Introducción a Scratch

Scratch es un entorno de programación diseñado para enseñar a niños y jóvenes los fundamentos de la programación de ordenadores. Ha sido desarrollado por el Laboratorio de Medios del Instituto Tecnológico de Massachusetts (MIT), en Boston (EUA).
Emplearemos la versión 2.0, que funciona en equipos Windows, Linux y Mac. Se usa online, accediendo a la web scratch.mit.edu con cualquier navegador que tenga instalado Flash Player.

Logotipo y mascota de Scratch.

Scratch es útil para crear animaciones, cuentos interactivos y juegos. Nosotros lo usaremos para programar videojuegos sencillos. En las siguientes miniunidades haremos un juego de frontón, un juego de obstáculos y un juego de asteroides. En esta miniunidad estudiaremos los siguientes puntos:

1. ¿En qué consiste programar un ordenador?
2. Lenguajes de programación.
3. Algoritmos.
4. Estructuras básicas de programación.

Ayuda de Scratch: si quieres profundizar en el estudio de Scratch, te recomendamos que consultes el apartado Ayuda de la web de Scratch: scratch.mit.edu/help. Encontrarás una guía de inicio, tarjetas con ejercicios sencillos y videotutoriales.

1. ¿En qué consiste programar un ordenador?

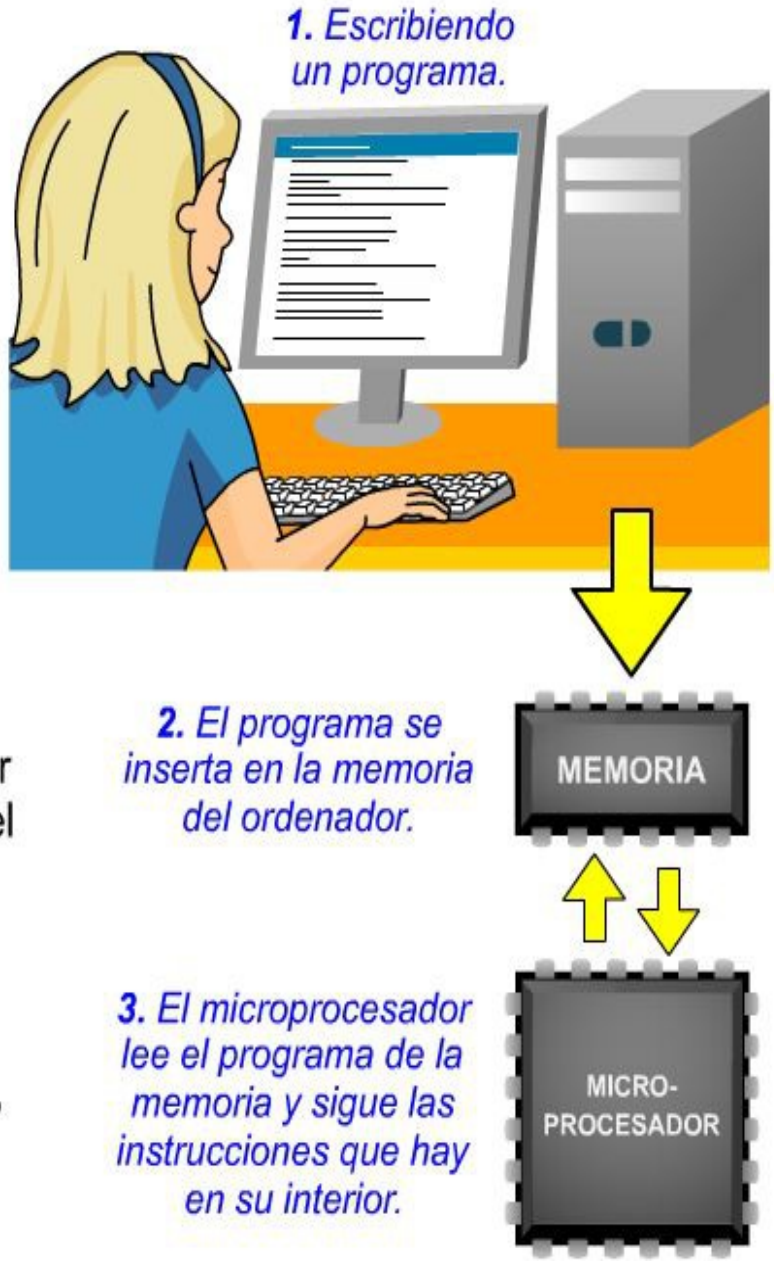
Un programa es un conjunto de instrucciones, dispuestas ordenadamente, que describen cómo debe trabajar un ordenador.

Llamamos programar a la tarea de diseñar y escribir las instrucciones que debe seguir un ordenador.

Un programa se ejecuta cuando se inserta en la memoria de un ordenador (un chip que almacena información) y el microprocesador (el chip más importante del ordenador, por analogía se dice que hace la función de "cerebro") lo lee y comienza a seguir las instrucciones que hay en su interior.

Los programas están formados por 3 bloques de instrucciones: la entrada, el proceso y la salida.

1. **Bloque de entrada:** son las instrucciones que indican al ordenador qué datos se deben captar de los periféricos de entrada (el ratón, el teclado, una pantalla táctil, etc.).
2. **Bloque de proceso:** son las instrucciones que describen qué se debe hacer con los datos que se han captado.
3. **Bloque de salida:** son las instrucciones que describen cómo se debe presentar el resultado en los periféricos de salida (la pantalla, los altavoces, la impresora, etc.).

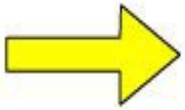


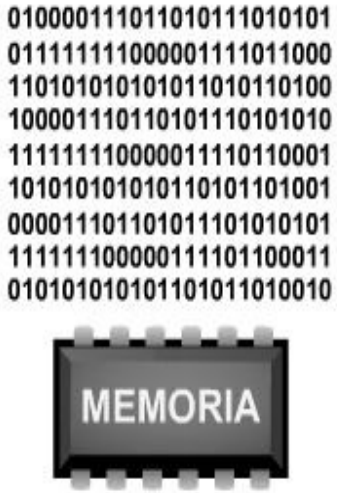
2. Lenguajes de programación. Qué son

Los ordenadores sólo entienden las instrucciones si se las damos como una secuencia determinada de señales eléctricas que entran por los terminales del microprocesador. Estas señales se almacenan en la memoria formando un conjunto de 0 (no entra corriente) y 1 (entra corriente). Es lo que se conoce como código máquina.
Un programa, por lo tanto, no es más que una secuencia de 0 y 1. El problema es que a las personas nos resulta muy difícil escribir instrucciones en forma de secuencias de 0 y 1. Para ayudarnos con este trabajo existen programas especializados denominados entornos de programación. Un entorno de programación permite escribir programas utilizando un código específico muy parecido al lenguaje humano, normalmente al inglés, llamado lenguaje de programación. En la pantalla de debajo puedes ver un pequeño fragmento del código de un programa, el llamado código fuente, escrito en el lenguaje de programación C++. Cuando se acaba de escribir el código fuente, el mismo entorno de programación se encarga de traducirlo a código máquina, para que pueda interpretarlo el ordenador. Esta traducción de un lenguaje a otro se denomina compilación.

1. Código fuente *en lenguaje de programación C++. Es escrito por un programador.*

2. Compilación: *Traducción del código fuente a código máquina.*





3. Código máquina *Formado por 0 y 1, es el que entiende el ordenador. Se almacena en la memoria.*

2. Lenguajes de programación. Programación textual y programación visual

Hay muchos lenguajes de programación: Visual basic, C++, Java, JavaScript, PHP, etc. Estos lenguajes, que son usados por los programadores profesionales, son lenguajes textuales. Se escriben línea a línea de forma similar a como escribimos un texto (ver imagen derecha). Cada línea de código es una instrucción que debe ejecutar el ordenador. Son lenguajes muy potentes, con poco código se pueden escribir muchas instrucciones. Sin embargo, se necesitan muchas horas para aprenderlos a usar. Para que personas sin experiencia puedan empezar a programar de forma sencilla, se han inventado los lenguajes visuales, como Scratch. En los lenguajes visuales se programa uniendo bloques, que son dibujos que tienen código asociado.

```
var age = 15;

if( age > 18 ){
  document.write("Qualifies for driving");
}
else{
  document.write("Doesn't qualify for driving");
}
```

Fragmento de código en JavaScript, un ejemplo de lenguaje de programación textual.

Scratch es un lenguaje de programación visual.

3. Algoritmos

Las instrucciones que forman un programa deben escribirse siguiendo un orden lógico para que el programa funcione correctamente. Antes de comenzar a escribir código hay que describir exactamente, paso a paso, qué queremos que haga el programa y en qué orden. Es lo que se conoce como algoritmo.

La representación gráfica de un algoritmo se hace mediante un diagrama de flujo. El proceso de cruzar un paso de peatones, por ejemplo, se puede representar con el diagrama de flujo de debajo.

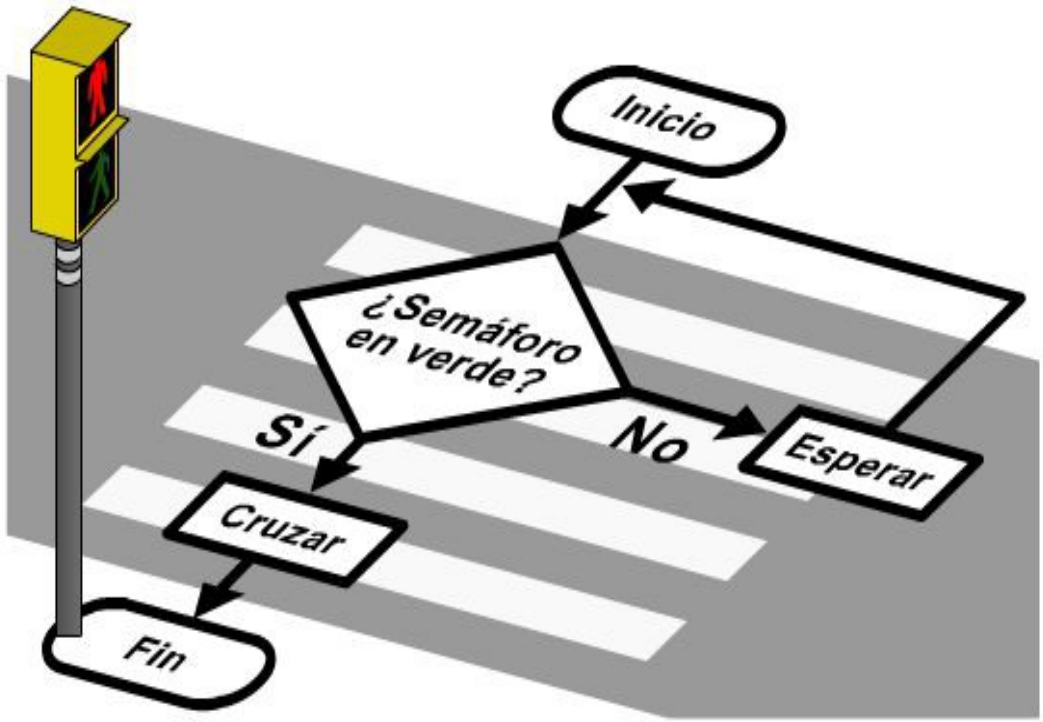
Símbolos básicos de los diagramas de flujo

Inicio o final de diagrama

Realización de una actividad/proceso/orden

Análisis de la situación y toma de decisiones

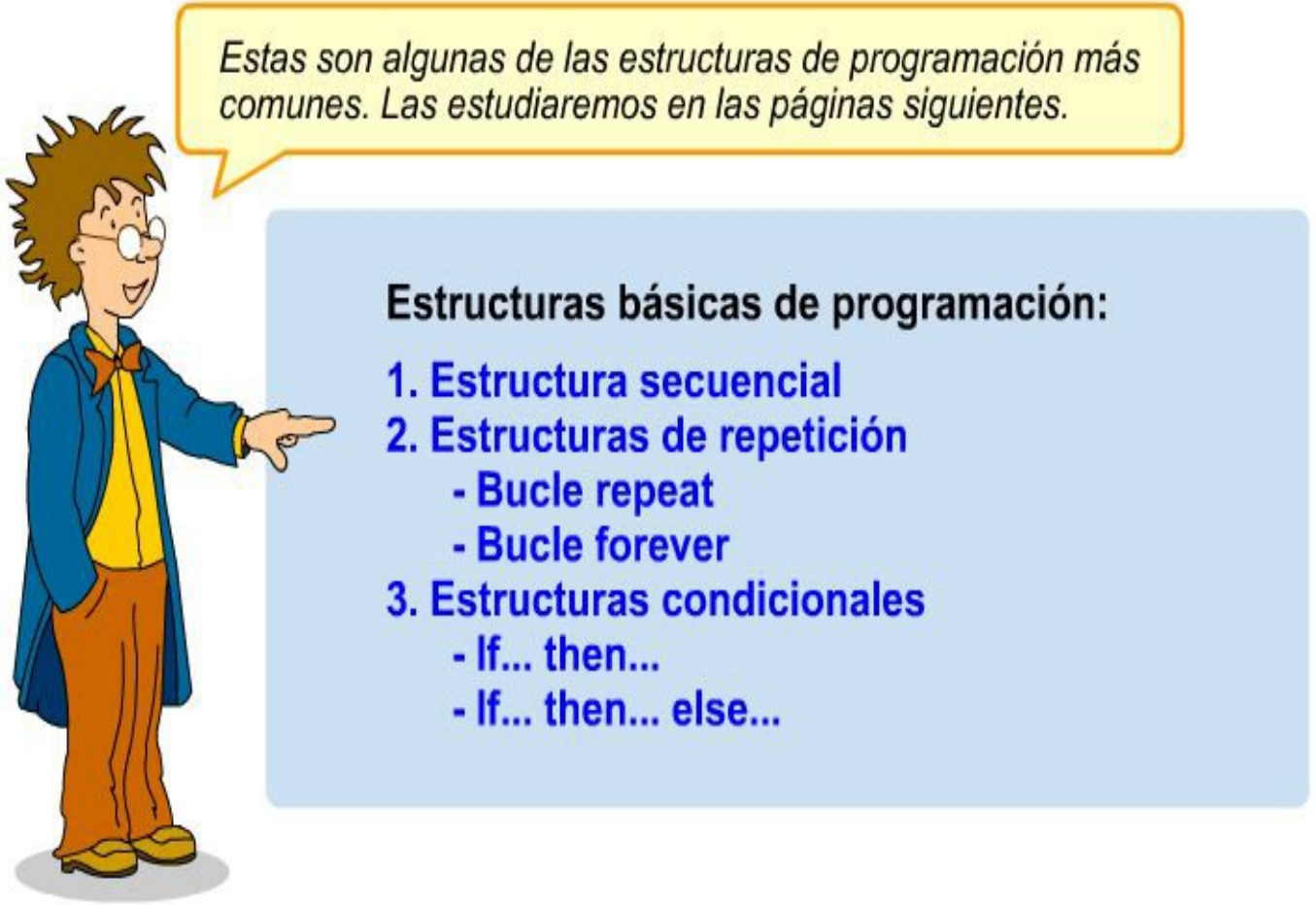
Indica el flujo del proceso



Algoritmo que seguimos para cruzar un paso de cebra con semáforo.

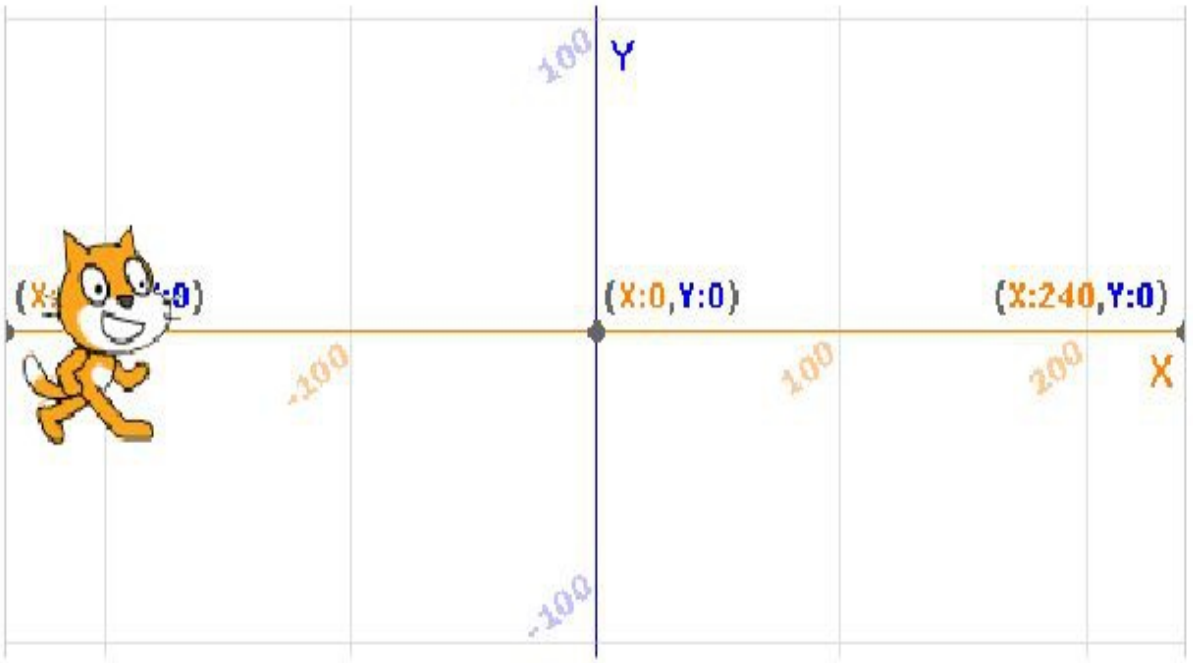
4. Estructuras básicas de programación

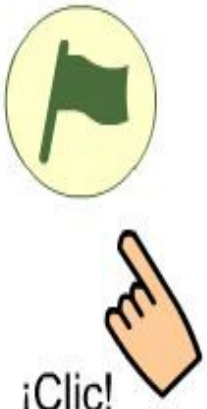
El ordenador lee las instrucciones que hay en un programa de una a una y siguiendo un orden preestablecido, es lo que se llama el flujo del programa. Para definir qué flujo tendrá un programa que estamos escribiendo, podemos combinar diferentes estructuras de programación. En las páginas siguientes veremos las estructuras de programación más comunes, son las que usaremos para hacer juegos en Scratch. Aprender a utilizar estas estructuras es fundamental para saber programar.

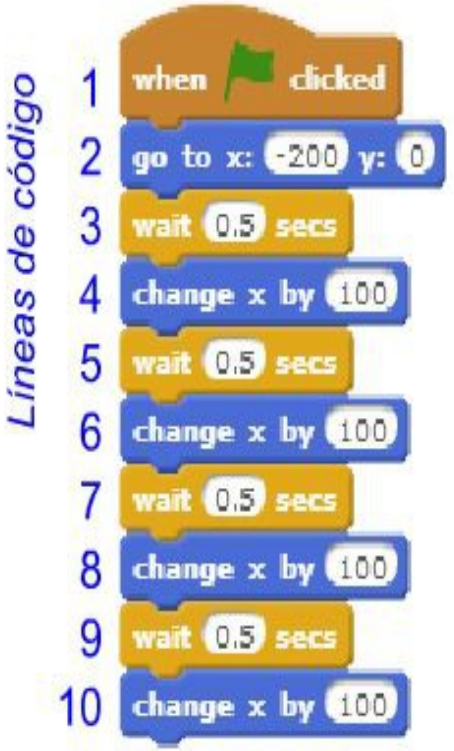


4.1. Estructura secuencial

La estructura básica de todos los programas es la secuencial. Esto quiere decir que las instrucciones se ejecutan una tras otra, siguiendo el orden en el que se han escrito. El ordenador lee la primera línea que encuentra (la de arriba) y la ejecuta. Cada línea de código es una instrucción, una orden que debe cumplir. Después lee la siguiente línea y la ejecuta. Así sucesivamente hasta que llega a la última línea (la de abajo), donde el programa se detiene. En este ejemplo, el programa se inicia cuando se clica la bandera verde (línea 1). A continuación la mascota de Scratch se sitúa en el punto x=-200, y=0 (línea 2). Después espera 0,5 segundos (línea 3). De la misma manera, se van leyendo las siguientes líneas hasta llegar a la última (línea 10). Como puedes ver, las líneas 5 a 10 son 3 repeticiones de las líneas 3 y 4.







Clica en la bandera verde para ejecutar el programa de la derecha.

Programa en Scratch.

4.1. Estructura secuencial. Algoritmo

Este es el algoritmo del programa de la página anterior representado con un diagrama de flujo.

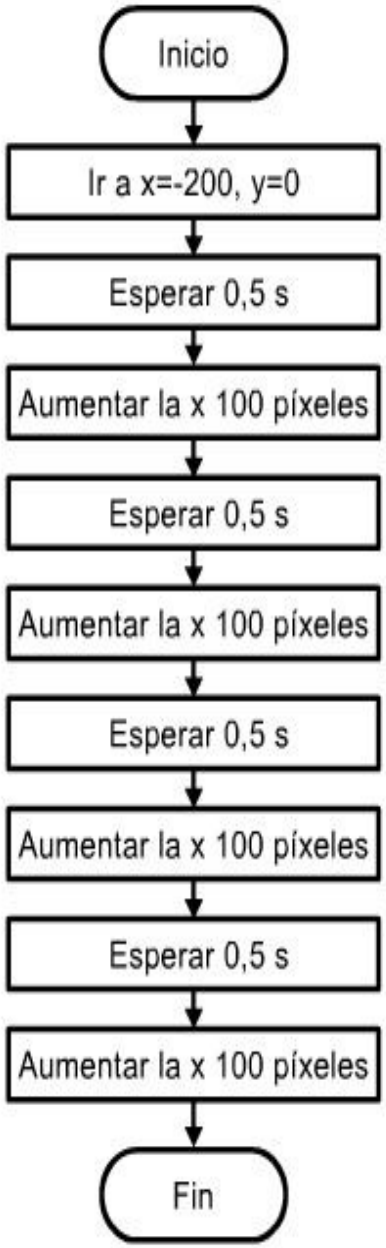
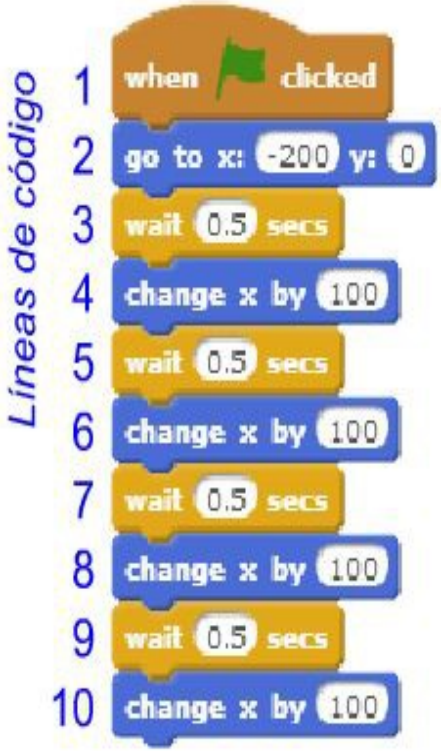


Diagrama de flujo del algoritmo.

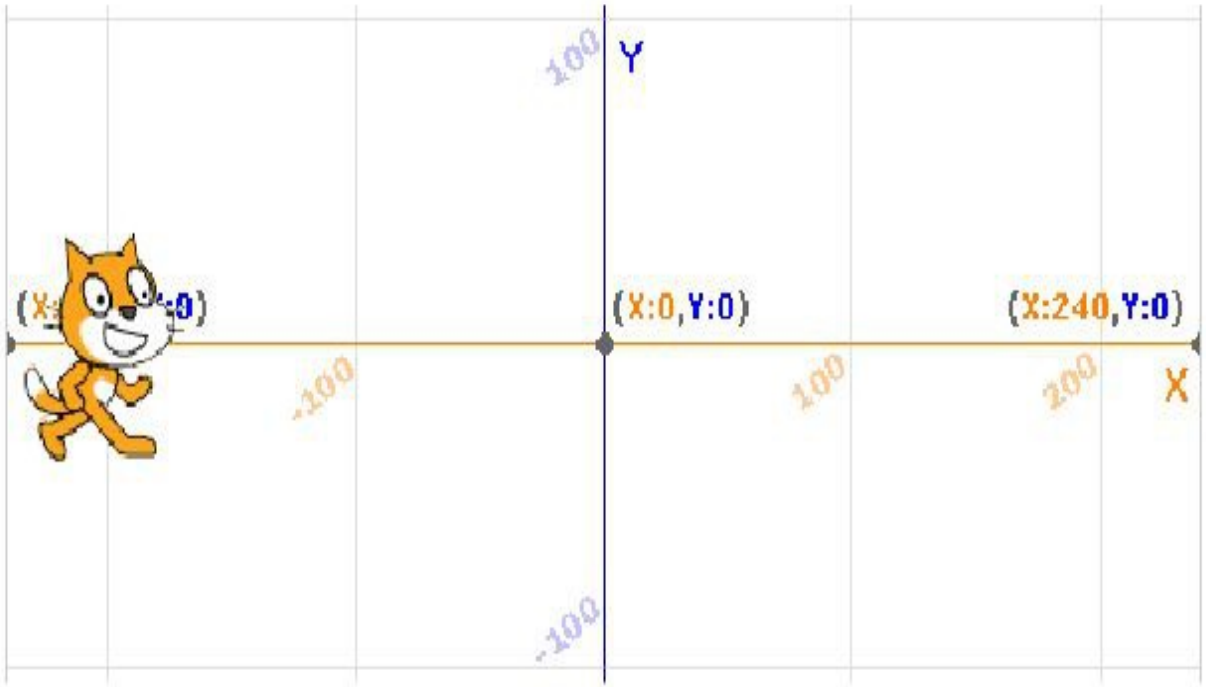


Programa en Scratch.

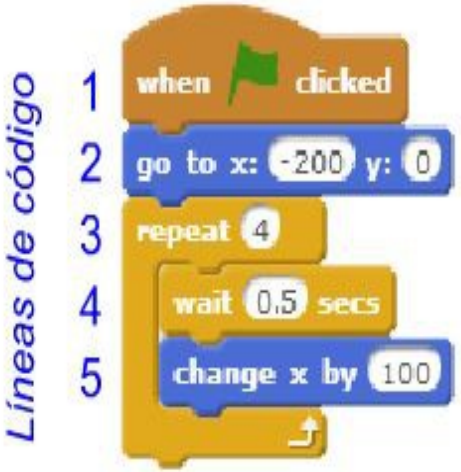
4.2. Estructuras de repetición. Bucle repeat

Aunque la estructura secuencial es la base de todos los programas, también son muy frecuentes los bucles. Los bucles son repeticiones de una o varias instrucciones. Se usan para no tener que escribir el mismo código varias veces. En este ejemplo se utiliza el bucle repeat para ordenar al programa que repita las líneas de código 4 y 5. El número que hay después de la palabra "repeat" indica las veces que el bucle debe repetirse antes de continuar con la siguiente línea de la secuencia principal del programa. Se suele utilizar la letra "i" para indicar el número de la iteración (repetición). Este programa es equivalente al del ejemplo anterior, pero con un código algo más breve.

Bloque repeat en Scratch.



Clica en la bandera verde para ejecutar el programa de la derecha.



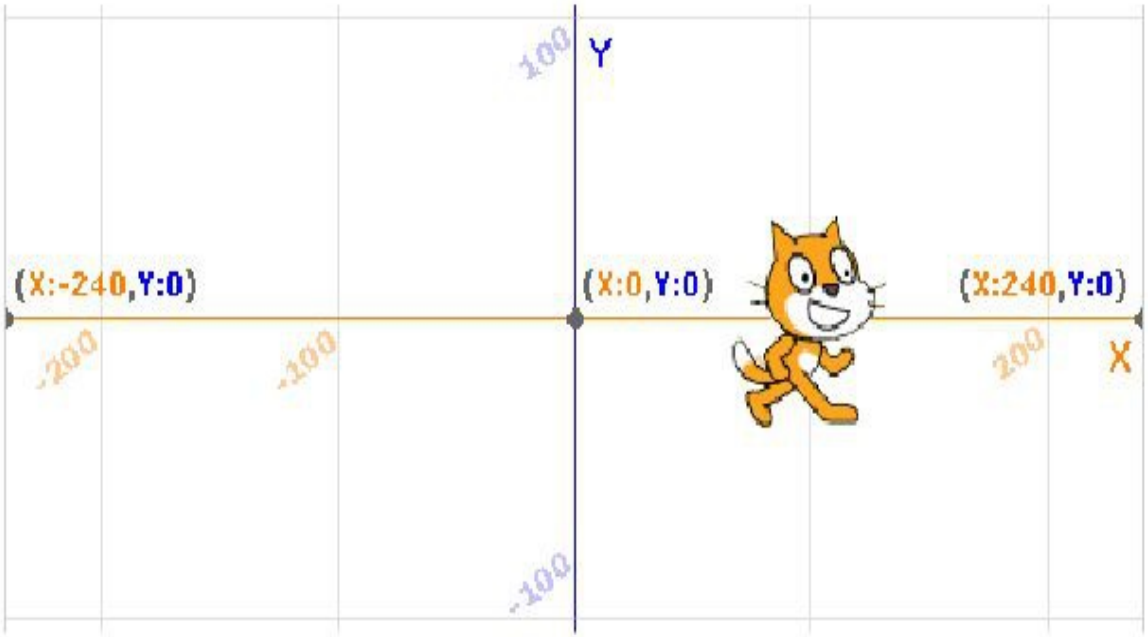
i = 0

Programa en Scratch.

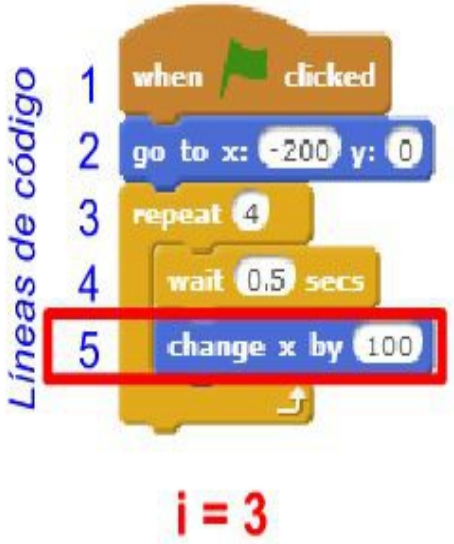
4.2. Estructuras de repetición. Bucle repeat

Aunque la estructura secuencial es la base de todos los programas, también son muy frecuentes los bucles. Los bucles son repeticiones de una o varias instrucciones. Se usan para no tener que escribir el mismo código varias veces. En este ejemplo se utiliza el bucle repeat para ordenar al programa que repita las líneas de código 4 y 5. El número que hay después de la palabra "repeat" indica las veces que el bucle debe repetirse antes de continuar con la siguiente línea de la secuencia principal del programa. Se suele utilizar la letra "i" para indicar el número de la iteración (repetición). Este programa es equivalente al del ejemplo anterior, pero con un código algo más breve.

Bloque repeat en Scratch.



Clica en la bandera verde para ejecutar el programa de la derecha.



Programa en Scratch.

4.2. Estructuras de repetición. Algoritmo del bucle repeat

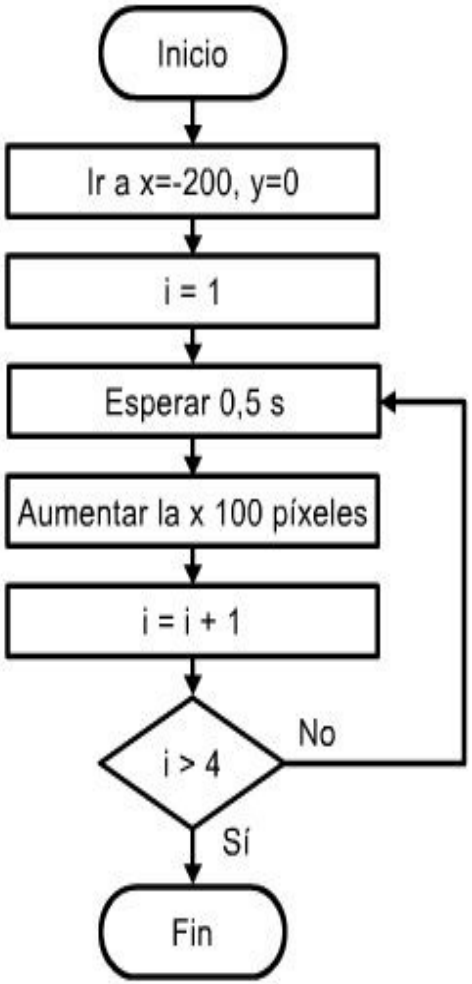
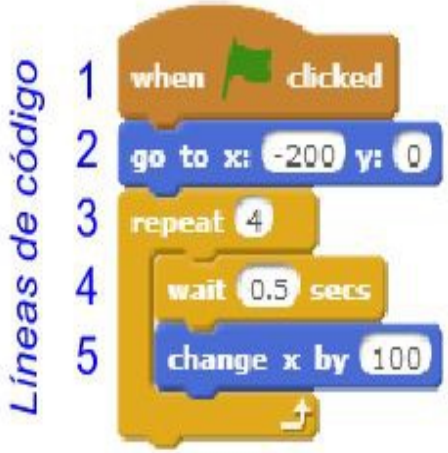


Diagrama de flujo del algoritmo.

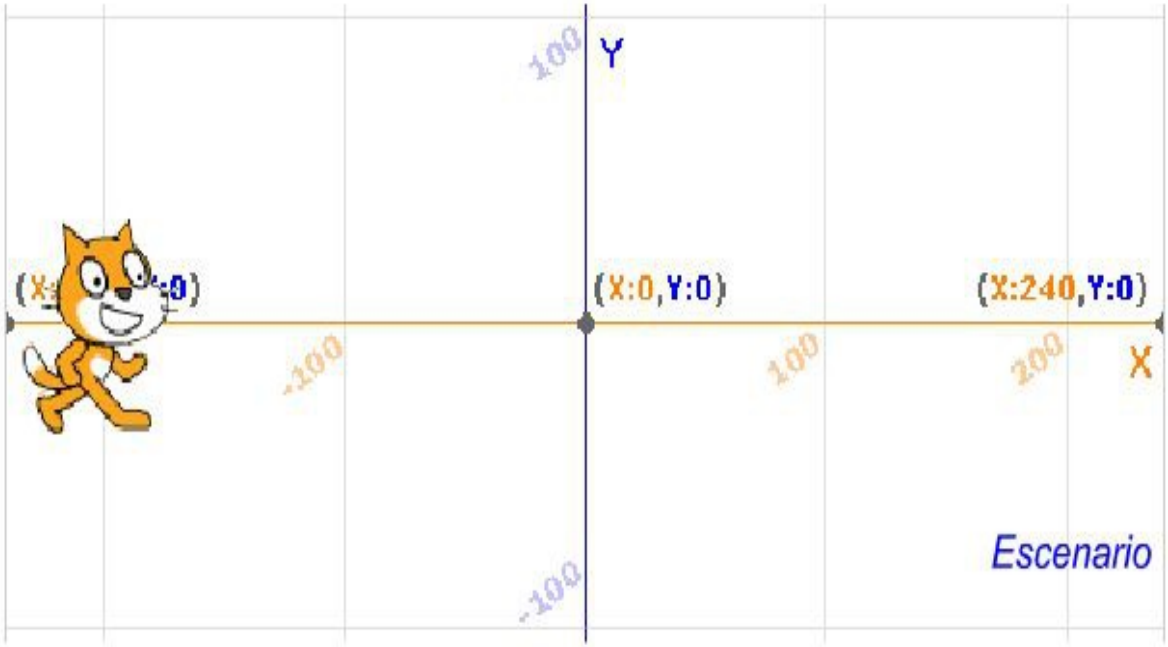


Programa en Scratch.

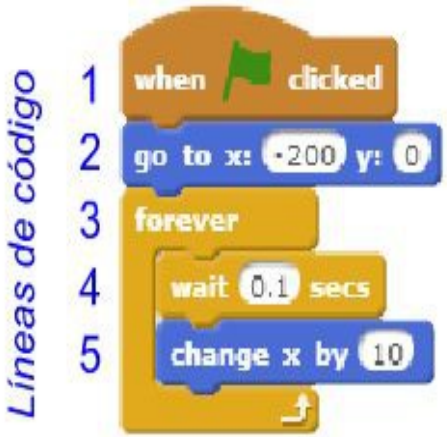
4.2. Estructuras de repetición. Bucle forever

El bucle forever (en inglés, para siempre) es muy similar al repeat, solo que no acaba nunca. En esta página se ha modificado el ejemplo anterior y se ha sustituido el bucle repeat por un bucle forever. Se ha reducido el tiempo de espera (línea 4) y el incremento de la x (línea 5). El resultado es que, al clicar en la bandera verde, la mascota de Scratch va a la posición x =-200, y=0 y luego se inicia un bucle sin fin que lo hace avanzar poco a poco. Siguiendo este programa, el gato avanzaría indefinidamente hacia la derecha. Se para cuando llega al extremo derecho porque en Scratch los objetos no pueden salir del escenario (el recuadro de debajo).

Bloque forever en Scratch.



Clica en la bandera verde para ejecutar el programa de la derecha.



Programa en Scratch.

4.2. Estructuras de repetición. Algoritmo del bucle forever

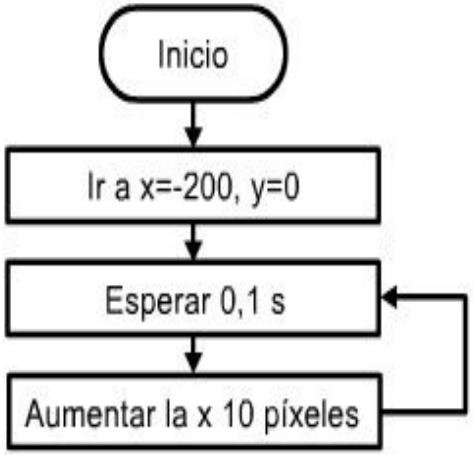
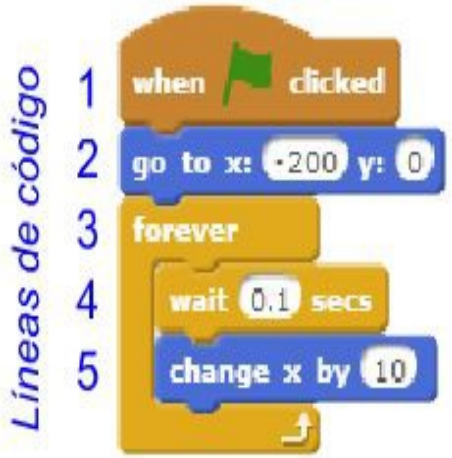


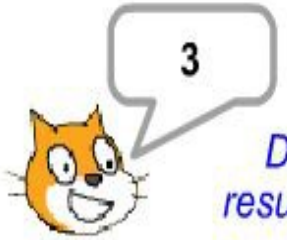
Diagrama de flujo del algoritmo.



Programa en Scratch.

Operadores

Antes de continuar, hacemos un paréntesis para hablar de los operadores. Son elementos de programación que permiten interrelacionar datos. Aquí puedes ver los más comunes:

Operador	Ejemplos
 Suma, resta, multiplicación y división.	 *Di "1+2"*  *Devuelve el número resultado de la operación*
 Condición "menor que", "igual que" y "mayor que".	 *Di "2>1"*  *Devuelve True (verdadero), si la condición es cierta y False (falso), si no lo es*

 Operador lógico AND: devuelve *True* si los dos operandos (los espacios hexagonales) son verdaderos, y *False* si al menos uno de los dos operandos es falso.

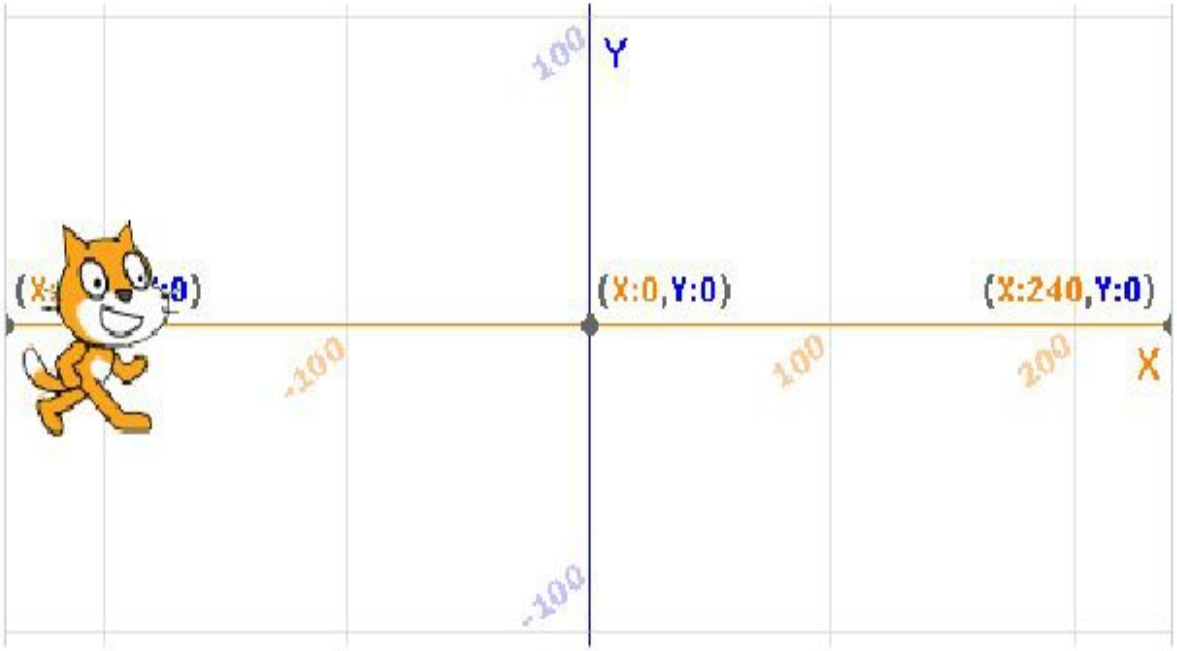
 Operador lógico OR: devuelve *True* si al menos uno de los dos operandos es verdadero.

 Número al azar: genera un número aleatorio comprendido entre los valores que indiquemos.

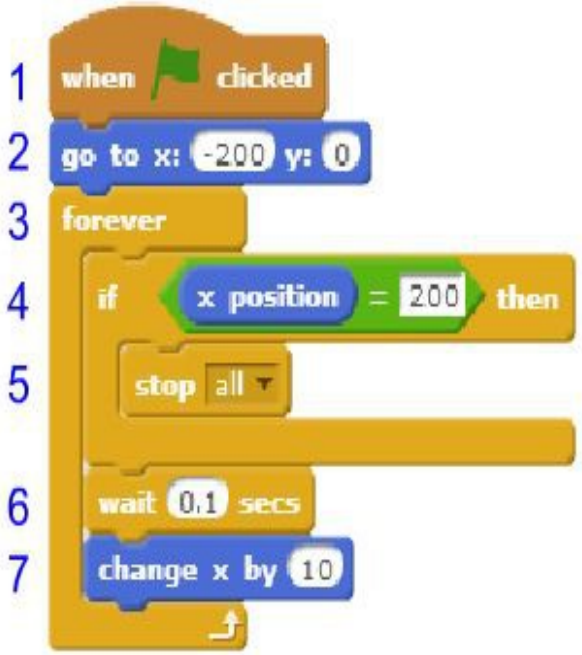
4.3. Estructuras condicionales. If... then...

En muchas ocasiones un programa tiene que tomar una decisión en función de una condición externa. Algo así como: "Si pasa esto, entonces haz esto otro". Esto se implementa en los lenguajes de programación con la expresión inglesa "If... then...". Un ejemplo de la vida diaria es un paso de peatones con semáforo. Nuestra actuación será: "Si el semáforo está en rojo, entonces me paro". En un programa quedaría: "If semáforo = rojo then parar". En el ejemplo de esta página se ha añadido una estructura if... then... al programa de la página 11/18 (líneas 4 y 5). El resultado es que el gato se para cuando llega a la posición x=200, en lugar de continuar a la derecha indefinidamente, como hacía antes.

Bloque if then en Scratch.



¡Clic!



Clica en la bandera verde para ejecutar el programa de la derecha.
Ten en cuenta que la animación va más lentamente que el programa real.

Programa en Scratch.

4.3. Estructuras condicionales. Algoritmo de if... then...

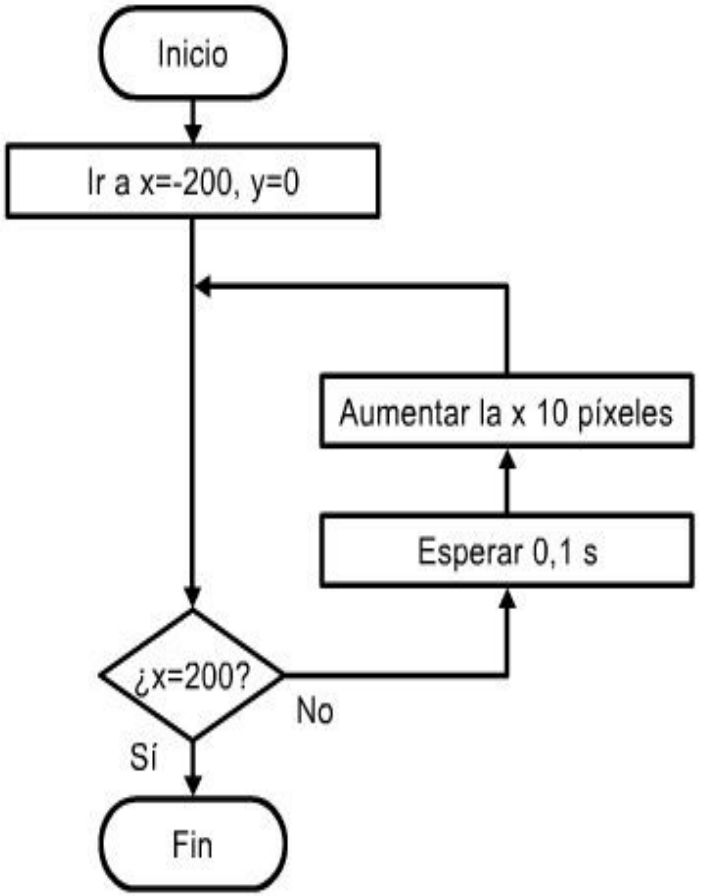
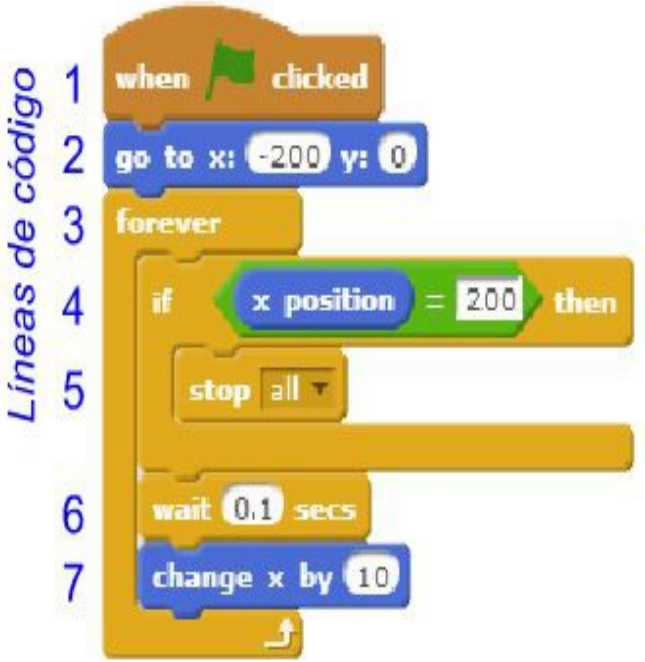


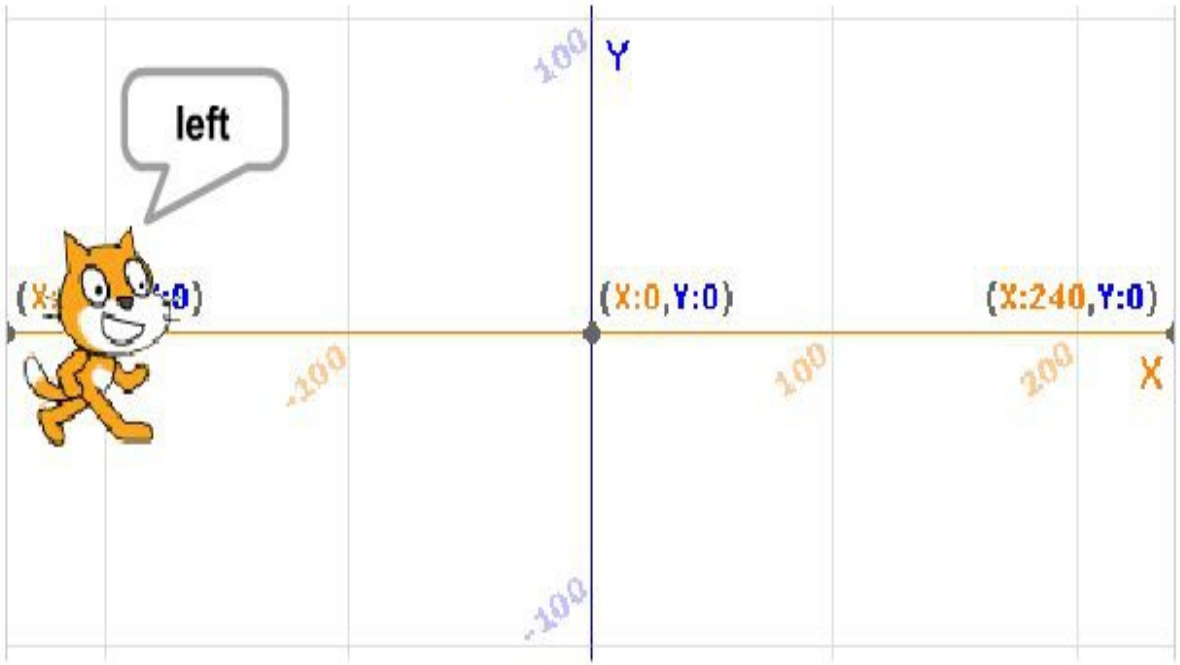
Diagrama de flujo del algoritmo.



Programa en Scratch.

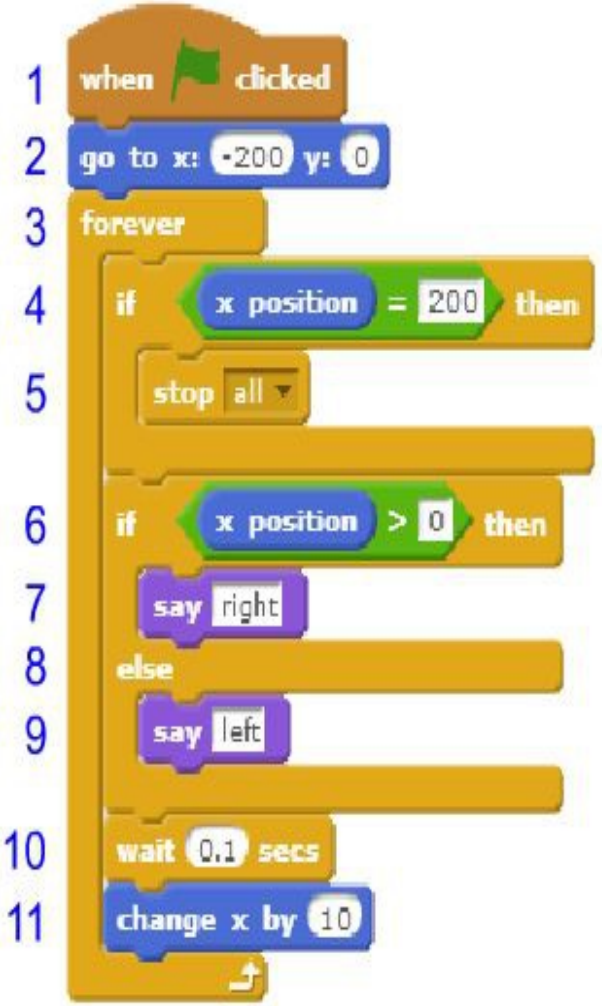
4.3. Estructuras condicionales. If... then... else...

A veces no es suficiente con especificar qué decisión tomar si se da una condición y también interesa determinar qué otra decisión tomar en caso de que la condición no se cumpla. Algo así como: "Si pasa esto, entonces haz aquello, si no, haz esto otro". En programación se implementa con la expresión inglesa "If... then... else...". Siguiendo con el ejemplo del paso de peatones, nuestra actuación será: "Si el semáforo está en rojo, entonces me paro, si no, paso". En el ejemplo de esta página se ha añadido una estructura if... then... else... al programa de la página 14/18 (líneas 6 a 9). Esta hace que el gato diga "right" cuando su posición x es mayor que 0 y "left" cuando no se da esta condición.



Clica en la bandera verde para ejecutar el programa de la derecha.
Ten en cuenta que la animación va más lentamente que el programa real.

Bloque if then else en Scratch.



Programa en Scratch.

4.3. Estructuras condicionales. Algoritmo de if... then... else...

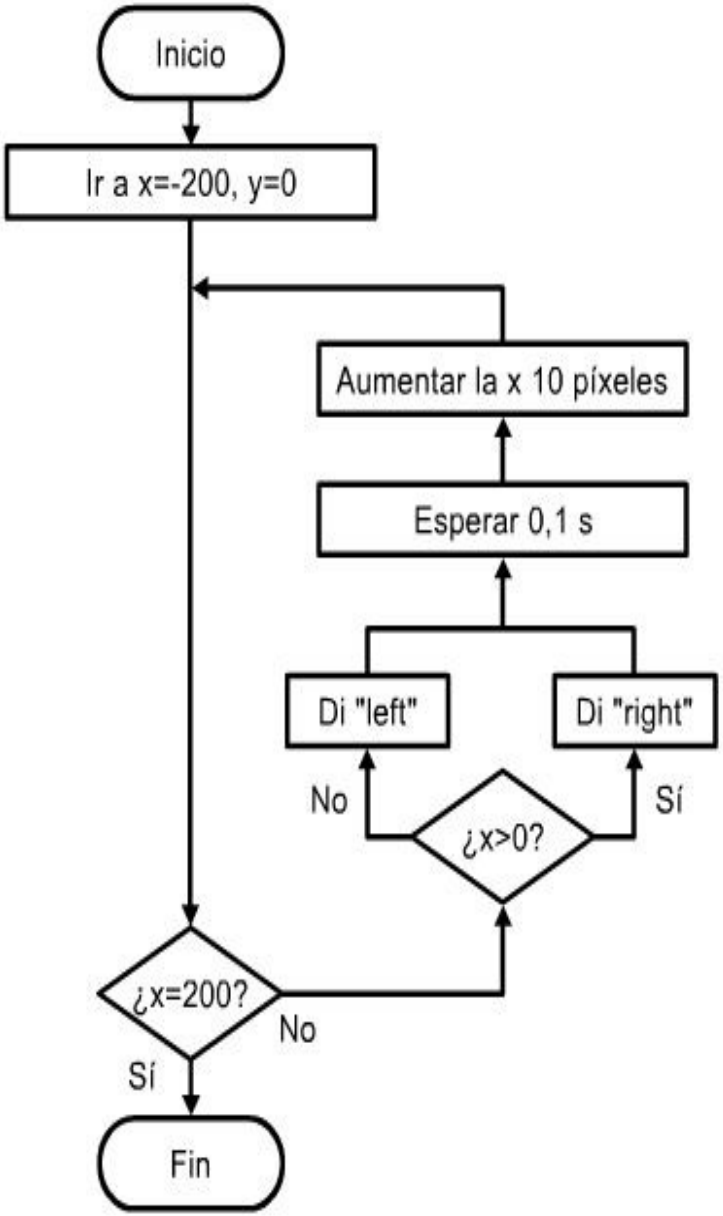
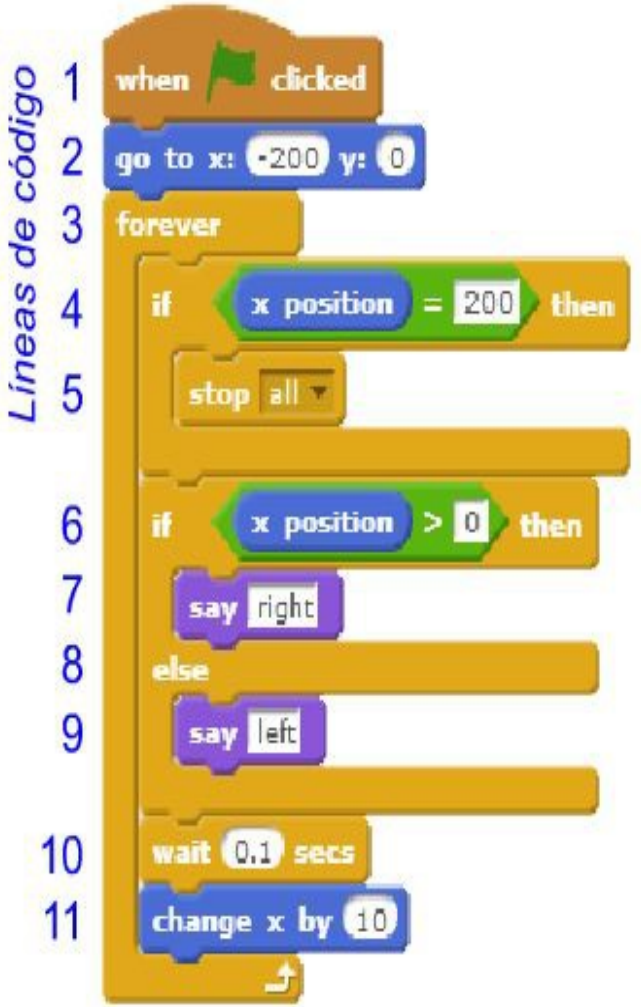


Diagrama de flujo del algoritmo.



Programa en Scratch.

Cuestionario

1. ¿Qué es Scratch?
2. ¿Qué es un programa de ordenador? ¿Qué es programar?
3. ¿En qué componente físico del ordenador se ejecutan los programas?
4. Los programas están formados por 3 bloques de instrucciones. Explícalos brevemente.
5. ¿Qué es un lenguaje de programación?
6. Pon 3 ejemplos de lenguajes de programación.
7. ¿Qué es un algoritmo? ¿Cómo se representa gráficamente?
8. Cita las estructuras básicas de programación más comunes.
9. Explica cómo funciona el programa de la página 16/18.
10. Cuando hayas hecho las miniunidades "Frontón 1" y "Frontón 2" y ya sepas cómo utilizar Scratch, reproduce los 5 programas que aparecen en esta miniunidad y observa cómo funcionan.