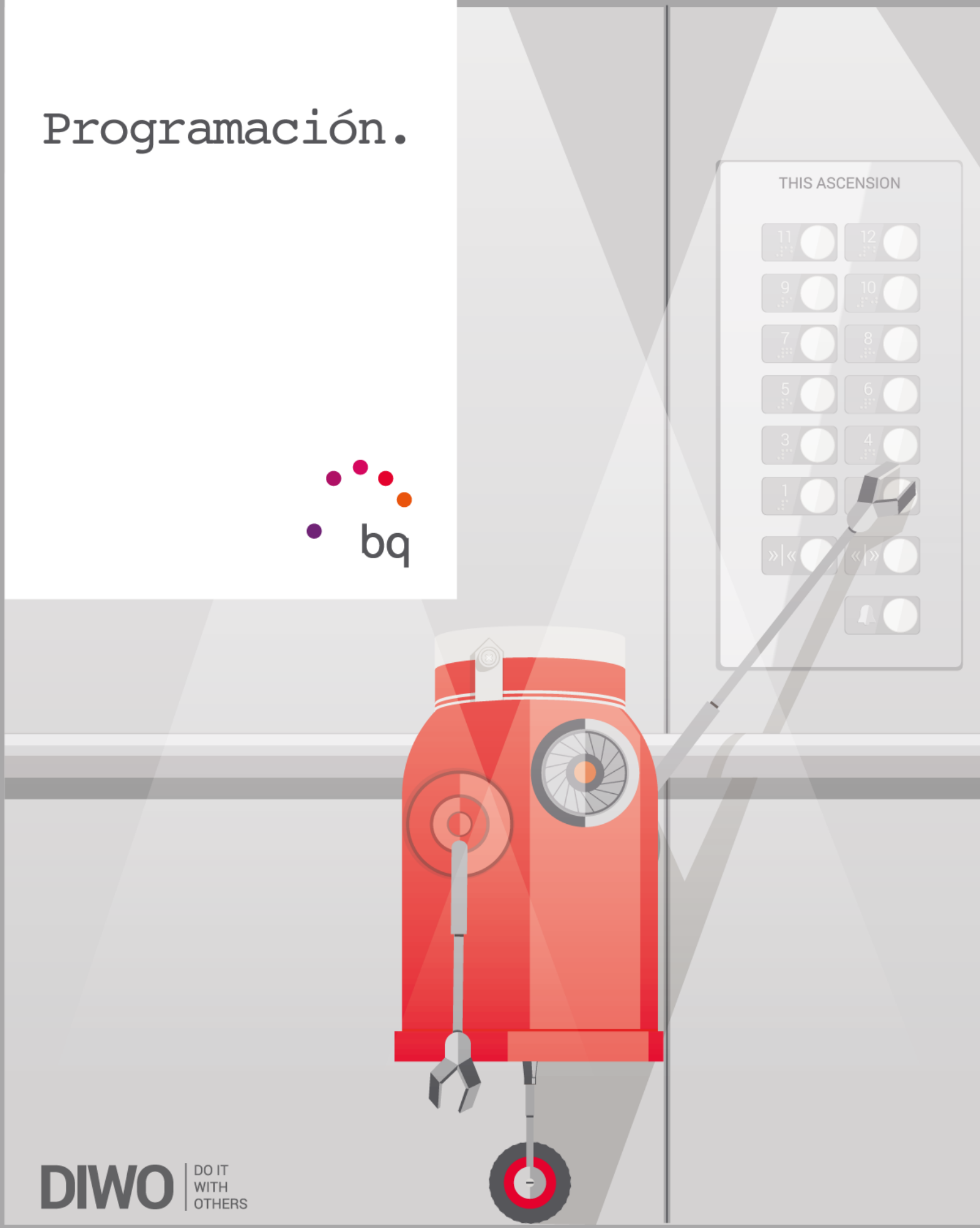
Programación.

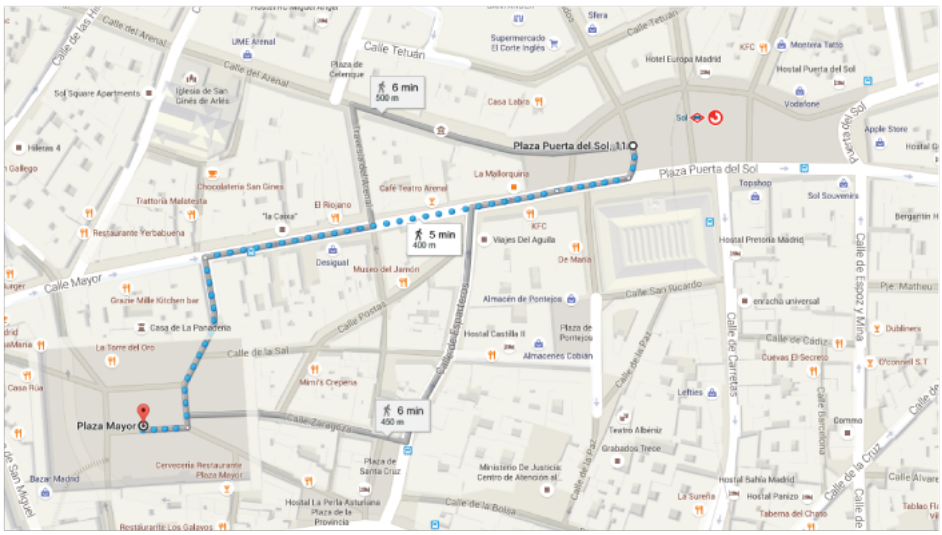


DIWO | DO IT WITH OTHERS

Algoritmos

Un algoritmo es una secuencia de pasos o instrucciones que hay que seguir para llegar al resultado que queremos obtener. El algoritmo, es decir, las instrucciones, han de ser claras y sencillas para que cualquier persona pueda seguirlas sin ningún problema.

Por ejemplo, las instrucciones que te dan los sistemas de navegación vía satélite (GNSS) para llegar desde un punto de partida a un destino específico. En la imagen se pueden ver las instrucciones para llegar desde la Plaza Mayor a la Puerta del Sol en la ciudad de Madrid.



Plaza Puerta del Sol | 28013 Madrid

Dirígete al sur hacia Plaza Puerta del Sol

Gira a la derecha hacia Plaza Puerta del Sol

Continúa por Calle Mayor.

Gira a la izquierda hacia Calle Felipe III

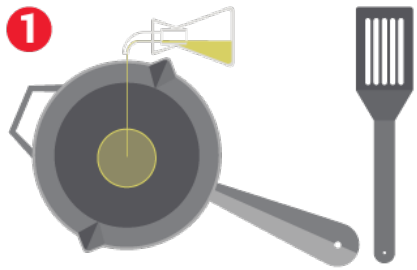
Gira a la derecha

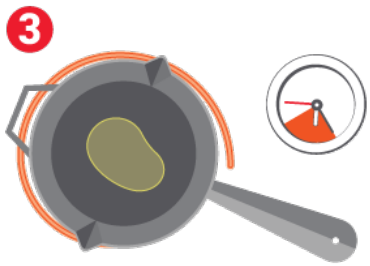
Plaza Mayor | Madrid

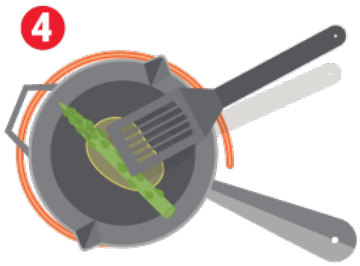
Fíjate que solo se indica una de las posibles rutas para llegar a la Puerta del Sol. Sin embargo, podríamos ir por otras rutas igual de válidas. A menudo existen varias soluciones o algoritmos para resolver un mismo problema.

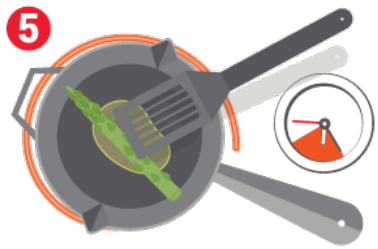
Otro ejemplo de algoritmo son las recetas de cocina. En una receta te pone paso por paso cómo cocinar el plato que quieras (con unos ingredientes específicos que te suelen aparecer en forma de lista). Incluso para hacer un espárrago frito, debemos seguir unas instrucciones:

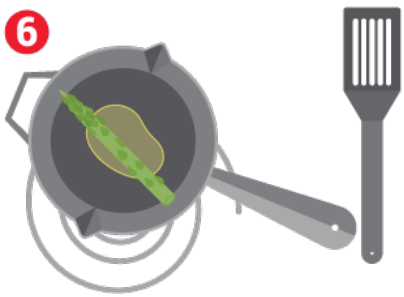
1. Poner aceite en la sartén
2. Poner la sartén al fuego
3. Esperar a que el aceite esté caliente
4. Echar el espárrago en la sartén
5. Cocinar hasta que esté listo
6. Apagar el fuego y retirar la sartén
7. Echar el espárrago en un plato

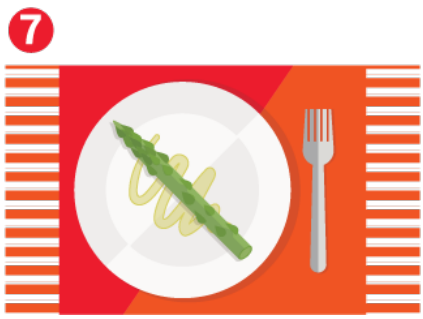






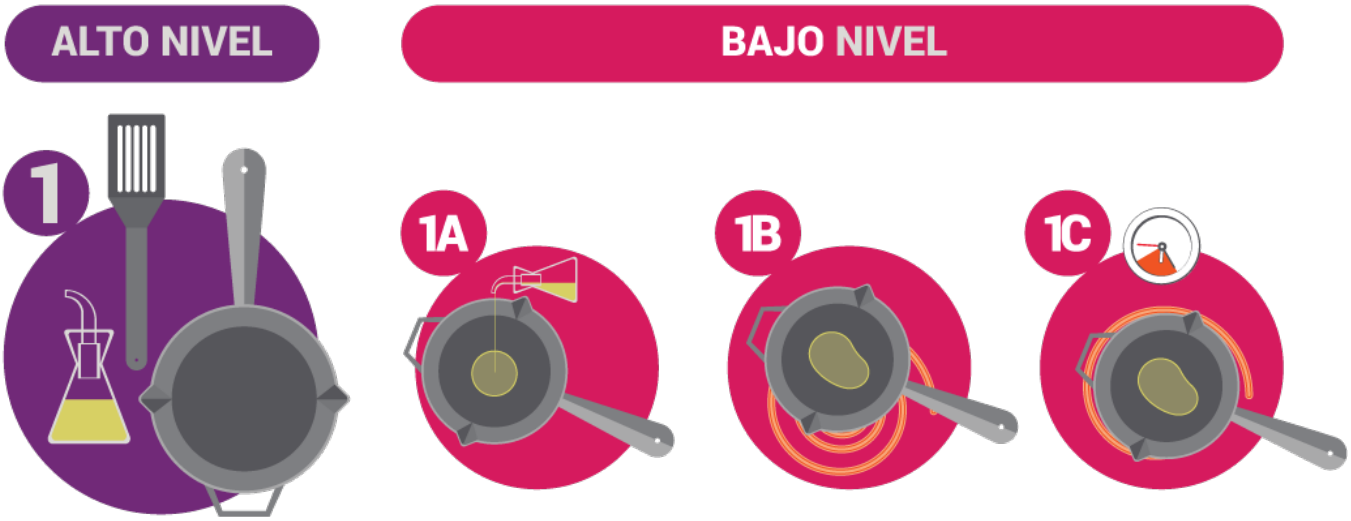




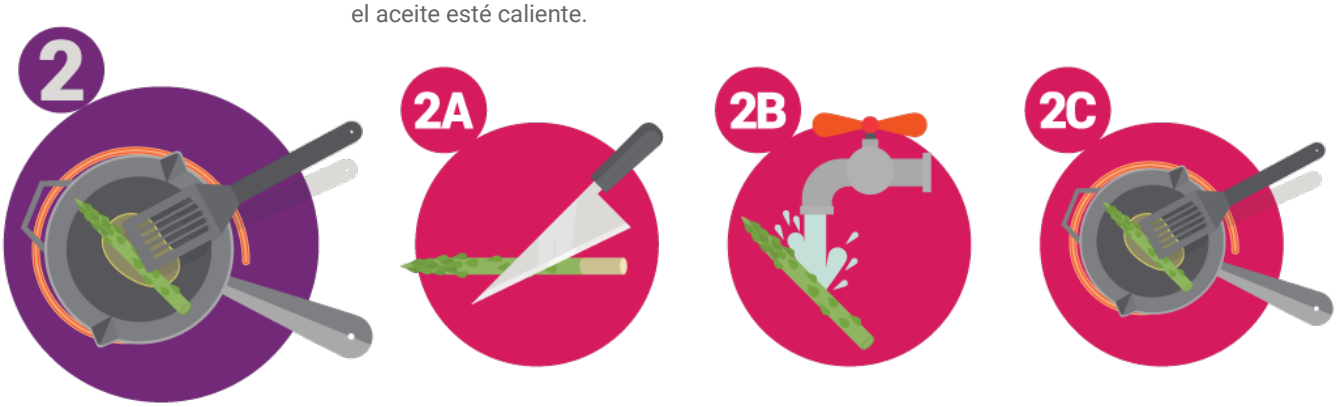


¿Y qué es eso de la programación a alto y bajo nivel?

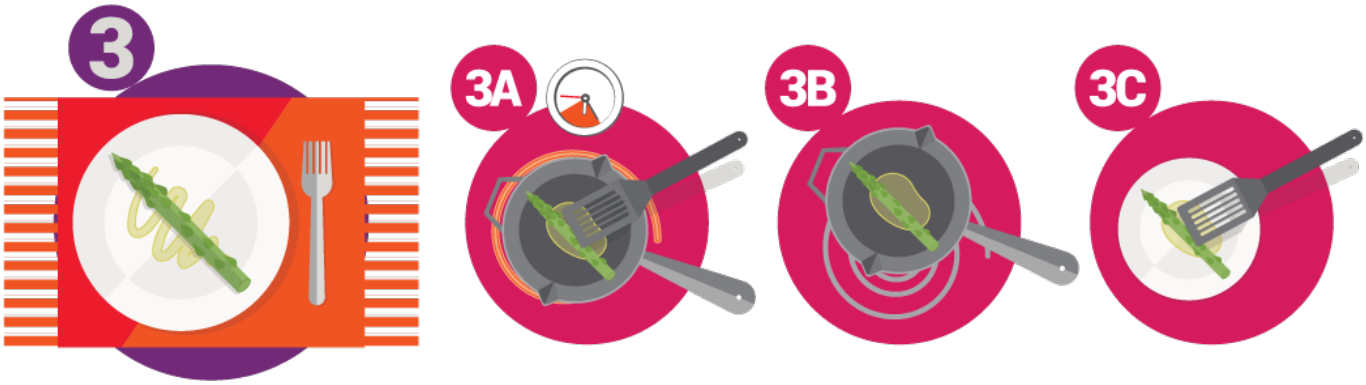
Los algoritmos pueden especificarse a diferentes niveles. Siguiendo con el ejemplo de la receta:



1. Preparar la sartén para freír:
1A. Poner aceite en la sartén | **1B.** Poner la sartén al fuego | **1C.** Esperar a que el aceite esté caliente.



2. Echar el espárrago a la sartén:
2A. Cortar la parte leñosa | **2B.** Lavar el espárrago | **2C.** Echar en la sartén.

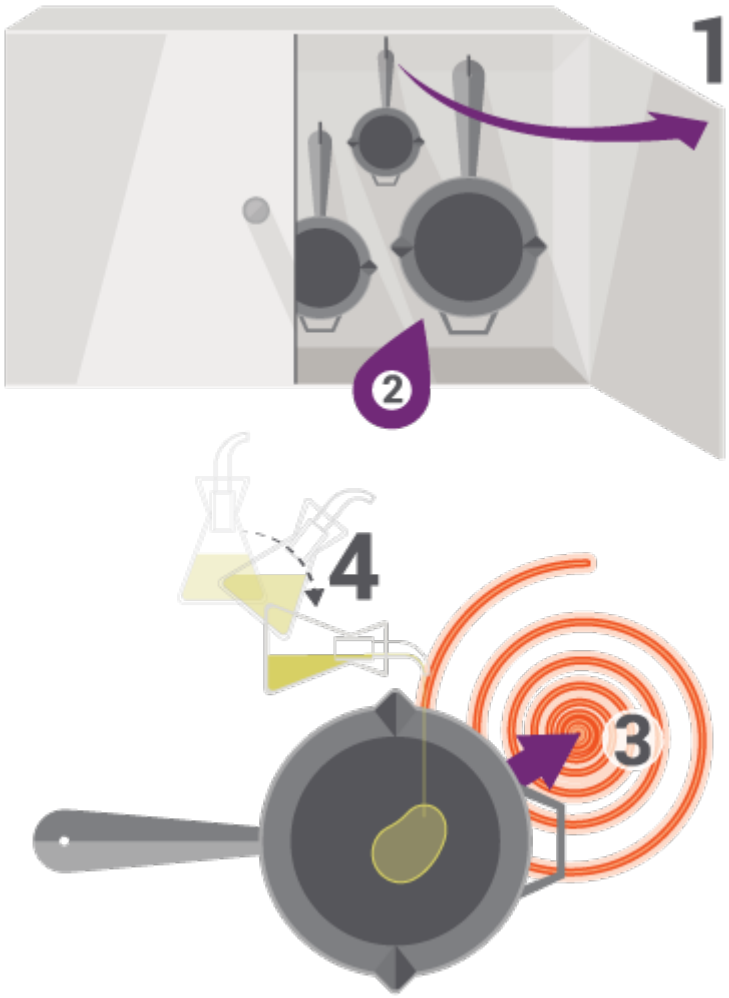


3. Cocinar y servir
3A. Cocinar hasta que esté listo | **3B.** Apagar el fuego y retirar la sartén | **3C.** Echar el espárrago en un plato.

Si conocemos un procedimiento que se repite mucho, éste se puede ejecutar a nivel interno y sólo hará falta indicarlo sin tener que especificar siempre todas sus acciones secundarias.

Si yo sé que para freír cualquier cosa necesito poner aceite en una sartén y dejarla al fuego hasta que se caliente el aceite, sólo me hace falta indicar “Preparar la sartén para freír”.

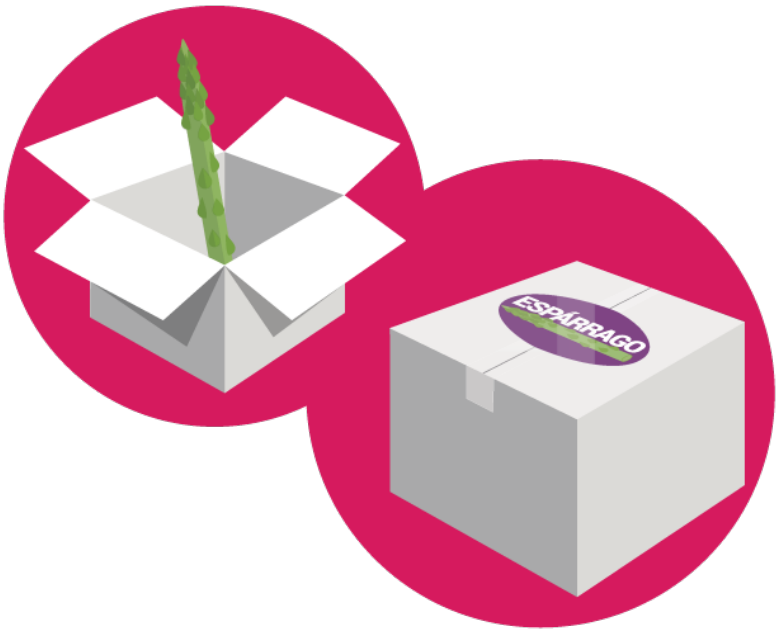
Actualmente existen muchos lenguajes de programación de alto nivel, haciendo más sencilla esta práctica. Esto es posible gracias a que existen programas que traducen órdenes más generales que se repiten a menudo a sus algoritmos completos, en los que se especifica cada pequeña acción. A su vez, hay otros programas a más bajo nivel que traducen instrucciones en otras más específicas.



Siguiendo de nuevo con el ejemplo del espárrago, la instrucción poner aceite en una sartén también se podría traducir en otro algoritmo más específico, a más bajo nivel:

1. Abrir el armario
2. Coger la sartén más adecuada
3. Ponerla sobre la cocina
4. Inclinar el aceite y echarlo en la sartén

Piensa que, en el fondo, toda la programación se reduce finalmente al lenguaje máquina, a código binario que se compone de instrucciones compuestas por 1 y 0.



Variables.

Una variable es como una “caja” donde guardamos un dato, como un espárrago, que podremos ver y recuperar más adelante durante el resto del programa.

Si no guardamos ese dato en una variable, luego no podremos utilizarlo, ya que el programa no lo recordará.

Además, en cualquier momento se puede cambiar ese valor guardado.

Declarar una variable.

Al declarar en nuestro programa una variable lo que hacemos es crear esa “caja” y **guardar por primera vez** un dato en ella. Puedes nombrar como quieras una variable. Esta cualidad te servirá para identificar en todo momento tu “caja”.

Podemos guardar datos de muchos tipos, como de tipo número o de tipo texto. Al declarar la variable defines de qué tipo es, por ejemplo: si lo primero que guardaste es un número, esa variable va a ser siempre para guardar números (*¡no mezcles tipos!*)

También podemos guardar el valor que nos devuelve un sensor.

Diferencia entre variable GLOBAL y variable LOCAL:

Es importante saber en qué momento del programa usamos y declaramos las variables, ya que el programa se ejecuta en orden (de arriba a abajo).

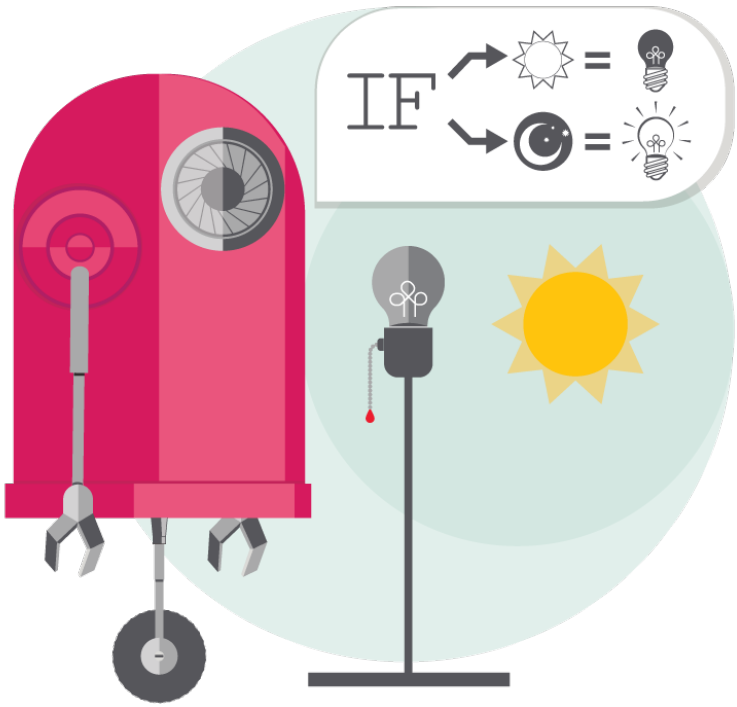
Se pueden programar variables LOCALES y GLOBALES.

Variables GLOBALES: se crean al inicio del programa, lo que las hace universales dentro de dicho programa. Pueden utilizarse en cualquier momento y lugar de éste.

Variables LOCALES: solo se pueden utilizar dentro del bloque de código donde han sido declaradas, pero esto es una ventaja, ya que así, cuando dejan de hacer falta, se borran solas y dejan espacio en la memoria del robot para otras cosas.

Al ejecutar un programa en orden estricto, necesitamos declarar las variables globales al principio del programa. También es fundamental tener en cuenta que si declaramos una variable local, ésta sólo se podrá utilizar dentro del bloque de código donde se declara.

Instrucción de control: Los condicionales



IF... (SI…)

Una sentencia condicional es una instrucción que se ejecuta o no en función del valor de una condición. Dependiendo de si la condición se da o no, se ejecutará un efecto u otro.

Es muy sencillo, por ejemplo: Si empieza a llover, abriré el paraguas. Si se hace de noche, encenderé mi linterna.

IF...ELSE (SI… DE LO CONTRARIO)

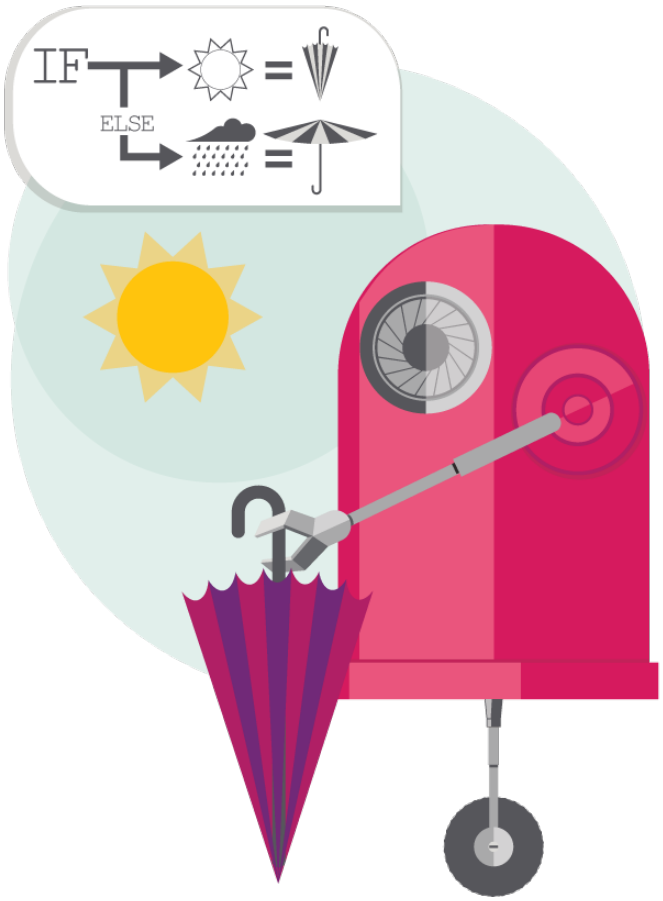
En muchos caso la sentencia IF se nos queda corta. Para los casos en los que una variable puede tomar dos valores es más adecuado utilizar un IF… ELSE.

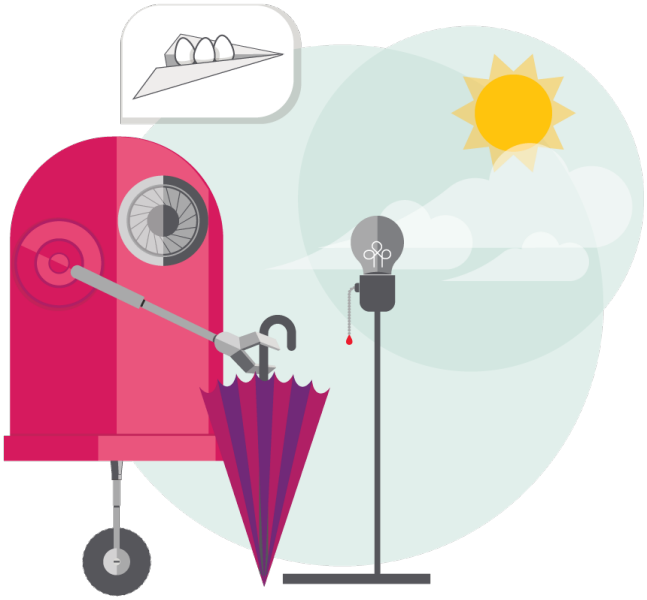
Por ejemplo, si nos encontramos ante una puerta, la puerta tendrá dos estados: abierta o cerrada. En caso de que esté abierta haremos una cosa (pasar) y, de lo contrario, en el caso de que esté cerrada haremos otra (llamar).

En los ejemplos de IF podéis pensar que también se podría hacer con un IF...ELSE. Efectivamente. Sin embargo, vamos a ver con más profundidad cada ejemplo:

Si empieza a llover, abriré el paraguas. Aquí partimos de que no está lloviendo, y sólo si empieza a llover ejecutaré la condición escrita: abrir el paraguas. En este caso el “de lo contrario” sería seguir sin abrir, que es lo mismo que el estado inicial por lo que no hace falta incluir esa segunda condición. Pero, ¡cuidado! si no incluimos la condición de cerrar el paraguas, seguiría abierto aunque deje de llover.

Si se hace de noche, encenderé mi linterna. Este caso es muy parecido al anterior, sólo si se hace de noche ejecutaré la condición: encender la linterna. El “de lo contrario” no nos haría falta, ya que seguiríamos en el estado inicial: no encenderla. Igual que el ejemplo anterior, si no incluimos la condición contraria seguirá encendida aunque salga el sol.





SWITCH...CASE

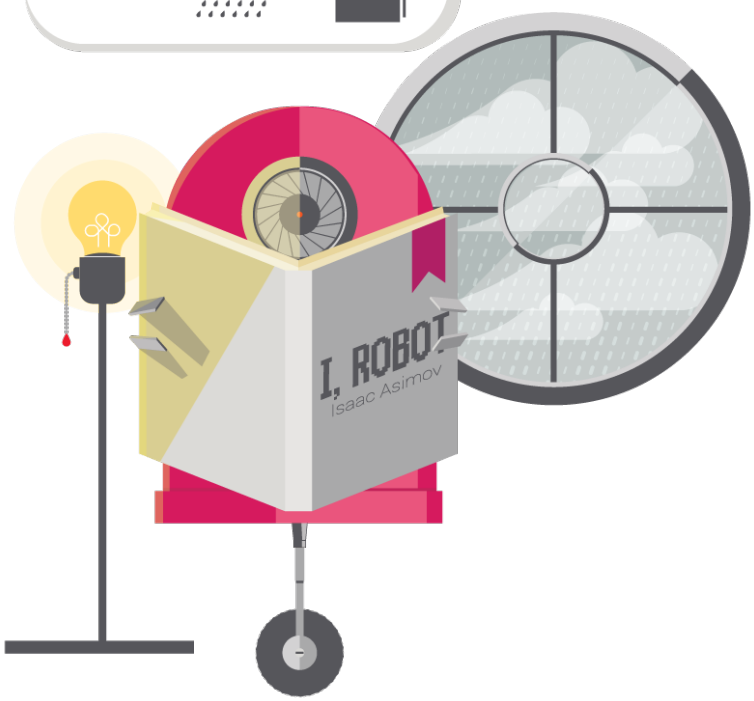
Otras veces una variable puede tomar más de dos valores. Para estos casos en los que una variable toma varios valores existe otra sentencia conocida como SWITCH...CASE.

Por ejemplo, mirando la siguiente imagen podemos ver que, según qué día de la semana sea, esta persona hace diferentes actividades. Para este caso, la variable es *día de la semana* y los valores que toma son 7: *lunes, martes, miércoles, jueves, viernes, sábado, domingo*.

```
lunes      fútbol
martes     inglés
miércoles  fútbol
jueves     inglés
viernes    hípica
sábado     tenis
domingo    golf
```

Instrucción de control: Los bucles

WHILE (mientras...)



Estos bucles se utilizan cuando queremos repetir la ejecución de unas acciones un número indefinido de veces, mientras que se cumpla una condición.

Por ejemplo, *mientras conduces, miras la carretera*.

Puedes pensar que esta acción también puede realizarse mediante un condicional (si conduzco, miro a la carretera). Sin embargo, si en tu algoritmo tienes órdenes tras la condición de mirar a la carretera, por ejemplo, llamar a un amigo, realizarás la acción de llamar a tu amigo y podrás tener un accidente.

Esto no ocurre con el bucle WHILE: mientras estés conduciendo, solo realizarás esas acciones y no continuarás con tu programa. Una vez dejes de conducir, llamarás a tu amigo por teléfono. Mucho mejor, ¿verdad?

FOR (contar…)

El bucle FOR se utiliza para repetir una o más instrucciones un determinado número de veces. Por tanto, utilizamos el bucle FOR solo cuando sabemos seguro el número de veces que queremos que se ejecute una acción.

Por ejemplo, si yo quiero abrocharme los 8 botones de mi camisa, tengo dos opciones:

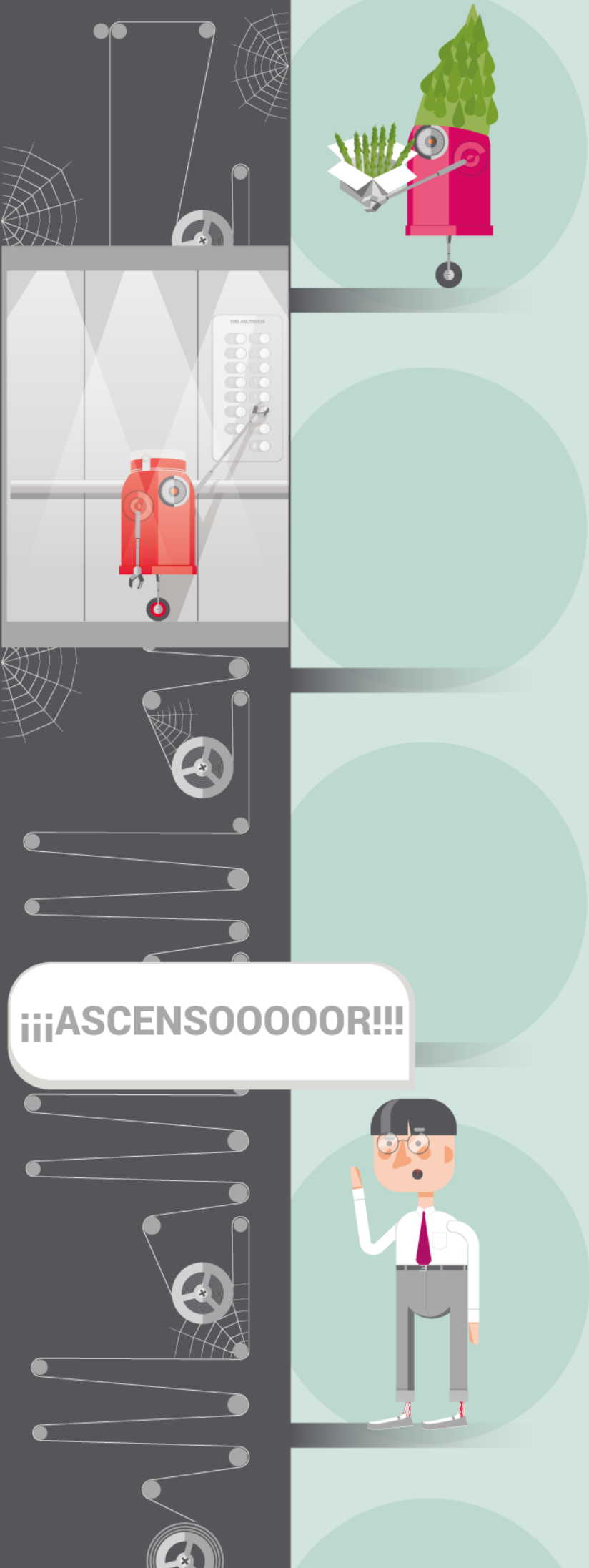
Crear el siguiente algoritmo:

1. Abrocha botón de la camisa
2. Abrocha botón de la camisa
3. Abrocha botón de la camisa
4. Abrocha botón de la camisa
5. Abrocha botón de la camisa
6. Abrocha botón de la camisa
7. Abrocha botón de la camisa
8. Abrocha botón de la camisa

Con esta opción abrocharía los 8 botones, pero no es la forma más eficiente de realizarlo.

Utilizar el bucle FOR:

Repite desde 1 hasta 8: Abrocha botón de la camisa.



Funciones

Cuando tengamos una tarea que se repite mucho, o cuando queramos organizar mejor nuestro proyecto, crearemos una función que realice dicha tarea cada vez que la llamemos. Por tanto, podríamos decir que una función es una tarea, una instrucción general que contiene varias instrucciones o pasos (un algoritmo específico), y que permite ejecutar dicha tarea con esa sola instrucción las veces que queramos.

Si retomamos el ejemplo anterior, diríamos que "freír un espárrago" sería una función, una tarea que se compone de las instrucciones antes descritas. Esta función o tarea la podremos reutilizar en todas las recetas que lleven espárrago frito: parrillada de verduras, risotto de espárragos, etc.

Traducido al lenguaje que utilizamos para programar: *parrillada de verduras* y *risotto de espárragos* serían programas concretos. *"Freír un espárrago"* sería una función, una tarea que repetimos a menudo y que si la escribimos de esta forma nos ahorra el tener que escribir todas las veces el conjunto de instrucciones (el algoritmo: *preparar la sartén, echar el espárrago, etc.*).

Otro ejemplo de función es "llamar al ascensor". Al solicitar esta tarea, estamos haciendo que se ejecuten una serie de instrucciones que nos permite lograr un resultado: que el ascensor nos transporte. Cuando se pone en marcha la función "llamar al ascensor", ejecuta un algoritmo con las siguientes instrucciones:

1. Localizar en qué planta/piso está el ascensor
2. Comparar si es mayor o menor que nuestra planta
3. Saber si tiene que subir o bajar para llegar
4. Poner en marcha los motores que permiten mover el ascensor
5. Comprobar en cada planta si ha llegado a su destino
6. Parar los motores cuando esté en nuestra planta
7. Abrir las puertas para que nos podamos montar

Qué contiene una función: PARÁMETROS

Los parámetros son los **datos que necesita una función** para que se ejecute correctamente.

Según el tipo de función, se necesitan determinados valores. Es importante distinguir un valor de una variable, las variables contienen valores. Las funciones trabajan con valores y no con variables, si modificamos un valor en una función, la variable no se verá alterada.

Por ejemplo, si para una función que expresa los minutos necesitamos el dato de una variable expresada en horas, dentro de nuestra función multiplicaremos el número de horas que hemos recibido como parámetro por 60 (horas * 60), pero la variable seguirá mostrando las horas.

Los parámetros se los damos a la función cuando la llamamos. Normalmente a continuación del nombre de la función.

¿Y cómo representamos todo esto? Diagramas de flujo

Un diagrama de flujo sirve para representar de forma gráfica un proceso, como por ejemplo un algoritmo. Para saber hacer diagramas de flujo tenemos que conocer primero los elementos básicos que podemos encontrar en uno de ellos.

Terminal

Representa el inicio y el final del diagrama.

Datos de entrada y salida

Representa los datos de entrada y los datos de salida.

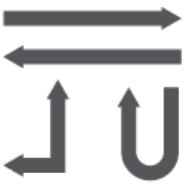
Proceso / Actividad

Representa cada proceso o acción concreta que se ejecuta a partir de unos datos. Estos datos pueden ser de entrada, de salida o como resultado de algún proceso anterior.

Decisión

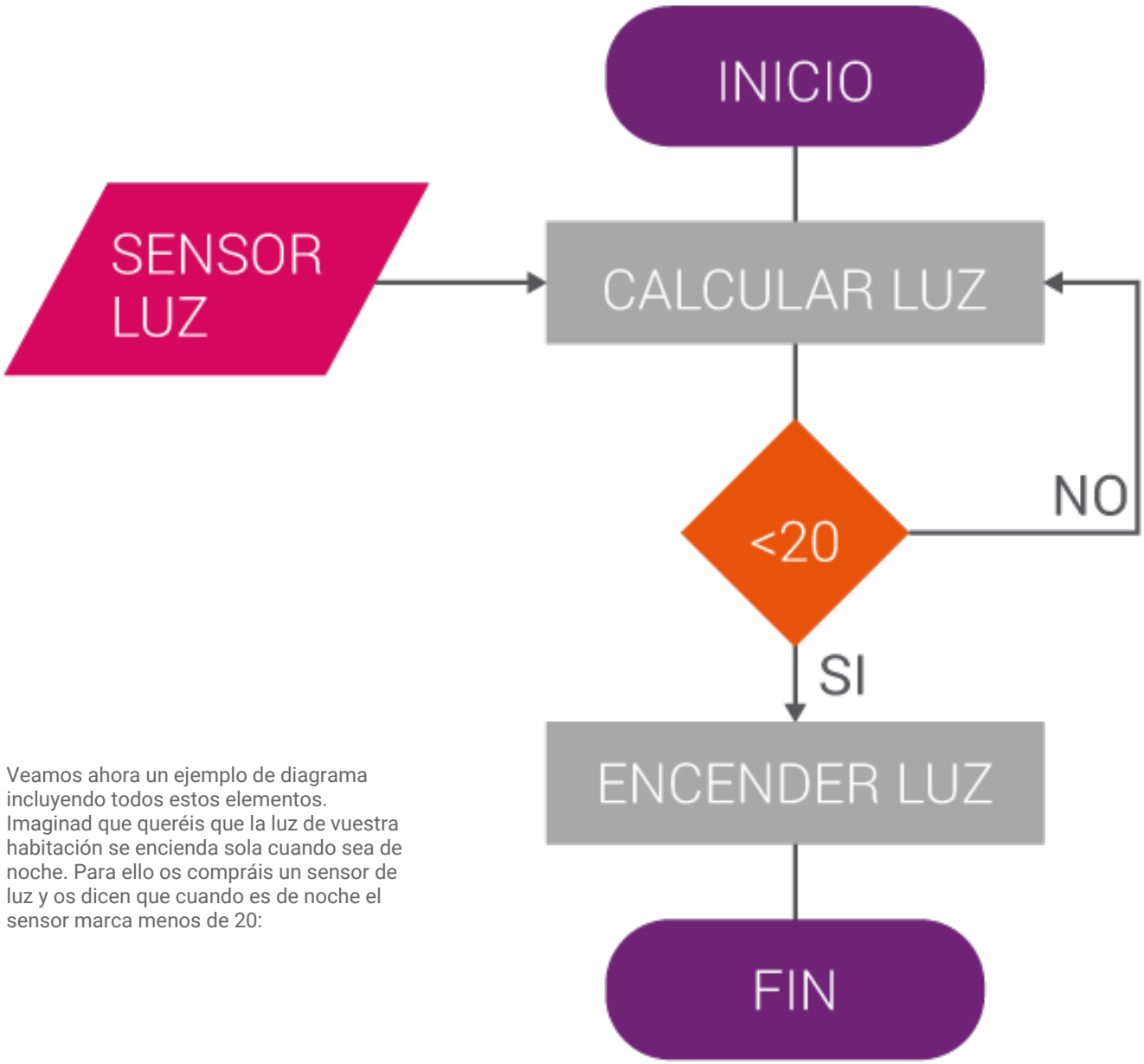
Representa una condición que puede resultar en dos valores: si/no, verdadero/falso.

En el caso de ser una condición numérica puede resultar en dos o tres valores: mayor que (número), menor que (número), igual a (número). En caso de que el igual se tome en conjunto con otra, resultarán dos posibles valores (por ejemplo, menor o igual que y mayor o igual que).



Líneas de flujo de información

Indican el sentido de los procesos, tanto de la información obtenida a partir del proceso anterior como de la información que será usada en algún proceso posterior



Veamos ahora un ejemplo de diagrama incluyendo todos estos elementos. Imaginad que queréis que la luz de vuestra habitación se encienda sola cuando sea de noche. Para ello os compráis un sensor de luz y os dicen que cuando es de noche el sensor marca menos de 20: