comandos.

Procedimientos

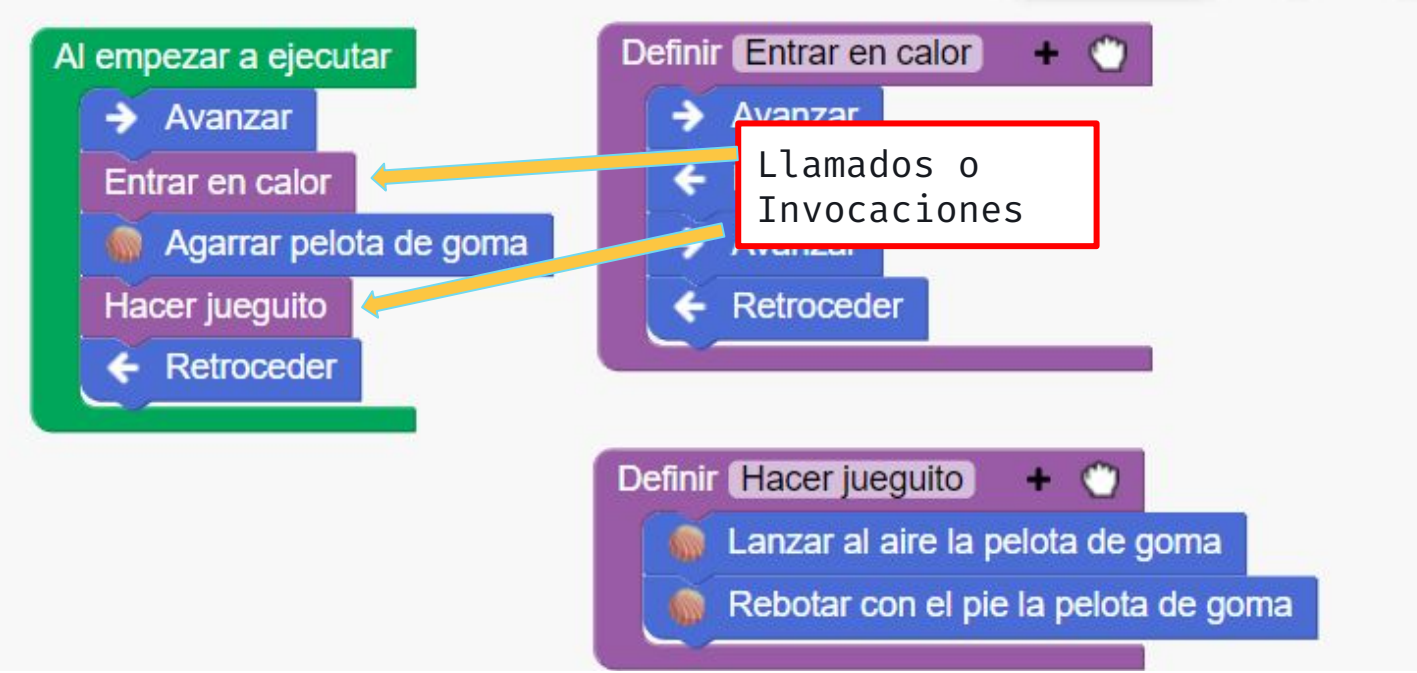


Ignorar
No vamos a usar este botón por ahora.

Manito
Permite obtener el comando asociado a esta definición.
También se puede encontrar en la sección “Mis procedimientos” en la barra de herramientas.

Procedimientos

Procedimientos



Procedimientos

Características

1. Un procedimiento tiene un nombre y un cuerpo.
2. El procedimiento se define, por un lado, y luego se invoca (se usa como un comando más)
3. Su nombre siempre comienza con un verbo en infinitivo.

Procedimientos - Ventajas

1	Crear nuevos comandos	Los procedimientos son una herramienta del lenguaje que permite al programador definir nuevos comandos.
2	Claridad y comunicación	Brindan claridad en nuestra solución, permitiendo comunicar claramente las ideas que el programador tenía en la cabeza al momento de realizar el código.
3	Legibilidad	Aportan legibilidad, haciendo que la solución se lea como una historia.
4	Modificabilidad	Proveen mayor modificabilidad, ya que con procedimientos es más fácil modificar una solución existente.
5	Reutilización	La definición puede ser única, pero puede invocarse muchas veces, en lugares distintos, permitiendo reutilizar código.

Procedimientos - Ventajas

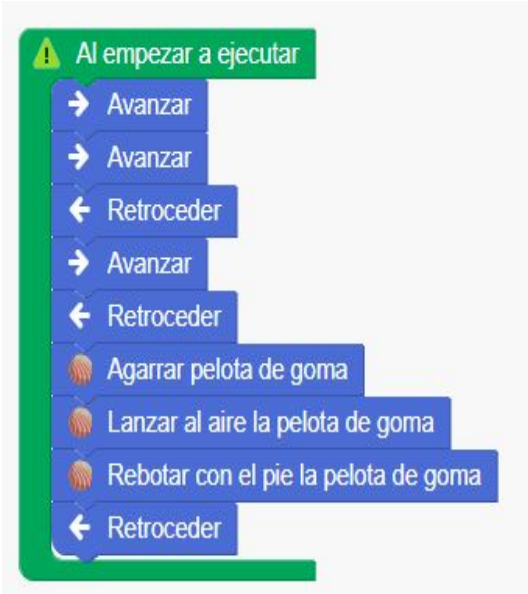
6	Estructuración	Van a permitir realizar soluciones más simples (cada procedimiento puede ser pensado como un programa independiente del resto, pero que resuelve solo una partecita del problema total). Es decir, sirven para aportar estructura al código, y pensar en términos de subtareas.
7	Reducción de errores	Reducen la cantidad de errores, ya que el código no se duplica y es más fácil identificar puntos de fallo en caso de que los hubiera.

Programar es comunicar

- Programar implica comunicar. Comunicamos la solución a la máquina (motivo por el cual tenemos que escribirla en un lenguaje de programación), quien la ejecutará para solucionar finalmente el problema.
- Pero los programadores no trabajan solos en la mayoría de los casos. También tenemos que comunicar a las personas, para que otros entiendan nuestras soluciones, y que las ideas y cosas que pensamos al solucionar un problema sean claras. Si comunicamos solo en español, las personas entienden, pero la máquina no.
- Es importante entender esta dualidad y tenerla siempre presente.
- Así, hay soluciones que son correctas, pero no adecuadas porque no comunican correctamente a las personas.

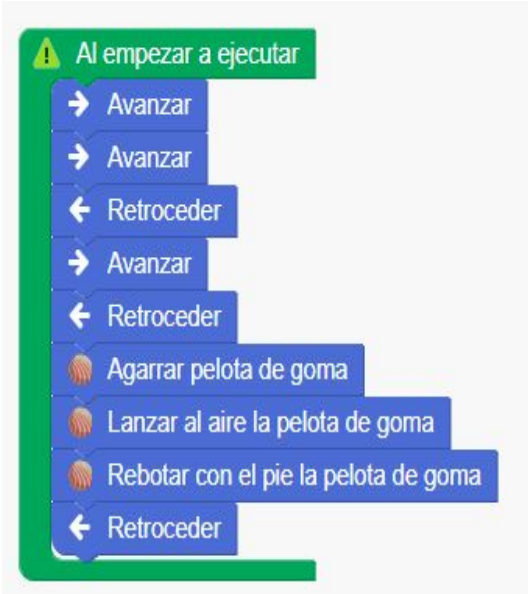
Programar es comunicar

Solución adecuada, las ideas se comunican claramente.



Solución inadecuada, funciona, pero no se entiende.

Programar es comunicar



Notar que la solución inadecuada es más corta, pero esto no la hace correcta. No buscamos la solución con menos bloques, sino la que es más clara en expresar las ideas.

Programar es comunicar

- Vamos a hablar entonces de “buenas prácticas” en la programación cada vez que hagamos referencia a aquellas pequeñas cosas que podemos hacer de manera más adecuada para aportar claridad a nuestro programa y facilitar así la comunicación de cómo éste resuelve el problema.

- Dicho de otro modo, no solo debemos preocuparnos porque el programa funcione, sino que también deberemos estar atentos y hacer buen uso de ciertas prácticas como por ejemplo la que hoy aprendimos: hacer una buena subdivisión de tareas utilizando procedimientos que resuelvan partes concretas de nuestro problema.

ACTIVIDAD Y TAREA DE LA CLASE 2

RESOLVER EL EJERCITARIO 1 - PRIMEROS PROGRAMAS: PRIMITIVAS Y PROCEDIMIENTOS