cREAgal

Viaje programado

Índice

cREAgal

3.1. Algoritmos y programación

Un **algoritmo** es como una receta. Es un conjunto de pasos que te dice exactamente qué hacer para resolver un problema.

Piensa en un algoritmo como las instrucciones paso a paso para hacer una tarea, como seguir un mapa para llegar a un lugar.

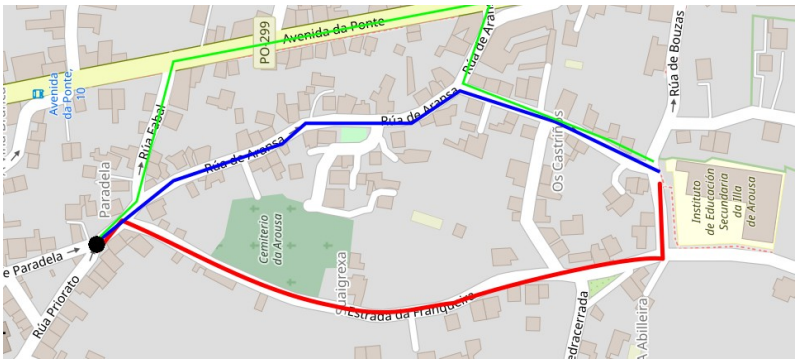
Programar es diseñar algoritmos para decirle a una máquina qué es lo que debe hacer en cada momento.

Por ejemplo, quieres ir en bici desde tu casa hasta el instituto. El hecho de planificar la ruta para poder llegar a tiempo a primera hora es un algoritmo.

Debes:

1. Comprobar la bicicleta. Revisa si las ruedas están bien infladas y que los frenos responden.
2. Planificar la ruta. Puedes usar una aplicación de mapas para encontrar la mejor ruta, priorizando las calles con carril bici u otros caminos seguros.
3. Revisar el tiempo. Consulta el pronóstico del tiempo para asegurarte de que no habrá condiciones adversas como lluvia intensa o vientos fuertes.
4. Preparar el equipo de seguridad. Ponte casco y asegúrate de llevar ropa visible o algo reflectante en la ropa.
5. Emprender el viaje. Sigue la ruta planificada, respeta las señales de tráfico y las normas de seguridad vial.
6. Evaluar la experiencia. Al llegar al instituto, reflexiona y piensa posibles mejoras a la hora de repetir el viaje.

Dado un mismo problema, puede ser resuelto por varios algoritmos diferentes. Por ejemplo, hay varias rutas posibles para ir desde casa al instituto.



Otro ejemplo de programación clásico es una receta de cocina.

Diseñar un algoritmo no es una tarea sencilla.

En programación, a veces, va a ser inevitable equivocarse, pero también va a ser imprescindible detectar los posibles errores y depurarlos.

La representación de algoritmos. Los diagramas de flujo.

Un algoritmo es una secuencia de pasos. Pero escribirlo en forma de lista no ayuda demasiado. Y si hubiera alguna forma para esquematizarlos de forma más gráfica.

El diagrama de flujo es la representación gráfica de un algoritmo. En un diagrama de flujo representamos simbólicamente los procesos a realizar y los unimos mediante flechas.

Los diagramas de flujo tienen una simbología estándar con la que se evita el uso de símbolos diferentes para los mismos procesos.

Los principales símbolos que se emplean en los diagramas de flujo son:

Inicio/Final

Se usa para indicar el inicio y el final del algoritmo. Del inicio solo puede salir una línea y al final solo puede llegar una línea.

Proceso general

Indica una acción o paso sencillo del algoritmo.

Entrada de datos

En general, indica la entrada de uno o más datos, por ejemplo, por el teclado, por un sensor, ...

Decisión

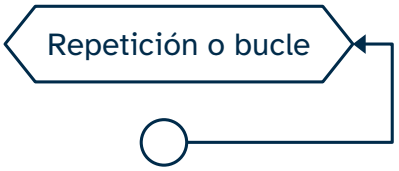
Indica una comparación de dos datos o una pregunta. Dependiendo del resultado lógico: Sí o No, Verdadero o Falso, ... se toma la decisión de seguir un camino u otro en el diagrama de flujo.

Salida de datos

En general, indica la salida de uno o más datos, por ejemplo, por pantalla, por PDF, por impresora, ...

Llamada a función o subprograma

A veces, para simplificar el código del algoritmo creamos pequeñas funciones o subprogramas que representan pequeños procesos.

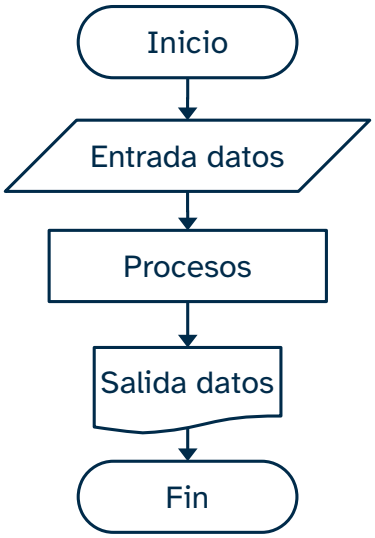


Repetición o bucle.

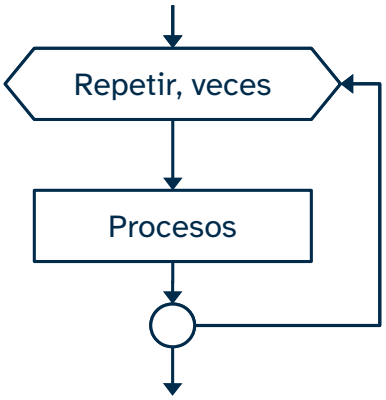
Indica que una instrucción o conjunto de instrucciones deben repetirse. Puede ser por siempre, por un número determinado de veces o hasta que se cumpla una condición.

Ejemplos de diagramas de flujo para las estructuras de programación básicas.

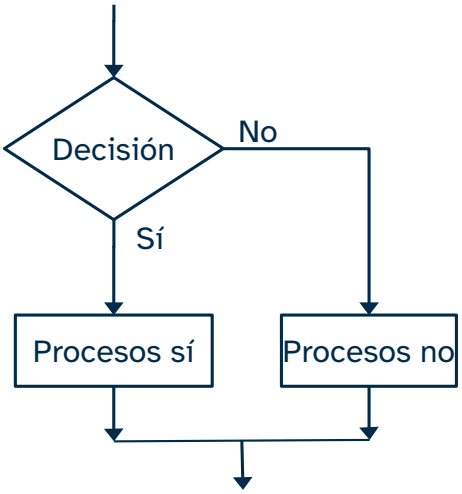
Las estructuras de programación se utilizan para crear algoritmos. Permiten organizar y controlar como se tiene que ejecutar el programa.



Estructura secuencial



Estructura repetitiva o bucle



Estructura condicional o decisión

La **secuencia** es la estructura de programación más básica.

Consiste en un conjunto de pasos o instrucciones que se van ejecutando una detrás de otra.

Los **bucles** consisten en la repetición de líneas de código. Instrucciones que se pueden repetir por siempre, un determinado número de veces o hasta que se cumpla una condición.

Los **condicionales** permiten a los programas tomar decisiones. Dependiendo de si una condición se cumple o no el programa ejecutará un conjunto u otro de instrucciones.

3.2. Lenguajes de programación. Scratch.

Una vez que tenemos diseñado un algoritmo tenemos que programar.

Programar va a ser decirle al sistema informático que debe hacer, cómo debe actuar para seguir las instrucciones del algoritmo que has diseñado.

Pero hay un pequeño problema, el sistema informático y tú no habláis el mismo lenguaje.

¿Qué lenguaje habla un sistema informático?

Como sabes los sistemas informáticos como tu ordenador, teléfono móvil o *tablet* funcionan con miles de pequeños circuitos eléctricos en su interior. Esos circuitos solo entienden dos situaciones: apagada o encendida, 0 o 1, lo que se conoce como código binario.

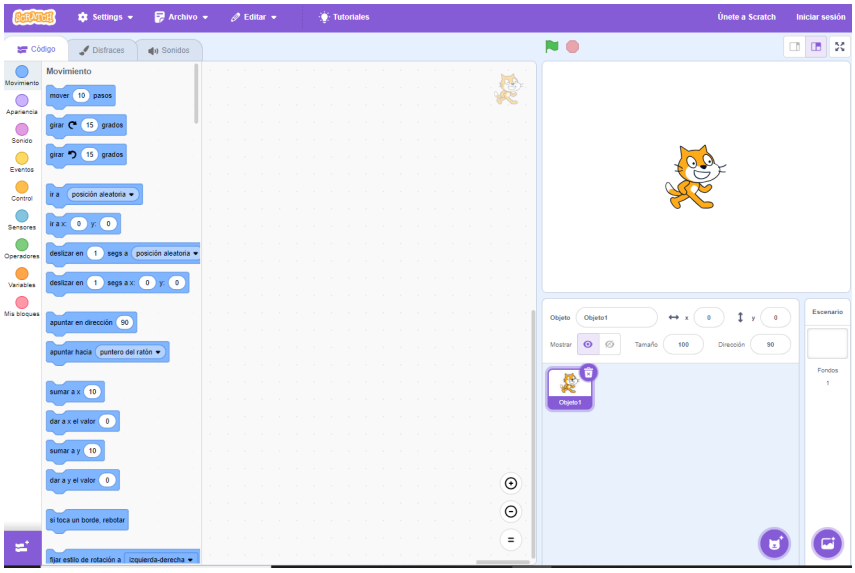
Pero, entre tú y yo, hablando con sinceridad, dar las instrucciones en forma de 0 y 1 es imposible para un ser humano. El lenguaje formado por 0 y 1 es el llamado **lenguaje máquina.** Este lenguaje se aleja demasiado de nuestra forma de pensamiento y comunicación.

La forma de dar instrucciones a los sistemas informáticos ha evolucionado para parecerse más al lenguaje humano y hacer que programar sea más sencillo.

Cuanto más se parece un lenguaje de programación a la forma en la que las personas pensamos y nos comunicamos decimos que los lenguajes son de más alto nivel.

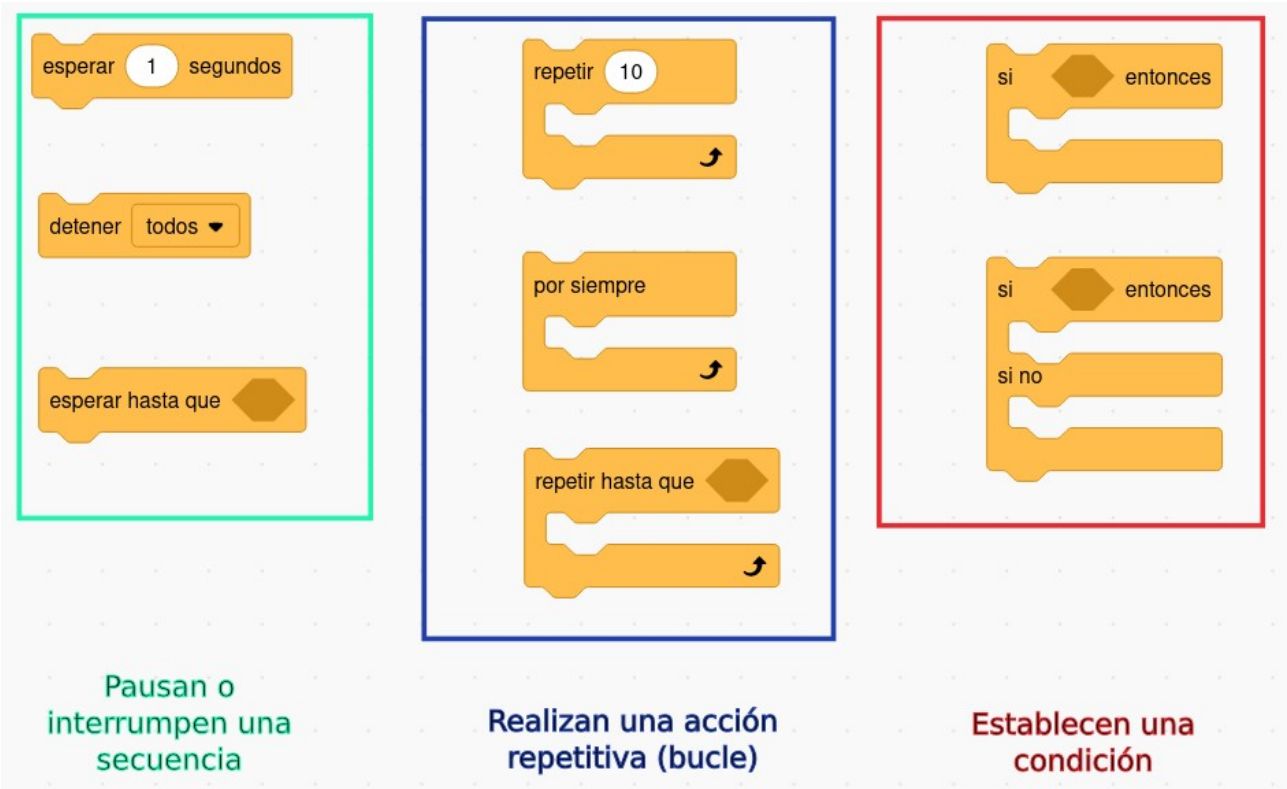
Para crear el videojuego necesitaréis un lenguaje de programación de alto nivel. El lenguaje que emplearemos será la plataforma de programación **Scratch**.

Scratch fue pionero en desarrollar una forma de programación especial y muy sencilla. Funciona mediante bloques que encajan como si fueran las piezas de un juego de construcción.

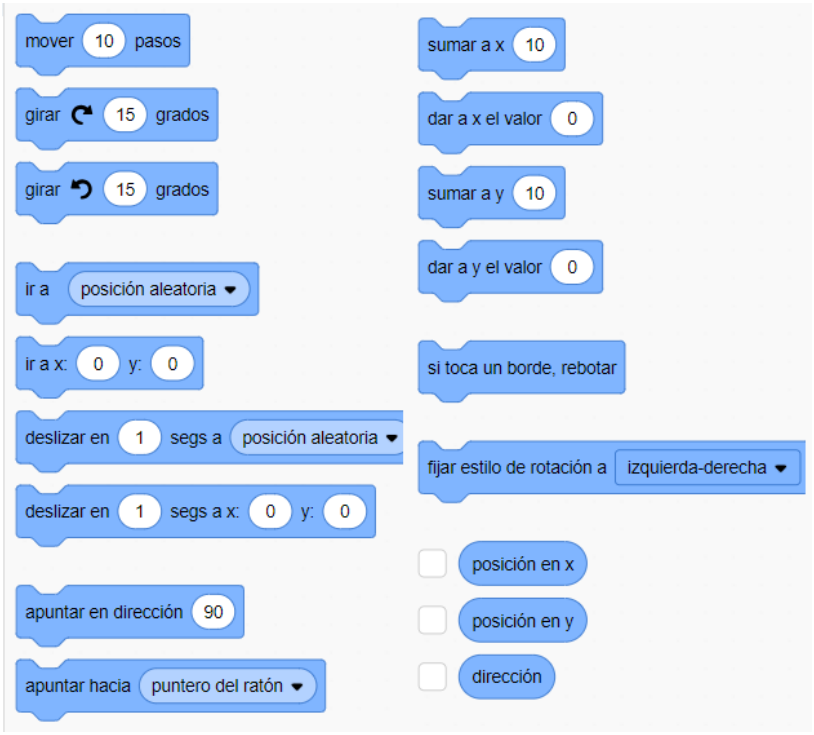


3.3. Entrena con bloques

Control y eventos

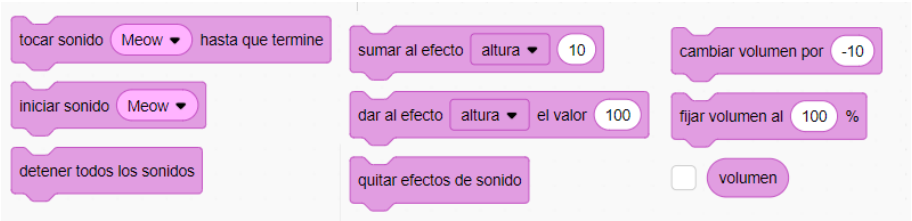

esperar 1 segundos
detener todos
esperar hasta que
repetir 10
por siempre
repetir hasta que
si entonces
si entonces
si no
Pausan o interrumpen una secuencia
Realizan una acción repetitiva (bucle)
Establecen una condición

Movimiento

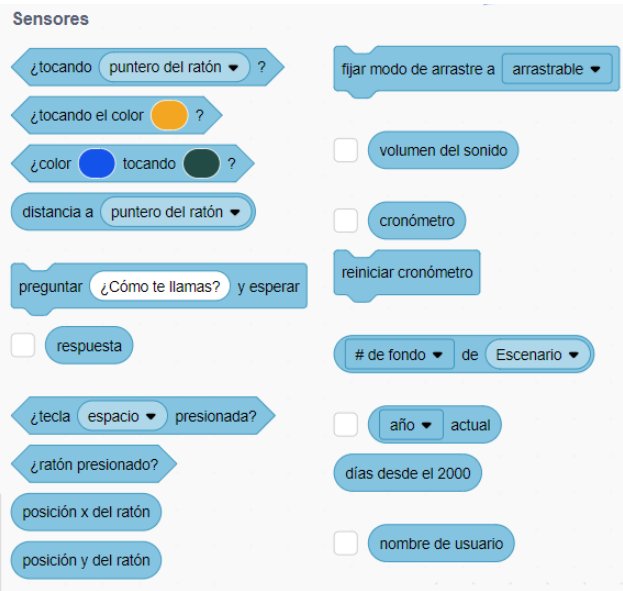

mover 10 pasos
girar 15 grados
girar 15 grados
ir a posición aleatoria
ir a x: 0 y: 0
deslizar en 1 segs a posición aleatoria
deslizar en 1 segs a x: 0 y: 0
apuntar en dirección 90
apuntar hacia puntero del ratón
sumar a x 10
dar a x el valor 0
sumar a y 10
dar a y el valor 0
si toca un borde, rebotar
fijar estilo de rotación a izquierda-derecha
posición en x
posición en y
dirección

Apariencia

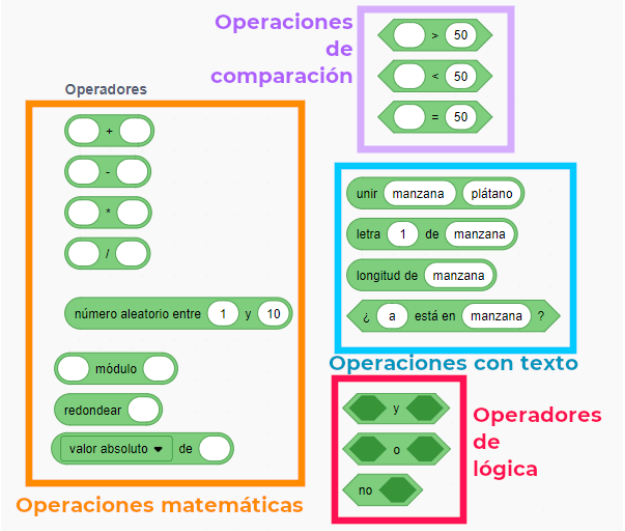
Sonido



Sensores



Operadores



Variables

Podemos pensar en una variable como en una etiqueta en la que tenemos un nombre y que está asociada a un valor.

En un videojuego, por ejemplo, al iniciar coges un valor 0 y le pones una etiqueta con el nombre puntos. A medida que avanza el juego ese 0 va cambiando su valor, mirando su etiqueta nunca perderás de vista lo que esa cifra significa.

Otras variables podrían ser las vidas, la energía, … a cualquiera de ellas les puedes asociar una etiqueta representativa de su significado.

En Scratch ya hay creadas algunas variables, de hecho, es posible que las tengas utilizado en algún programa.

Podemos llamarlas variables internas de Scratch o **bloques de reporte**.

Estos bloques "de reporte" son etiquetas que están asociadas a una información o valor concreto y nos reportan o dicen su valor en cada momento.

A diferencia de un bloque normal, de los que se apilan unos encima de otros, los bloques de reporte van dentro de otro bloque.

Para trabajar con variables creadas por ti hay una serie de **pasos que debes seguir:**

1. Crear la variable.
2. Inicializar la variable.
3. Programar el comportamiento de la variable.

3.4 Consejos para tus programas

- Da nombres significativos
- Evita el código muerto
- Incluye condiciones iniciales
- Crea hilos para las acciones simultáneas
- Añade comentarios
- Añade un final al juego

4.3. Prueba y depura

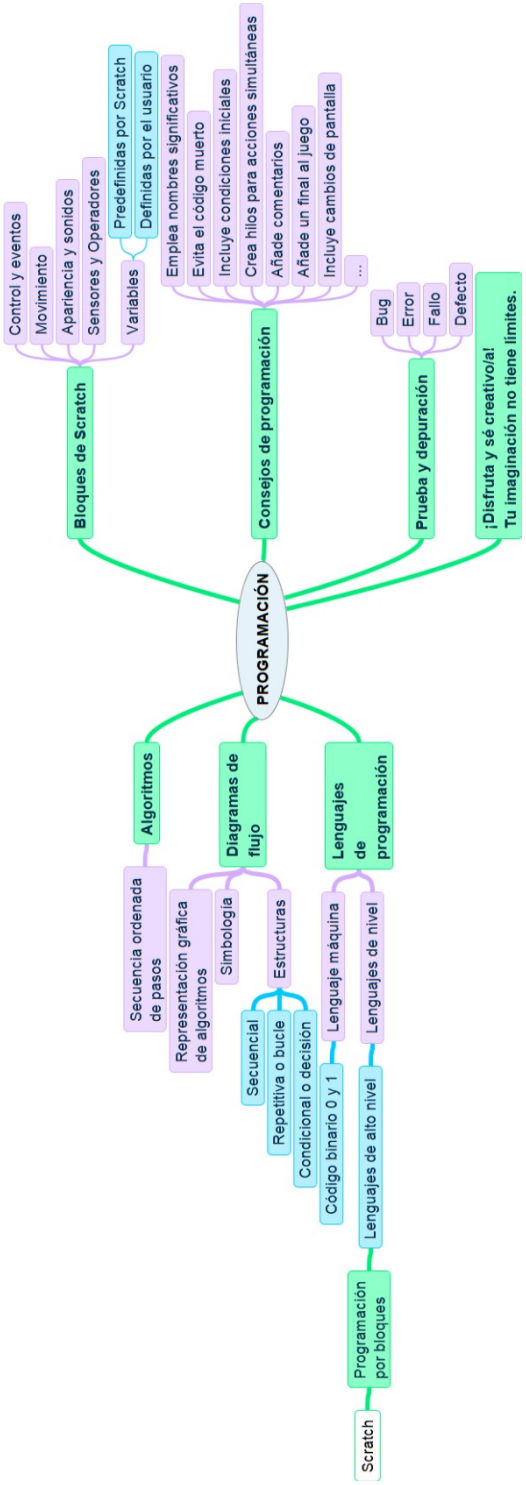
Todos los programas necesitan prueba y depuración para eliminar los errores.

Depurar es encontrar y corregir errores o problemas en un programa para mejorar su funcionamiento.

Algunos términos que son similares, aunque presentan algunas diferencias. Normalmente a todos ellos se les suele llamar, de forma general, BUG

- ***Error.-*** Es una equivocación del programador.
    - Por ejemplo, una instrucción mal escrita.
- ***Defecto.-*** Es la diferencia entre el valor que esperaba conseguir y lo que realmente se ha obtenido.
    - Por ejemplo, has escrito > en lugar de >=, lo que hará que, en algún momento, el programa se comporte de modo no esperado.
- ***Fallo.-*** Inconsistencia o problema que se detecta en la fase de pruebas.
    - De repente, sale un mensaje de fallo, o el programa deja de funcionar.
- ***Bug.-*** Suele englobar a todos los anteriores.

Error --> Defecto --> Fallo --> Bug



Esquema de la unidad