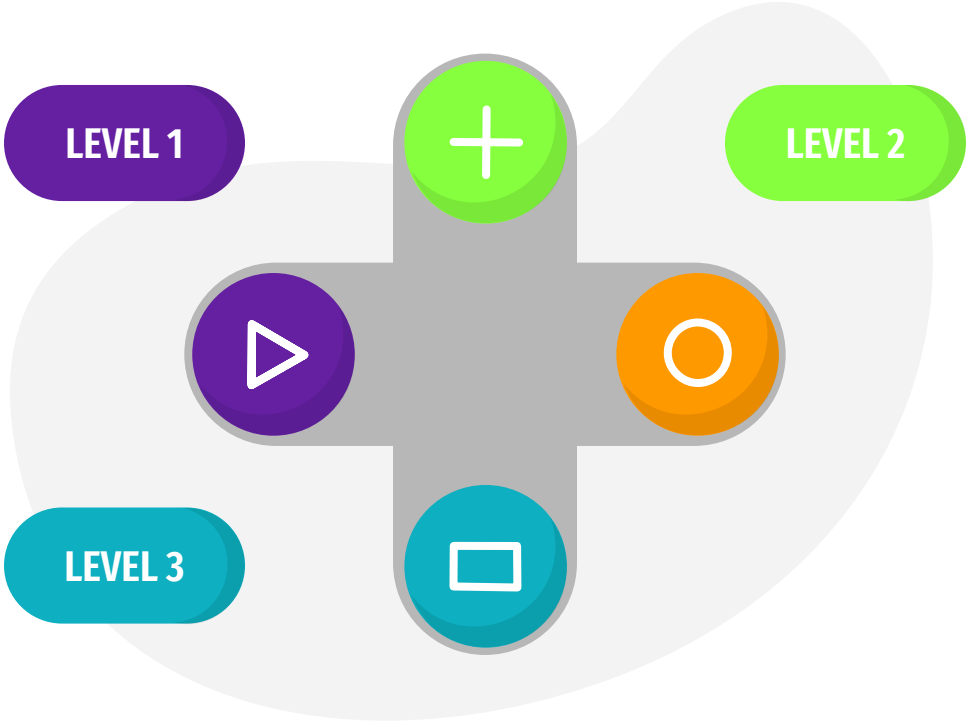
Fundamentos de Programación en Videojuegos I

Clase 6

Contenido de la clase

Repetición Simple: Forma y utilidad
Expresiones (Números)
Control de flujo de la repetición

Estrategia de solución



Legibilidad y Comunicación
Importancia de no anidar estructuras
Elección de buenos nombres: propósito y verbos

Repetición Simple: Forma y utilidad

A trabajar

Realizamos del libro del “Nivel Intermedio”, el ejercicio “No me canso de rebotar”, de la sección “Autómatas, Comandos, Procedimientos y Repetición”.

No me canso de rebotar: ¿Qué aprendimos?

Primera aproximación

¡¡¡Horrible!!!

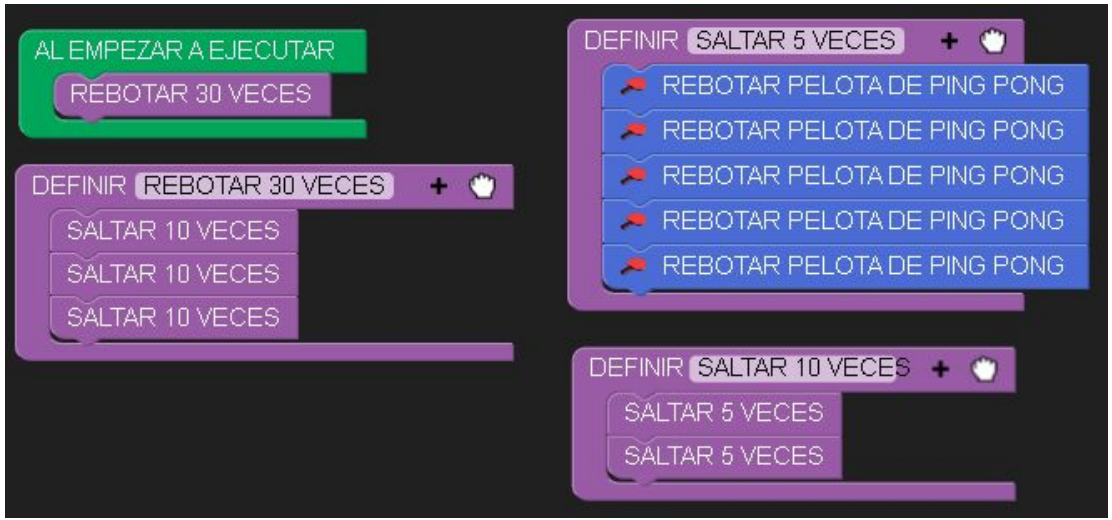
Ya a esta altura, es claro que esta no es una solución adecuada.

No me canso de rebotar: ¿Qué aprendimos?

Segunda aproximación

Mucho mejor. Hace uso de las herramientas aprendidas, y deja más clara la idea. Se puede mejorar haciendo más procedimientos.

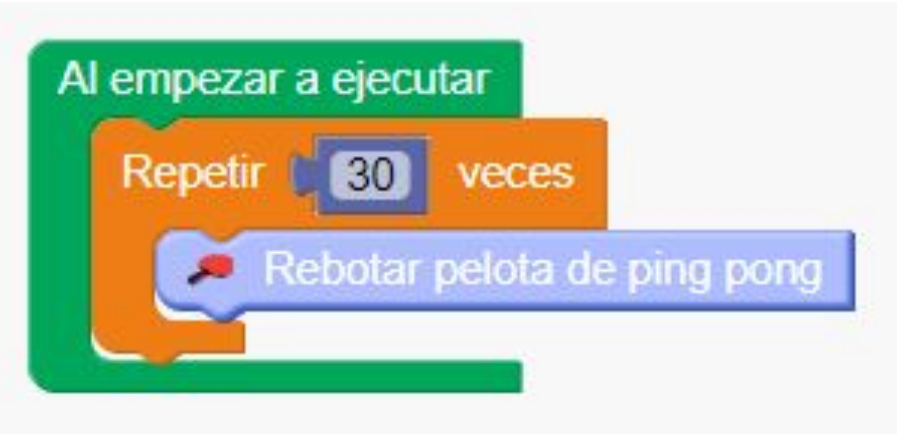
No me canso de rebotar: ¿Qué aprendimos?



Tercera aproximación

¡Muy bien!. Las ideas quedan claras, y bien expresadas en términos de procedimientos, pero podríamos agregar una herramienta ya conocida…

No me canso de rebotar: ¿Qué aprendimos?



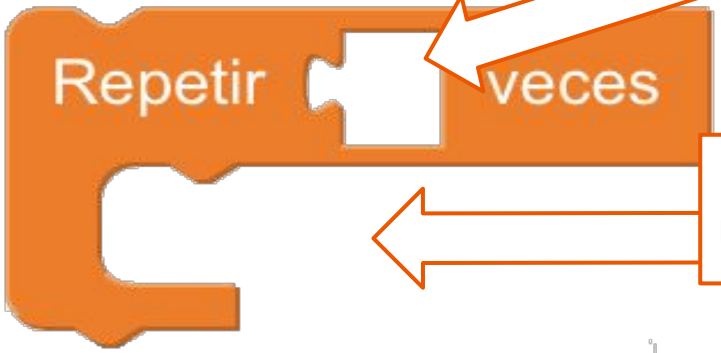
Cuarta aproximación

¡Excelente!. Esta sería una excelente solución a este problema, que hace muy buen uso de la nueva herramienta.

Repetición Simple

- La repetición simple es un comando. Notar que encastra dentro de otros cuerpos, igual que los comandos primitivos (los azules) o los definidos por procedimiento (los violetas).
- Este comando tiene una semántica especial, ya que permite cambiar la forma en la que se ejecuta el código (cambia el flujo natural del programa)
- El comando tiene 2 partes importantes, un cuerpo y una expresión numérica.
- Es lo que llamamos un comando compuesto, ya que es un comando que tiene un cuerpo.

Repetición Simple



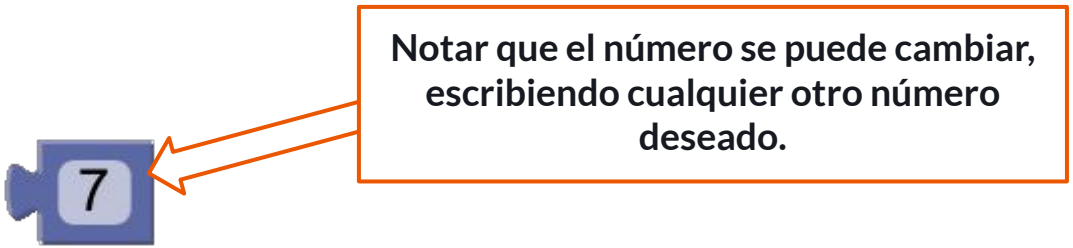
Expresión numérica
Un número que indica la cantidad de veces a repetir

Cuerpo
Una serie de comandos que quieren repetirse.

Expresiones

- Una expresión es la descripción de información (de un dato, un valor).
- Las expresiones no pueden estar solas en un cuerpo (En Pilas Bloques es imposible hacerlo, porque directamente no encastran, ya que su forma es distinta a la de los comandos).
- Sirven para brindar información a los comandos, y alterar de forma coincidente la semántica del mismo.
- Hasta ahora, solo los números literales, que usamos en la repetición.

Expresiones



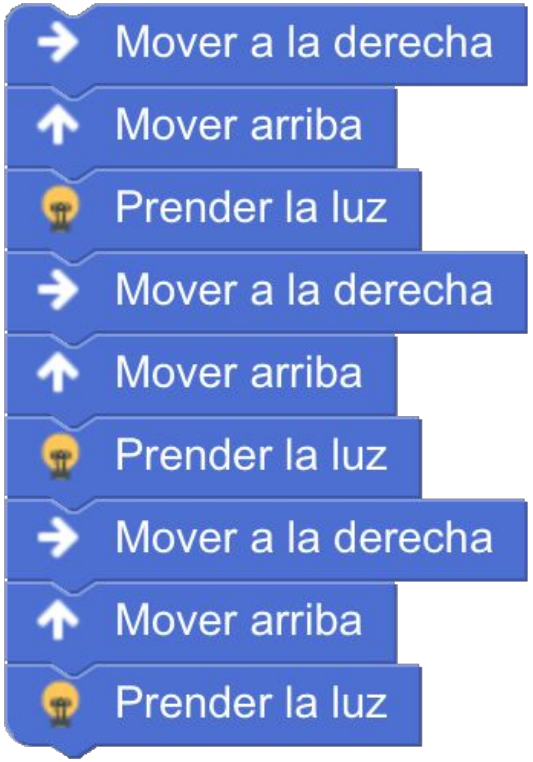
- El entorno permite ingresar números naturales, enteros y hasta racionales.
- Por ahora solo usamos esta expresión para las repeticiones, por lo que ¿tiene sentido decir “repetir -4 veces”? ¿O “repetir 7,2 veces”?
- A pesar de que el entorno permita ciertos números, no cualquiera tiene sentido.

Control de flujo: Repetición

- La REPETICIÓN es otra forma en la que puede organizarse el código, además de la secuencia.
- La semántica de un bloque de repetición implica ejecutar los bloques del cuerpo de ese comando, tantas veces como sea necesario hasta completar la repetición. En la repetición simple, tantas veces como indica la expresión numérica asociada.
- Notar que los comandos dentro de la repetición se ejecutan en secuencia.

Control de flujo: Repetición

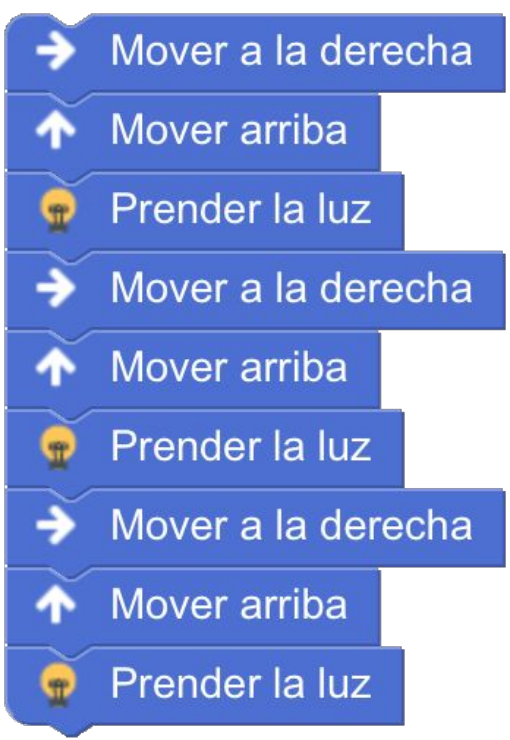
=?



Control de flujo: Repetición

=?

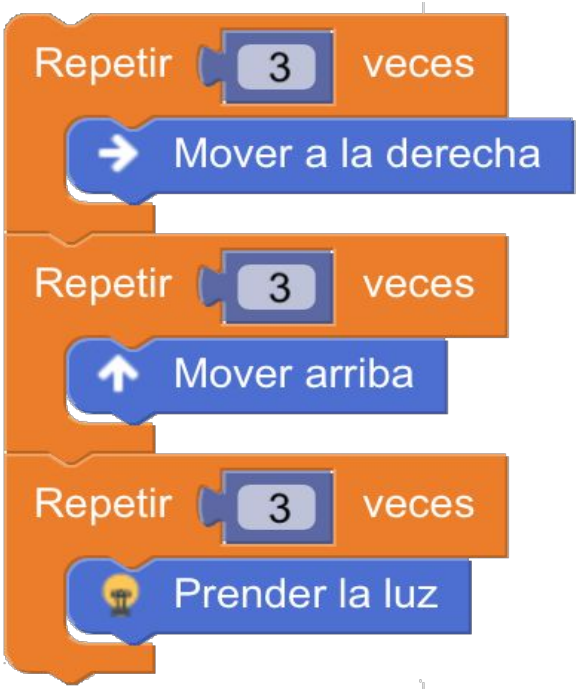
Control de flujo: Repetición



=?

Control de flujo: Repetición

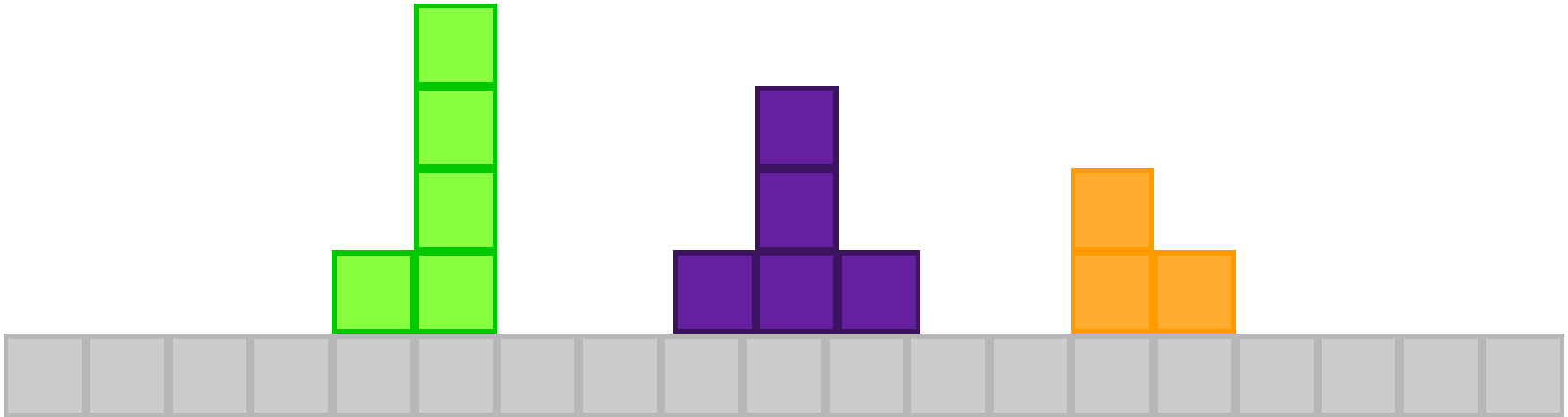
=?



Legibilidad y Comunicación
Importancia de no anidar estructuras
Elección de buenos nombres: propósito y verbos

ACTIVIDAD

Realizamos del libro del “Nivel intermedio”, el ejercicio “Reparadora de telescopios”, de la sección “Autómatas, Comandos, Procedimientos y Repetición”.



Reparadora de telescopios: ¿Qué aprendimos?

Nuevamente, ¿infinitas soluciones?

Reparadora de telescopios: ¿Qué aprendimos?

Las primitivas disponibles en este ejercicio hacen que algunas estrategias no sean posibles.

Por ejemplo, no se puede elegir comenzar por los telescopios de arriba, pues no hay primitiva para mover hacia abajo.

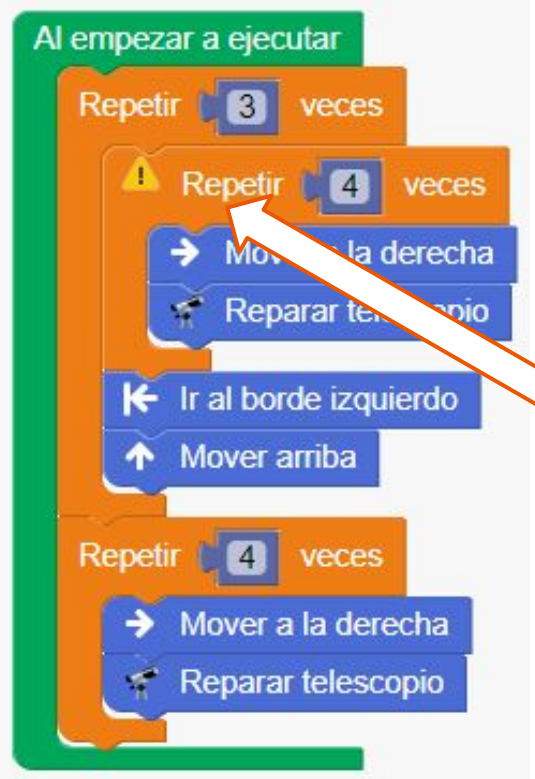
Reparadora de telescopios: ¿Qué aprendimos?

Primera aproximación

¡¡¡Horrible!!!

¿Y los procedimientos?

Anidación de código



Vamos a incorporar otra buena práctica a la programación

Repeticiones anidadas

Cuando eso sucede decimos que se está "anidando" (la idea de colocar un bloque dentro de otro bloque, en un cuerpo).

Anidación de código

Como regla general podemos decir que:

¡NUNCA, NUNCA, JAMÁS DE LOS JAMASES DEBERÍA HABER UN BLOQUE NARANJA DENTRO DE OTRO!

Anidación de código

En realidad, siempre van a haber elementos anidados, como **“Mover a la derecha”** que está anidado dentro del repetir, pero el problema son los **“bloques naranjas”**.

Estos bloques cambian el flujo del programa, y al anidarlos es difícil entender qué hace el programa sin tener que ejecutarlo en nuestra cabeza.

Dicho de otra forma, anidar estructuras de control, no comunica correctamente las ideas.

Reparadora de telescopios: ¿Qué aprendimos?

Segunda aproximación

¡¡¡Hermosa!!!

Los procedimientos ayudan a entender claramente las ideas.

Notar como en la estrategia se aprecia la idea de la repetición de reparar por filas, y se ve claramente el caso de borde.

La importancia de los nombres

¿Y en este caso?

El código es “casi” igual que la versión anterior, pero cambian los nombres de los procedimientos.

¿Queda clara la estrategia?

La importancia de los nombres

Hay nombres de procedimiento buenos, y nombres malos. Recordamos:

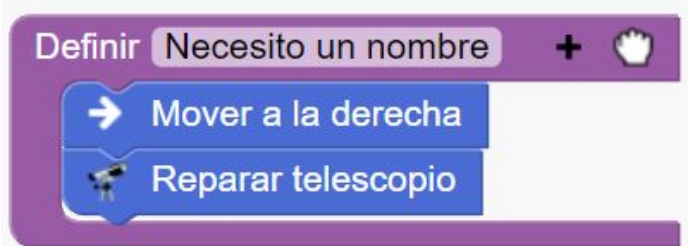
- Un procedimiento es una forma de definir un comando.
- Un comando es la descripción de una acción
- ¿Con qué expresamos acciones en el español?

Los verbos son la forma que tenemos de expresar acciones. Un buen nombre de procedimiento debería comenzar con un verbo, para describir claramente la acción que se va a realizar.

Además, usamos verbos en infinitivo (ej. Mover, en lugar de Mueve).

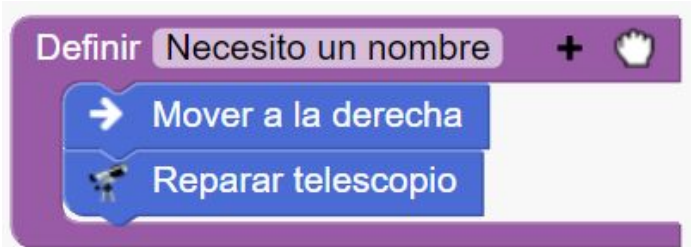
¿Cuáles son buenos nombres?

- Arreglar los telescopios de la fila
- Arreglar
- Arregla
- Arreglar telescopios
- Telescopios
- Arreglar telescopios fila
- Reparar telescopio a la derecha
- Reparar telescopios de la fila
- Reparar muchos telescopios



¿Cuáles son buenos nombres?

- Arreglar los telescopios de la fila
- Arreglar
- Arregla
- Arreglar telescopios
- Telescopios
- Arreglar telescopios fila
- Reparar telescopio a la derech
- Reparar telescopios de la fila
- Reparar muchos telescopios



Tratemos de escribir en español y no sonar como neardentales

Nombres y propósito

Como regla del pulgar podemos decir que un nombre es bueno si al leerlo, nos queda más que claro qué es lo que va a suceder en el escenario, sin siquiera tener que mirar el código de ese procedimiento.

Podemos decir que las reglas son:

- Comienza con un verbo en infinitivo
- Deja en claro el propósito del procedimiento
- Está escrito en español, claro y legible

Repeticiones anidadas

Agregamos entonces a las buenas prácticas, el de colocar buenos nombres a los procedimientos y evitar la anidación

Nombres y propósito

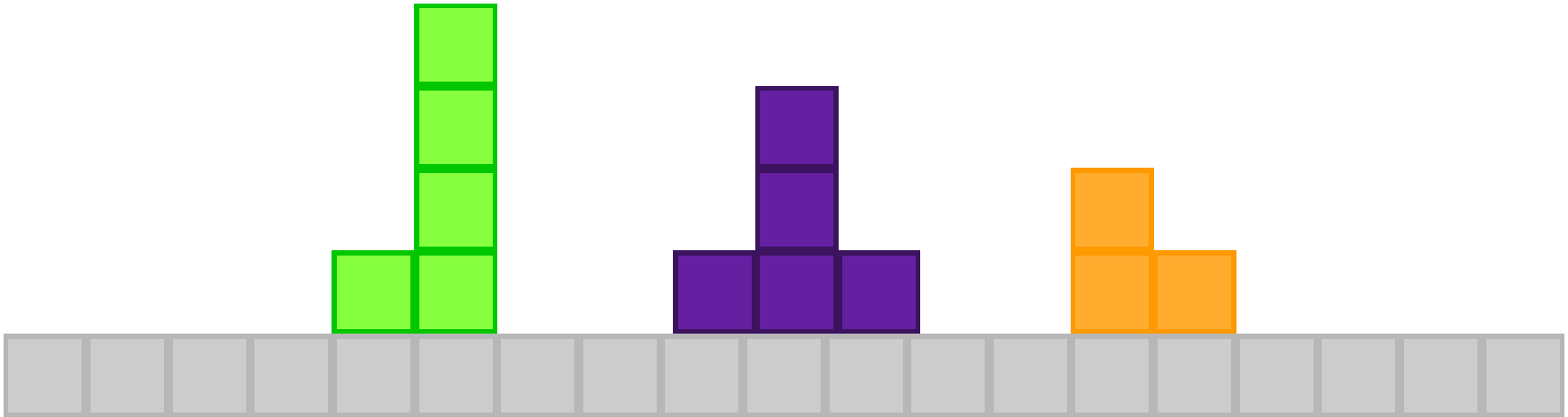
Recordatorio:

- **¡¡¡Programar es comunicar!!!**
- Tus programas deberían quedar claros a partir de la lectura.
- Sí comunica mal, no es adecuado. Sí no es adecuado, no sirve para la materia.
- Sí leo el punto de entrada, tiene que quedar más que explicita la estrategia elegida.
- Usamos procedimientos para la claridad, legibilidad y expresar la estrategia.
- Es importantísimo elegir nombres adecuados para los procedimientos que definimos.

Estrategia de solución

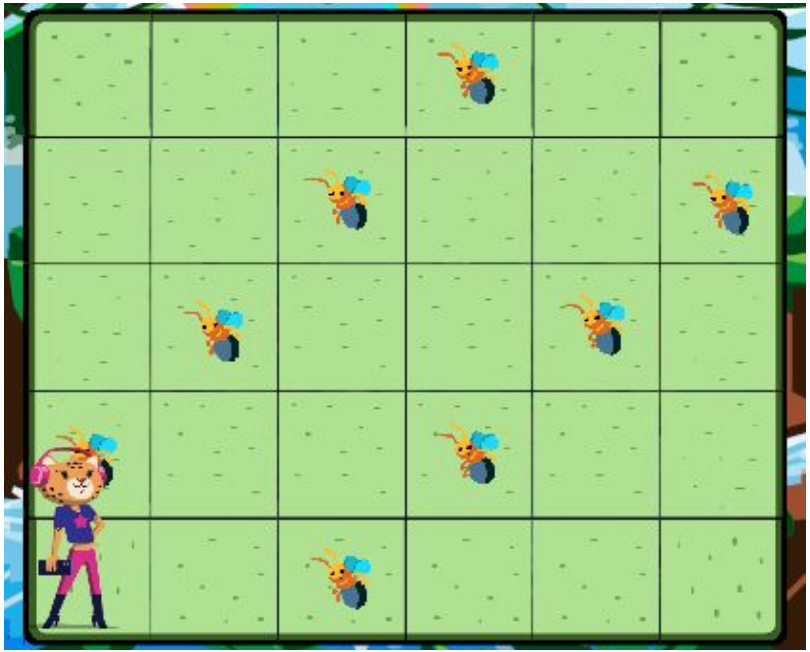
ACTIVIDAD

Realizamos del libro del “Nivel intermedio”, el ejercicio “Yvoty y las luciérnagas”, de la sección “Autómatas, Comandos, Procedimientos y Repetición”.



Yvoty y las luciérnagas: ¿Qué aprendimos?

¿Cuántas soluciones posibles hay?



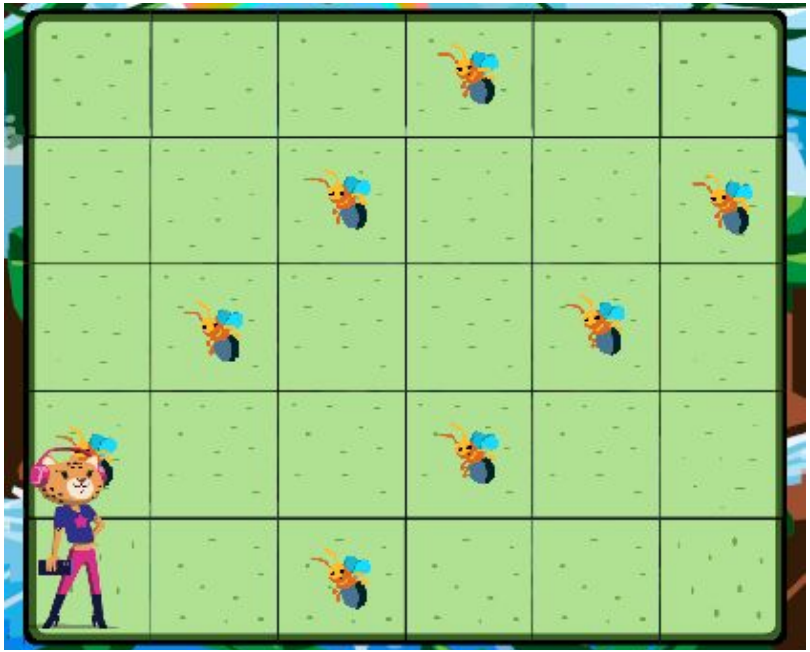
Yvoty y las luciérnagas: ¿Qué aprendimos?

Infinitas, claro.

Las variaciones de procedimientos, estrategia, nombres, etc. hacen que sea difícil encontrar dos soluciones idénticas.

Podemos clasificarlas según la estrategia general usada.

Ya a esta altura no vamos a analizar soluciones incorrectas o poco adecuadas.



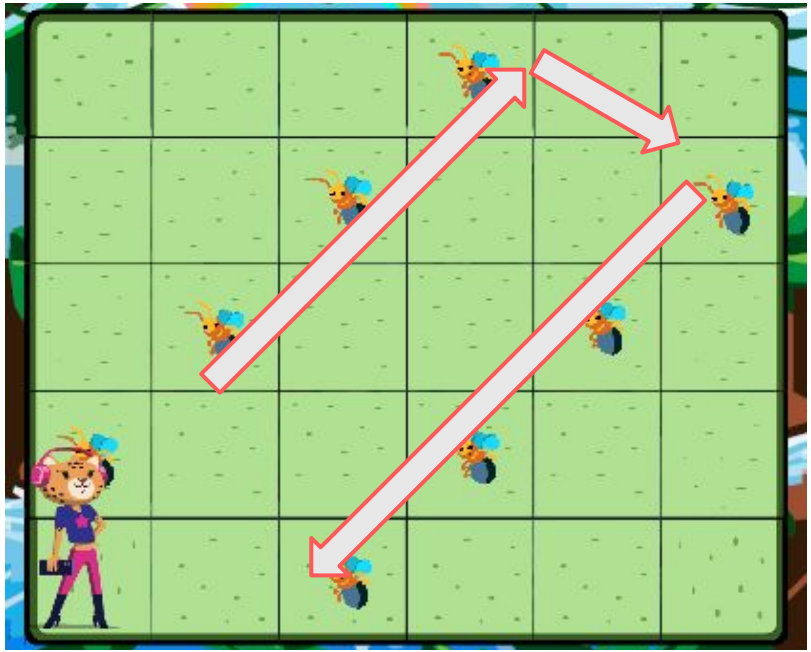
Yvoty y las luciérnagas: ¿Qué aprendimos?

Estrategia de subir y bajar

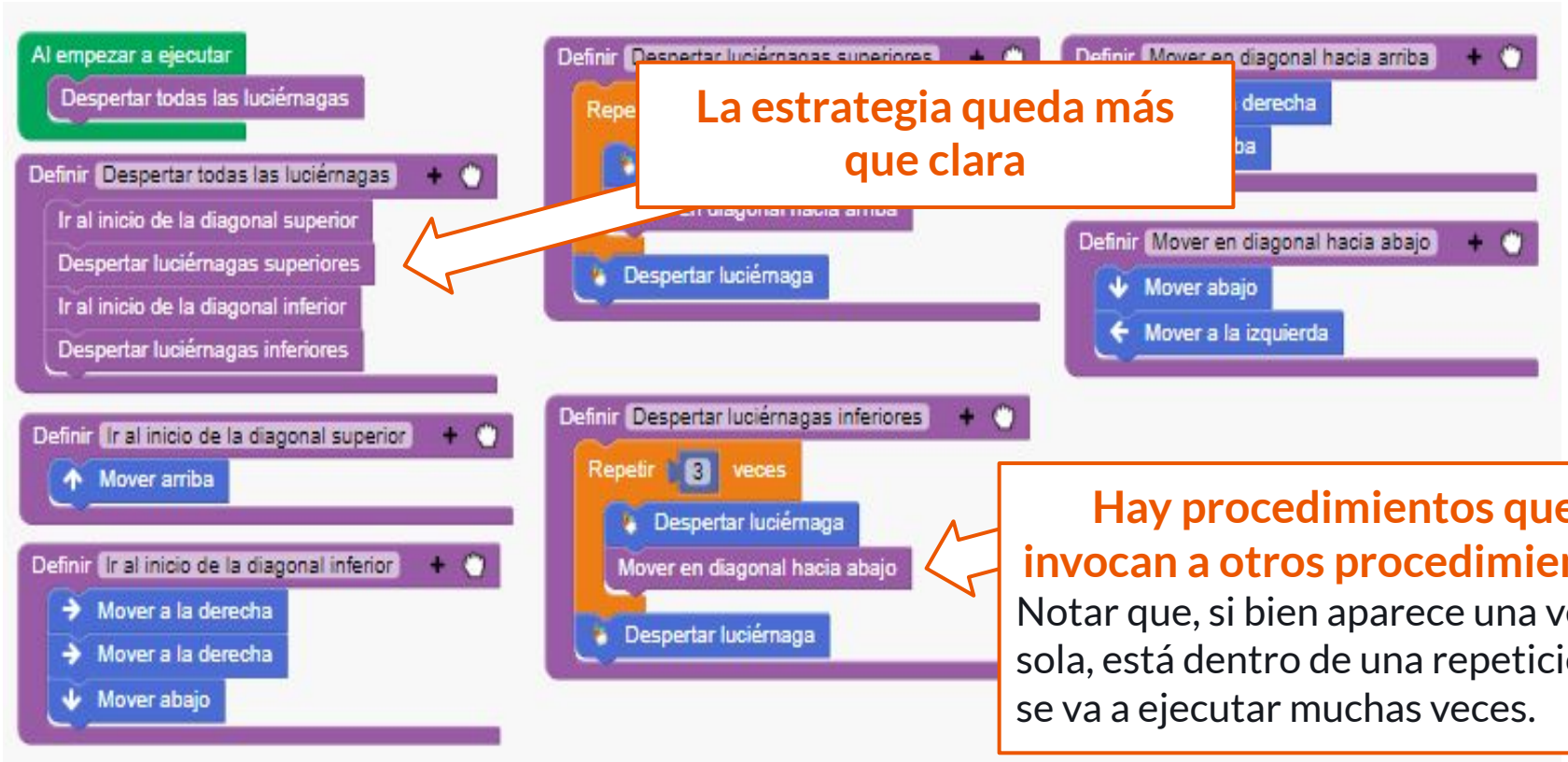
Las luciérnagas de arriba se despiertan mientras se mueve a Yvoty hacia arriba.

Las luciérnagas de abajo se despiertan mientras se mueve a Yvoty hacia abajo.

Ir de una diagonal de luciérnagas a otra implica moverse a la luciérnaga superior de la otra diagonal.



Yvoty y las luciérnagas: ¿Qué aprendimos?



La estrategia queda más que clara

Hay procedimientos que invocan a otros procedimientos

Notar que, si bien aparece una vez sola, está dentro de una repetición y se va a ejecutar muchas veces.

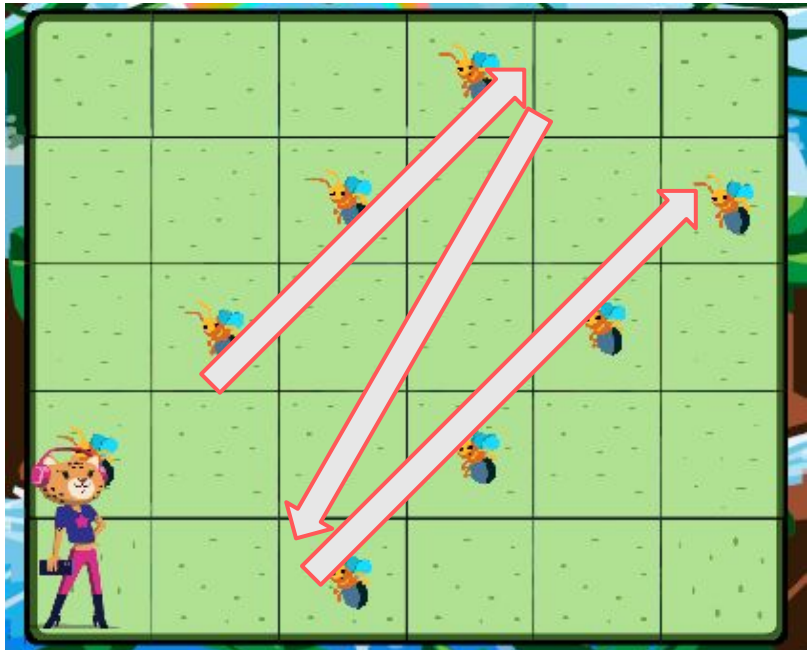
Yvoty y las luciérnagas: ¿Qué aprendimos?

Estrategia de subir y subir

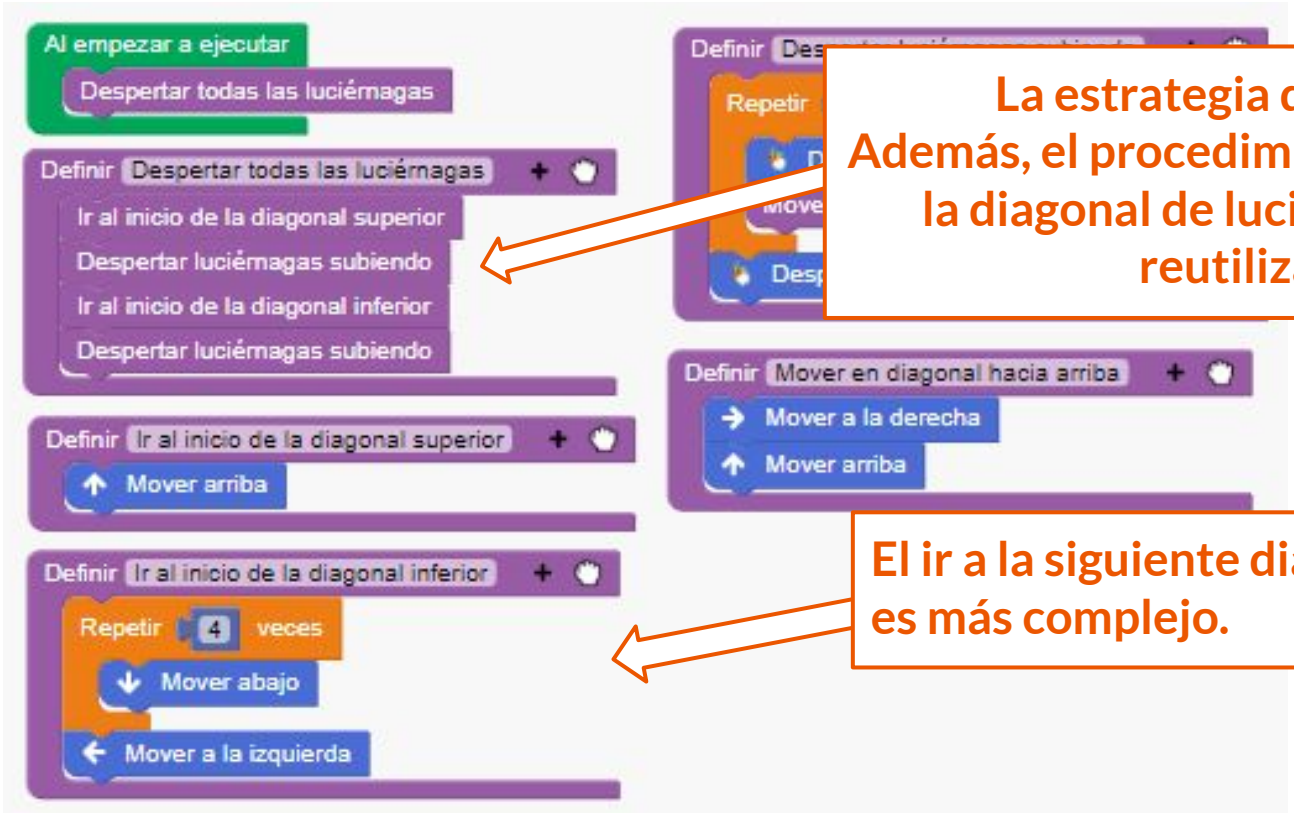
Las luciérnagas de arriba se despiertan mientras se mueve a Yvoty hacia arriba.

Las luciérnagas de abajo se despiertan igual que la superior.

Ir de una diagonal a otra implica ir hasta la luciérnaga más abajo de la diagonal.



Yvoty y las luciérnagas: ¿Qué aprendimos?



La estrategia queda clara.
Además, el procedimiento de despertar la diagonal de luciérnagas puede reutilizarse.

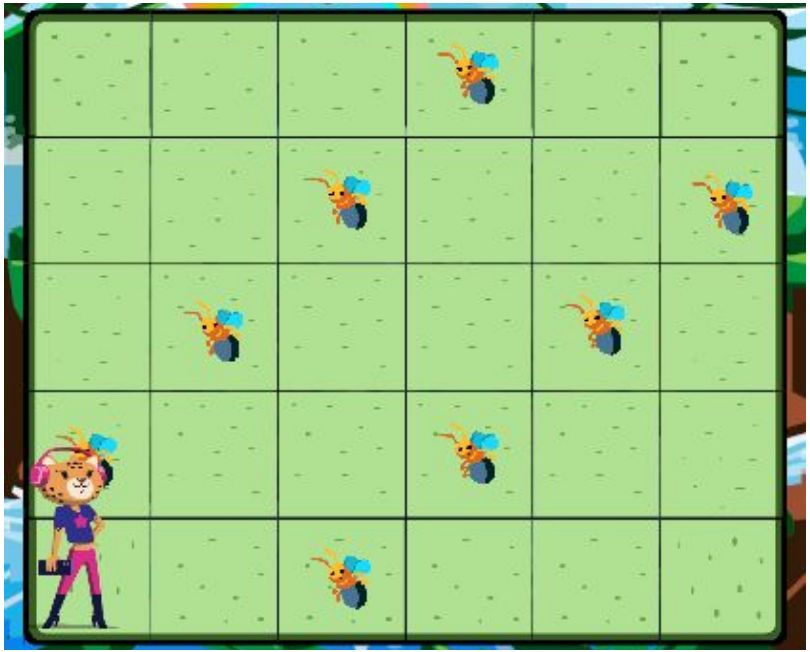
El ir a la siguiente diagonal es más complejo.

Yvoty y las luciérnagas: ¿Qué aprendimos?

Ambas son adecuadas, aunque la segunda estrategia es un poco más linda, porque se reutiliza mejor el código.

Existen muchas otras variantes, por ejemplo, empezando por la diagonal inferior.

Incluso podemos salir del pensamiento de diagonales, y pensar en resolver por filas o columnas.

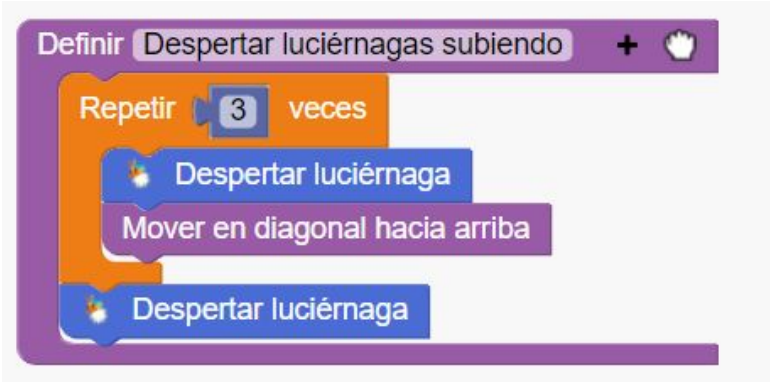


Yvoty y las luciérnagas: ¿Qué aprendimos?

Además, algunos de los procedimientos que debemos realizar nos dejan aprendizajes muy importantes.

Veamos el procedimiento de despertar luciérnagas subiendo.

¿Por qué no repetir 4 veces? Sí son 4 luciérnagas.

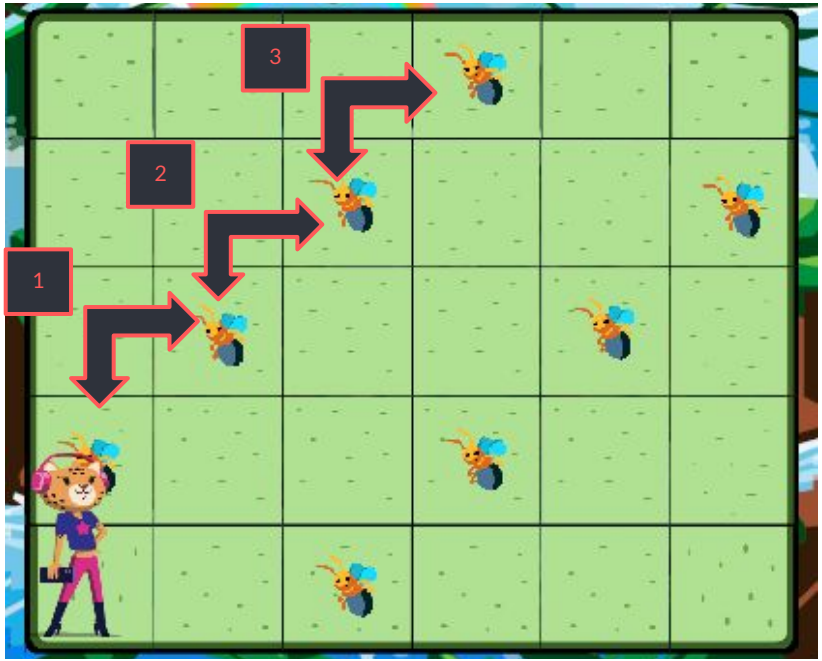


Yvoty y las luciérnagas: ¿Qué aprendimos?

El problema es que, si bien son 4 luciérnagas a despertar, no son 4 movimientos los que hay que realizar, sino 3.

Si intentamos movernos una cuarta vez, nos saldremos de los límites del escenario, e Yvoty se quejará de ello.

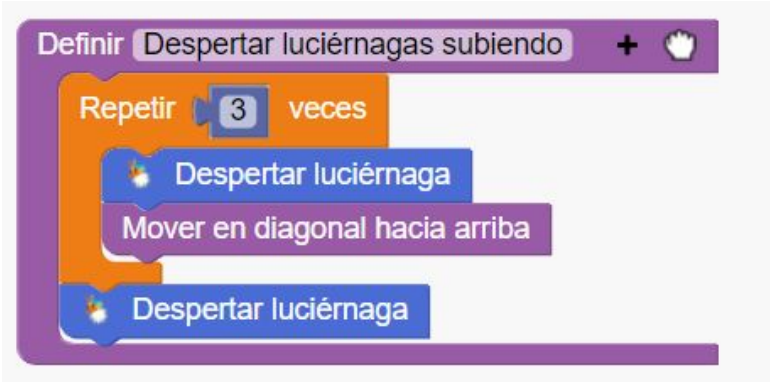
Solo debemos movernos 3 veces. El caso de la cuarta luciérnaga, es un caso especial, pues sólo queremos despertarla, pero no movernos.



Casos de borde

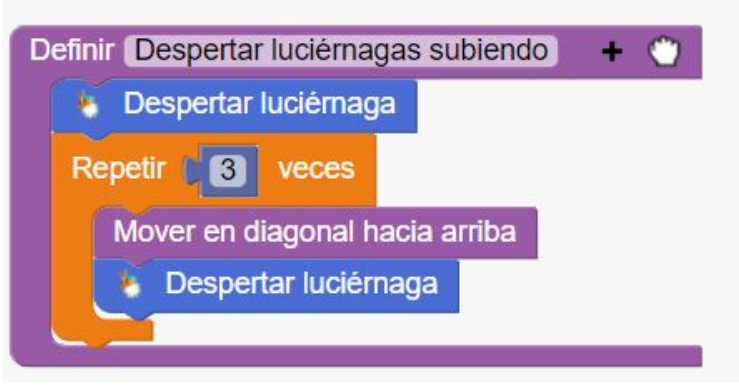
Llamamos a este tipo de situaciones **“casos de borde”**, donde la cantidad de movimientos no es igual a la cantidad de procesamientos (despertar la luciérnaga en este caso) y se deben contemplar en los bordes de la repetición.

Despertar la última luciérnaga es un caso de borde de este problema.



Casos de borde: también al principio

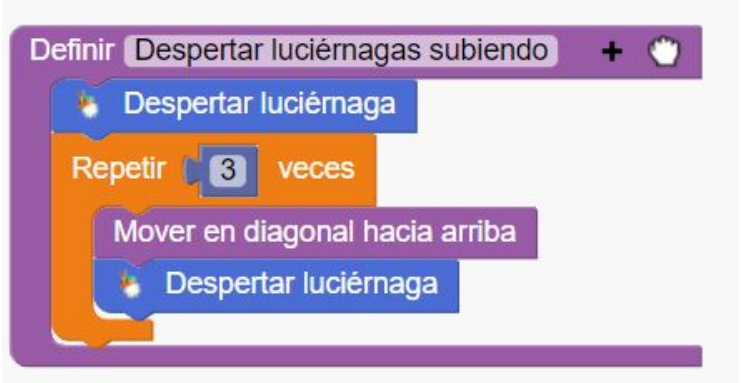
También podría ser la primera de las luciérnagas la que se considere de forma diferente, teniendo el caso de borde al comienzo.



¡La invocación también es un comando!

De la misma forma **“Mover en diagonal hacia arriba”** es un comando más, y por tanto puede estar invocado en cualquier cuerpo, ya sea el de un procedimiento, o el de una repetición dentro de un procedimiento, es indistinto.

Esto permite que haya procedimientos, que llaman a procedimientos, que llaman a procedimientos, que llaman a procedimientos... tantos niveles como sea conveniente para dividir el problema en partes simples.



Estrategia de solución

Cuando encaramos un ejercicio, primero debemos hacer dos cosas:

1. Comprender claramente la información o transformación esperada. Es decir, cuál es el estado inicial del programa y cuál el estado final esperado.
2. Plantear una estrategia que realice dicha transformación. Es decir, pensar en español, que grandes pasos deben realizarse para llegar del estado inicial al estado final

Estrategia de solución: Entender el problema

La mejor forma de entender el problema es seguir los siguientes pasos:

1. Comprender los estados tiene que ver con leer atentamente el enunciado, interpretando bien lo que está escrito, y analizar los ejemplos dados si los hubiera.
2. Entender que los ejemplos son eso, ejemplos, y que podría haber otros casos de los cuales no se hayan mostrado ejemplos, pero que estén expresados en el enunciado claramente.
3. Volver a leer el problema sí hay algo que no quedó claro.
4. VOLVER A LEER EL PROBLEMA
5. Consultar con el docente si aún me quedan dudas.
6. VOLVER A LEER EL PROBLEMA

Estrategia de solución: Plantear una estrategia

- La estrategia tiene que ver con lo que uno ve cuando analiza el problema. No todos vemos lo mismo y, por tanto, no todos vamos a plantear exactamente la misma estrategia.
- También tiene que ver con las herramientas y primitivas disponibles. Como vimos en “Reparadora de telescopios” hay veces que las primitivas limitan las estrategias posibles.
- Otras veces como en “Yvoty y las luciérnagas” hay infinitas estrategias posibles (no todas adecuadas para la materia por no respetar las buenas prácticas de la programación. Ej. una estrategia donde no se separe en subtareas).
- **La estrategia debemos pensarla en español, y a grandes rasgos, ANTES de empezar a programar.**

Estrategia de solución: Plantear una estrategia

Ej. para “Yvoty y las luciérnagas” podríamos pensar la siguiente estrategia:

- Debo llevar a Yvoty a la diagonal que tiene justo arriba.
- Debo despertar a todas las luciérnagas de dicha diagonal.
- Debo llevar a Yvoty a la parte inferior de la otra diagonal.
- Debo despertar a todas las luciérnagas de dicha diagonal.

Notar que la estrategia no está escrita en términos de las primitivas, sino en base a ideas generales de lo que se debe realizar.

Estrategia de solución: Plantear una estrategia

Ej. para “Yvoty y las luciérnagas” podríamos pensar la siguiente estrategia:

- Debo llevar a Yvoty a la diagonal que tiene justo arriba.
- Debo despertar a todas las luciérnagas de dicha diagonal.
- Debo llevar a Yvoty a la parte inferior de la otra diagonal.
- Debo despertar a todas las luciérnagas de dicha diagonal.

Notar que la estrategia planteada es la que llamamos “subir y subir”. Cada gran parte del problema terminará siendo una subtarea, la cual expresamos mediante un procedimiento.

Estrategia de solución: Plantear una estrategia

Ej. para “Yvoty y las luciérnagas” podríamos pensar la siguiente estrategia:

- Debo llevar a Yvoty a la diagonal que tiene justo arriba.
- Debo despertar a todas las luciérnagas de dicha diagonal.
- Debo llevar a Yvoty a la parte inferior de la otra diagonal.
- Debo despertar a todas las luciérnagas de dicha diagonal.

Los nombres de dichos procedimientos deberían reflejar claramente la idea que tuve al pensar la estrategia.

Estrategia de solución

¿Cómo resolvemos el procedimiento **“Despertar luciérnagas subiendo”**?

- La idea es que volvemos a realizar exactamente lo mismo que hicimos para el problema principal, solo que ahora lo hacemos limitándonos exclusivamente a una parte del problema, el de despertar una única diagonal de luciérnagas. De ahí el nombre “subtarea”.
- Es decir, para ese procedimiento, planteamos una estrategia, la expresamos con subtareas. Para esas subtareas, pensamos una estrategia, y la planteamos con subtareas. Y así hasta que los elementos sean tan simples que puedan resolverse con primitivas.

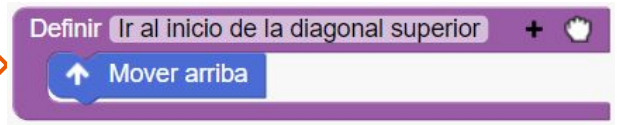
Estrategia de solución: Tampoco cosas innecesarias

Notar que es perfectamente válido mezclar en una subtarea procedimientos y comandos primitivos. Las partes del problema que pueden ser resueltas con primitivas deben hacerse así.

Solo tiene sentido plantear un procedimiento que solamente llame a una primitiva cuando el nombre que esta tiene (nombre que viene dado) no permite comunicar con claridad nuestra estrategia.

Pero si es suficientemente claro y solo lo renombramos utilizando sinónimos decimos que es un procedimiento innecesario.

Este procedimiento tiene sentido ya que el nombre que le damos permite comprender mejor la estrategia



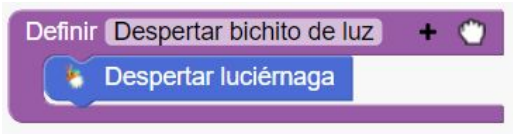
Estrategia de solución: Tampoco cosas innecesarias

Notar que es perfectamente válido mezclar en una subtarea procedimientos y comandos primitivos. Las partes del problema que pueden ser resueltas con primitivas deben hacerse así.

Solo tiene sentido plantear un procedimiento que solamente llame a una primitiva cuando el nombre que esta tiene (nombre que viene dado) no permite comunicar con claridad nuestra estrategia.

Pero si es suficientemente claro y solo lo renombramos utilizando sinónimos decimos que es un procedimiento innecesario.

Este procedimiento es innecesario. Se puede usar directamente la primitiva “Prender la luz”

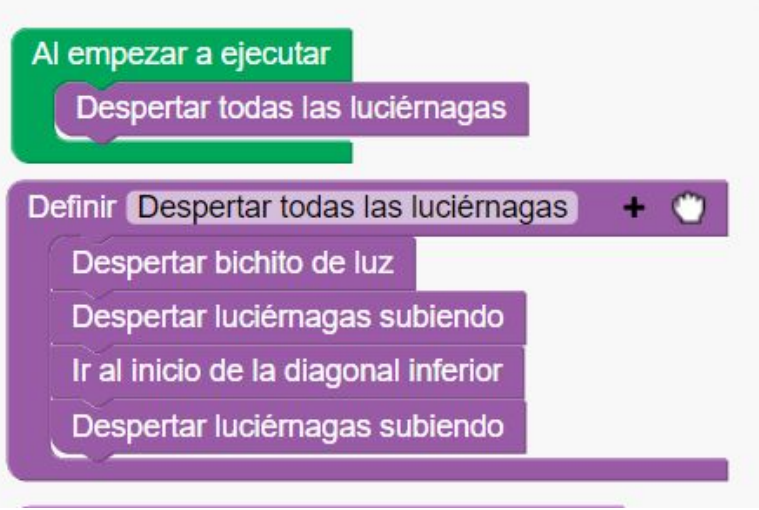


Nombrando la tarea principal

Si podemos darle nombre a una subtarea para expresar claramente su propósito, entonces también deberíamos poder hacerlo con el programa principal.

De hecho, no hay diferencia conceptual entre la idea de un “programa” (es decir, el cuerpo que ponemos en el punto de entrada) y de un procedimiento.

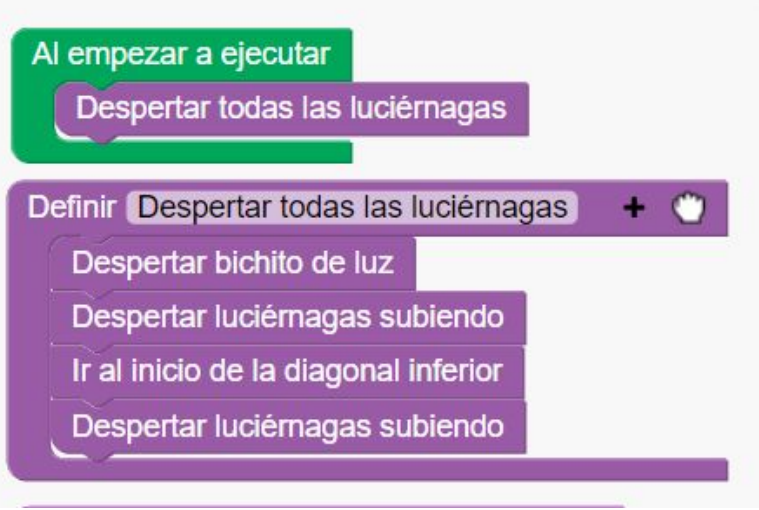
Aunque si hay una diferencia práctica. La máquina necesita saber por dónde comenzar a ejecutar, sí o sí.



Nombrando la tarea principal

Vamos a querer entonces que nuestro punto de entrada sea simplemente llamar a un único procedimiento, que resuelve la tarea principal en su totalidad.

El punto de entrada termina siendo entonces algo casi anecdótico, necesario desde el punto de vista de la máquina, pero irrelevante desde lo conceptual.



VAMOS A QUERER SIEMPRE CREAR UN PROCEDIMIENTO PRINCIPAL PARA SOLUCIONAR EL PROBLEMA.

ACTIVIDAD Y TAREA DE LA CLASE 6

Realizamos del libro del “Nivel intermedio”, los ejercicios “Campeone desordenade”, “Mañic y los planetas”, “Cargando los celus”, “Instalando juegos”, “El gran escape” y “Limpiando el humedal” de la sección “Autómatas, Comandos, Procedimientos y Repetición”.

EJERCITARIO 3: REPETICIONES SIMPLES Y ESTRATEGIA DE SOLUCIÓN