

I  
ESO

# Programación, Inteligencia Artificial y Robótica



Arturo Gómez Gilaberte  
Eva Parramón Ponz  
Carmen Sánchez-Seco Peña

EDITORIAL  
DONOSTIARRA

Pokopandegi, nº 4 - Pabellón Igaralde - Barrio Igara  
Teléfonos 943 215 737 - 943 213 011 - 943 214 406  
20018 - SAN SEBASTIÁN  
[info@editorialdonostiarra.com](mailto:info@editorialdonostiarra.com) - [www.editorialdonostiarra.com](http://www.editorialdonostiarra.com)

Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra sólo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley. Diríjase a CEDRO (Centro Español de Derechos Reprográficos) si necesita fotocopiar o escanear algún fragmento de esta obra ([www.conlicencia.com](http://www.conlicencia.com); 91 702 19 70 / 93 272 04 47).

© EDITORIAL DONOSTIARRA, S.A.

Arturo Gómez Gilaberte

Eva Parramón Ponz

Carmen Sánchez-Seco Peña

Diseño y maquetación: Eva Cuesta

Ilustración: Israel Luengo

Corrección: David Aguilar España

Edita: Editorial Donostiarra

Pokopandegi, 4, 20018 - San Sebastián (España)

Imprime: Digitarte

Polígono Ibaiondo - Pabellón 9  
20120 Hernani (Guipúzcoa)

ISBN: 978-84-7063-698-1

Depósito legal: DL SS-00329-2023

Impreso en España - Printed in Spain

# Presentación



La materia optativa Programación, Inteligencia Artificial y Robótica ofrece las bases para que el alumnado pase de estar familiarizado con el uso diario de los dispositivos electrónicos y con las tecnologías de la comunicación a comprender su funcionamiento y a adoptar un papel activo usando determinadas técnicas y formas de analizar, organizar y relacionar ideas a la hora de resolver problemas, que pueden ser extrapoladas a otros ámbitos de la vida y a otras disciplinas. Los contenidos de la materia se organizan en tres bloques: programación, inteligencia artificial y robótica.

El libro de texto de esta asignatura se adapta a la ley de educación LOMLOE para facilitar que el alumnado desempeñe las **competencias específicas** y adquiera los **saberes básicos** establecidos en esta ley.

En él se ponen a disposición del profesorado todo tipo de recursos, para que pueda elegir en cada momento lo que más se aadecue a sus necesidades:

- **Contenidos** que desarrollan los conocimientos, destrezas y actitudes que constituyen los **saberes básicos**, explicados de una forma clara y con ejemplos y actividades.
- Gran cantidad de **actividades de refuerzo**, estructuradas según los bloques de contenido de la unidad y graduadas en dificultad, que facilitan **situaciones de aprendizaje** diversas, organizadas y adaptadas al nivel de competencia curricular del alumnado.



# Secciones de cada unidad



## Portada y “Descubre”



**Portada.** Incluye una foto representativa de la unidad, unas preguntas motivadoras y el índice general de contenidos (saberes básicos), así como la situación de aprendizaje que sirve de hilo conductor de la unidad.

**“Descubre”.** La segunda página de cada unidad contiene un QR para ver un vídeo introductorio, un breve texto de presentación y unas actividades de iniciación para que el alumno descubra sus conocimientos previos sobre cada tema.



## Contenidos teóricos

En esta sección, marcada con el color azul, se recogen los conocimientos y destrezas de los saberes básicos, siempre de una manera clara y concisa.

Los conceptos y definiciones más importantes se muestran marcados con color amarillo para que sean fácilmente identificables por el alumno.

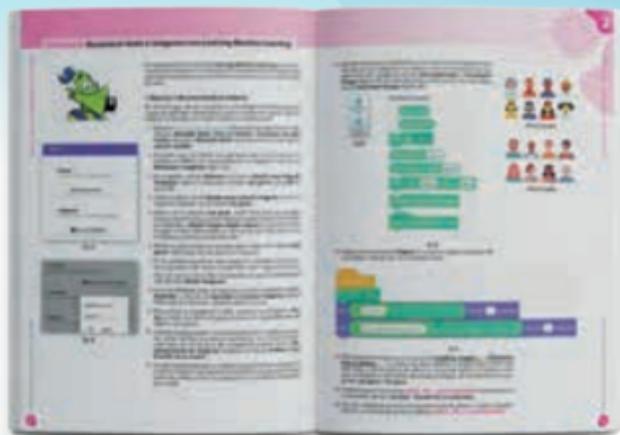
Se incluyen también:

- **Ejemplos** resaltados para favorecer su identificación
- **Clips** para destacar contenidos relacionados o que tienen especial relevancia
- **Códigos QR** de enlace a vídeos explicativos relacionados con los contenidos



## "Aprende con el ordenador"

En esta sección, identificada con el color rosa, se generan situaciones de aprendizaje que ayudarán al alumno a asimilar conocimientos, destrezas y actitudes al tiempo que se familiariza con las **herramientas informáticas**. Facilitan la labor del profesor, ya que se trata de prácticas guiadas que los alumnos pueden hacer de forma autónoma. Suelen incluir también **desafíos**.



## "Actividades de refuerzo"

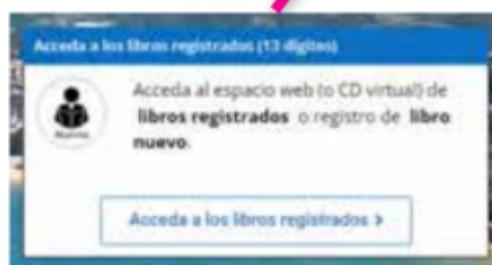
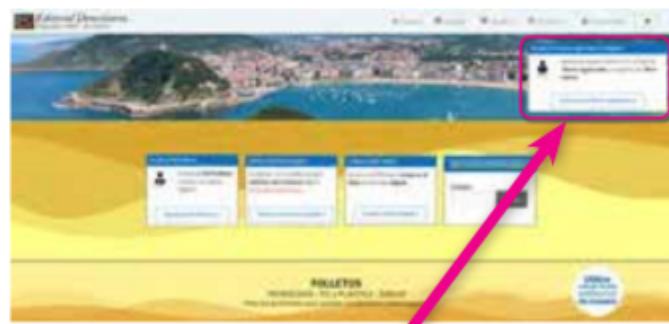
Las actividades han sido diseñadas para crear situaciones de aprendizaje y facilitar al alumnado el razonamiento y la adquisición de los saberes básicos. Las de esta sección, marcada con el color marrón, están organizadas siguiendo los apartados de la unidad, para que el profesor identifique fácilmente su contenido, y, como todas las demás, clasificadas según el grado de dificultad con una, dos o tres marcas.

Estas actividades se pueden responder desde la **plataforma digital** y son todas ellas de **autocorrección**, de forma que cuando el profesor crea la clase puede realizar un seguimiento directo del trabajo individual de cada alumno, que es calificado de forma automática. Además, los alumnos reciben información sobre los logros que van consiguiendo.

## **Espacio web**

Los alumnos pueden acceder a la **zona web** de Editorial Donostiarra, que contiene todos los archivos necesarios para realizar las prácticas del libro.

Para **register el libro y acceder al espacio web**, entra en [www.editorialdonostiarra.com](http://www.editorialdonostiarra.com) y haz clic en **Acceda a los libros registrados**:



Necesitarás el **código individual** que figura en la parte interior de la **contraportada**.

Además, si lo deseas, una vez registrado podrás solicitar **gratuitamente la licencia digital** de este libro.



# Índice

## 1 Programación

1. Pensamiento computacional PÁG. 12
  2. Qué son los algoritmos PÁG. 13
  3. Algoritmos. Representación gráfica PÁG. 14
  4. Lenguajes de programación. Tipos PÁG. 16
  5. Programación por bloques: Scratch PÁG. 17
  6. Análisis y validación de software PÁG. 20
  7. Software libre y software propietario PÁG. 21
- Aprende con el ordenador PÁG. 22
- Actividades de refuerzo PÁG. 56

## 2 Inteligencia artificial

1. Qué es la inteligencia artificial PÁG. 66
  2. Enfoques de la inteligencia artificial PÁG. 66
  3. Machine learning. Tipos PÁG. 68
  4. Impacto social de la inteligencia artificial. Los sesgos PÁG. 68
  5. Aplicaciones de la inteligencia artificial PÁG. 70
- Aprende con el ordenador PÁG. 72
- Actividades de refuerzo PÁG. 90

- Estructuras de los algoritmos

- Tipos de lenguajes de programación

- Scratch

- Fundamentos de la programación por bloques

- **Práctica 1.** Resolución de problemas. Infografía con Canva
      - **Práctica 2.** ¿Cómo se hacen los diagramas de flujo?
      - **Práctica 3.** Traducir una receta de cocina a un algoritmo
      - **Práctica 4.** Depurar algoritmos. Esquema de pasos con Padlet
      - **Práctica 5.** Juego con algoritmos. La Hora del Código
      - **Práctica 6.** Tus primeros programas. Gobstones
      - **Práctica 7.** Programar un algoritmo. Me presento con Scratch
      - **Práctica 8.** Programar un algoritmo. El problema del acné
      - **Práctica 9.** Programar un algoritmo. Mis ahorros
      - **Práctica 10.** Programar un videojuego. *El elefante hambriento*
      - **Práctica 11.** Extensiones de Scratch. Integración de gráficos y sonidos
      - **Práctica 12.** Polígonos de colores. Subalgoritmos en Scratch

- Modelo top-down

- Modelo bottom-up

- Sesgos de la inteligencia artificial

- **Práctica 1.** Akinator, el genio de la Web
      - **Práctica 2.** Quick Draw! ¡A dibujar!
      - **Práctica 3.** AutoDraw. Dibujar nunca fue tan fácil
      - **Práctica 4.** Generar imágenes con Stable Diffusion
      - **Práctica 5.** Conversaciones inteligentes y creación de imágenes con la IA de Bing
      - **Práctica 6.** Moral Machine, la máquina moral
      - **Práctica 7.** Detectar sentimientos con Machine Learning for Kids
      - **Práctica 8.** Reconocer texto e imágenes con Learning Machine Learning
      - **Práctica 9.** Trabajar con datos en Learning Machine Learning
      - **Práctica 10.** El modo avanzado de Learning Machine Learning
      - **Práctica 11.** Reconocer componentes eléctricos con Learning Machine Learning



## 3 Robótica

1. Automatismos y robots **PÁG. 98**

2. Robots: tipos. Grados de libertad **PÁG. 99**

- Tipos de robots
- Grados de libertad

3. Funcionamiento de un robot **PÁG. 100**

4. Cómo percibe un robot **PÁG. 101**

5. Cómo actúa un robot **PÁG. 102**

6. Cómo controla un robot **PÁG. 103**

7. Software de programación **PÁG. 104**

- Ventana de MakeCode

Aprende con el ordenador **PÁG. 105**

- Práctica 1. Primeros pasos con la tarjeta micro:bit y MakeCode
- Práctica 2. Dado digital (con micro:bit y MakeCode)
- Práctica 3. Podómetro (con micro:bit y MakeCode)
- Práctica 4. Termómetro (con micro:bit y MakeCode)
- Práctica 5. Linterna detectora de luz (con micro:bit y MakeCode)
- Práctica 6. Caja de música (con micro:bit y MakeCode)
- Práctica 7. Enviar mensajes entre tarjetas micro:bit
- Práctica 8. Comunicar Scratch con micro:bit y probar el juego *Flying Bicycle*
- Práctica 9. Theremín (con micro:bit y Scratch)
- Práctica 10. Elementos de un robot
- Práctica 11. Movimiento, luces y sonido en el coche Maqueen
- Práctica 12. Maqueen siguelíneas
- Práctica 13. Maqueen esquivaobstáculos
- Práctica 14. Cutebot siguelíneas
- Práctica 15. Cutebot controlado por control remoto desde otra tarjeta micro:bit
- Práctica 16. Reconocer y seguir una línea mediante inteligencia artificial con AI Lens y Cutebot
- Práctica 17. Pulsómetro
- Práctica 18. Robot gusano
- Práctica 19. Un monstruo "de la leche"

Actividades de refuerzo **PÁG. 134**

# Autorización de creación de cuentas de usuario para menores

## MODELO PARA FOTOCOPIAR, RELLENAR Y ENTREGAR AL PROFESOR

NOMBRE DEL ALUMNO/A: \_\_\_\_\_ CURSO: \_\_\_\_\_

CENTRO EDUCATIVO: \_\_\_\_\_

Los padres o tutores:

- AUTORIZAN** al alumno/a a **crear las cuentas de usuario** necesarias para que pueda realizar las prácticas de la asignatura **Programación, Inteligencia Artificial y Robótica I**, recogidas en el libro, **bajo la supervisión del profesor y CONOCEN** que son responsabilidad del alumno/a la utilización de las distintas cuentas con fines educativos y la custodia de las contraseñas.
- NO AUTORIZAN** al alumno/a a crear las cuentas de usuario.

Puede consultarse el libro de texto para conocer los sitios web donde se van a crear cuentas, con el fin de acceder a las aplicaciones educativas y conocer la política de privacidad relativa a la protección de datos de carácter personal de dichas aplicaciones.

El alumno/a puede consultar sus derechos en el Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, relativo a la protección de datos, y en la Ley Orgánica 3/2018, de 5 de diciembre, de protección de datos personales y garantía de los derechos digitales.

En \_\_\_\_\_, a \_\_\_\_ de \_\_\_\_\_ del 20\_\_\_\_

Dña. \_\_\_\_\_

Firmado: la madre o tutora

D. \_\_\_\_\_

Firmado: el padre o tutor

**Nota para el profesor:** Esta autorización tiene carácter orientativo y puede modificarse para adaptarse a las características del centro educativo.

# 1

# Programación

¿Sabes qué es un algoritmo? ¿Lo has utilizado alguna vez?  
¿Conoces algún lenguaje de programación?

## Situación de aprendizaje

Dar respuesta a un problema de la vida diaria estableciendo algoritmos que permitan su resolución y codificándolos en lenguajes de programación sencillos

## Contenidos de la unidad

1. Pensamiento computacional
2. Qué son los algoritmos
3. Algoritmos. Representación gráfica
4. Lenguajes de programación. Tipos
5. Programación por bloques: Scratch
6. Análisis y validación de software
7. Software libre y software propietario



Escanea este código para ver el video "¿Qué es un algoritmo?", del canal BBC Learning.



¿Sabes qué pasos sigue un ordenador para resolver un problema? El pensamiento computacional es una forma de pensamiento en la que analizamos y resolvemos los problemas de la vida diaria como si fuésemos informáticos. Así, analizamos datos, estudiamos problemas y planteamos algoritmos que puedan resolverlos.

Los algoritmos son conjuntos ordenados y finitos de operaciones que nos permiten solucionar problemas. Pueden representarse gráficamente mediante esquemas o diagramas; pueden plantearse en pseudocódigo, lo que facilita la depuración de errores; y pueden programarse con lenguajes de programación, para facilitar su realización mediante herramientas informáticas.



## Descubre

### ACTIVIDADES DE INICIACIÓN

Descubre tus conocimientos previos contestando en tu cuaderno las siguientes preguntas:

- 1. Piensa en las diferentes actividades que realizas para preparar un examen. Anótalas en tu cuaderno de forma ordenada y explica qué puede ocurrir si te saltas alguna de ellas.
- 2. Si estás enfermo y acudes al médico, ¿qué pasos sigue él para curarte? Describe en tu cuaderno qué ocurrió la última vez que fuiste al médico y cómo actuó.
- 3. Piensa y anota el nombre de cinco objetos que tú creas que funcionan mediante un programa.
- 4. ¿Eres capaz de identificar cinco programas que utilices habitualmente? ¿Sabrías decir en qué lenguaje de programación se han codificado?

**1.**

## Pensamiento computacional

En ocasiones, trabajar como lo haría un ordenador nos puede ayudar a resolver problemas de la vida diaria y desarrollar soluciones sencillas y creativas para problemas habituales.

Esta forma de trabajo se denomina **pensamiento computacional** y consiste en abordar los problemas planteados como si fuéramos científicos informáticos:

# 1



Hacemos deducciones, planteamos hipótesis, imaginamos situaciones... Utilizamos el **pensamiento abstracto**.

# 2

**Simplificamos** los elementos de un problema y lo dividimos en otros más sencillos.



# 3



Identificamos el **aspecto esencial** de un problema (datos, condicionantes y restricciones).

# 4

Desarrollamos un modelo que pueda ser una **solución**.



# 5

La solución obtenida debe poder ser ejecutada por un sistema informático.



Escanea este código para acceder a la web Programamos y conocer los juegos CodyRoby (puedes descargar las instrucciones y las cartas de juego).

## 2. Qué son los algoritmos

A lo largo de la historia, la humanidad se ha encontrado con problemas a los que ha tenido que dar una respuesta para continuar con su evolución.

En la antigüedad, los primeros matemáticos y filósofos utilizaron distintos algoritmos para hallar la solución a los problemas planteados (algoritmos de sumas y restas, cálculos de distancias astronómicas, cálculos de magnitudes físicas, etc.).

**Un algoritmo** es una secuencia ordenada de pasos que resuelven un problema en un tiempo finito.

Los algoritmos tienen las siguientes características:

1. Contienen instrucciones concretas, sin ninguna ambigüedad.
2. Deben terminar, es decir, **son finitos**.
3. Todos sus pasos son simples y **están ordenados**.

Los algoritmos, una vez definidos, se escriben en pseudocódigo. El **pseudocódigo**, o “código falso”, es un lenguaje hecho para que lo entiendan los humanos y no las máquinas, por lo que no puede ejecutarse en un ordenador. La finalidad del pseudocódigo es ayudarnos a detectar errores en los algoritmos y a depurarlos.

Actualmente, todos los algoritmos desarrollados pueden resolverse con la ayuda de los ordenadores y los lenguajes de programación.

**Un programa** es la traducción de un algoritmo a un lenguaje de programación capaz de ser entendido por un ordenador y procesado por él.



Escanea este código para ver el video “¿Cómo funcionan los algoritmos en Internet?”, del canal GCFAPrendeLibre.

### 3. Algoritmos. Representación gráfica

Los esquemas y los gráficos nos permiten obtener una idea general de lo que estamos tratando. Así, hacemos esquemas de las ideas principales de un tema o utilizamos señales gráficas, como las señales de tráfico, porque nos muestran de forma clara indicaciones sin necesidad de leer grandes textos.

Los algoritmos seguidos para resolver problemas también pueden representarse gráficamente. Así nos proporcionan una visión general de los pasos que hay que seguir.

El gráfico utilizado para representar un algoritmo se denomina **diagrama de flujo** u **organigrama**, y muestra mediante símbolos unidos por flechas la secuencia de las acciones que se han de realizar.

Los **símbolos utilizados en los diagramas de flujo** son los siguientes:

#### Terminal

Representa el comienzo o el fin del desarrollo de un algoritmo.



#### Proceso

Permite representar cada una de las acciones que hay que realizar para desarrollar el algoritmo.



#### Entrada o salida de información

Se utiliza cuando es necesaria información (datos adicionales para desarrollar el algoritmo) o se presentan datos o resultados.



#### Decisión

Se utiliza cuando es necesario decidir entre dos o más opciones y señala el camino que habrá que seguir según cuál sea la opción elegida.



#### Línea de flujo

Señala el orden en que se desarrollan las acciones en el algoritmo.



Escanea este código para ver el vídeo "Representación de algoritmos", del canal Open Education Edinburgh.

## Estructuras de los algoritmos

Atendiendo a su estructura, podemos clasificar los algoritmos en:

- **Algoritmos de estructura secuencial**, en los que las instrucciones que las componen se cumplen siguiendo el orden en que aparecen
- **Algoritmos de estructura selectiva (o condicional)**, en los que las instrucciones solamente se llevan a cabo si se cumple o no una condición
- **Algoritmos de estructura iterativa (o de repetición)**, en los que algunas instrucciones se repiten varias veces dentro de un bucle



### EJEMPLO RESUELTO DE DIAGRAMA DE FLUJO

Dibuja el diagrama de flujo del algoritmo seguido para multiplicar dos números  $a$  y  $b$ , siempre que ambos sean mayores que 0.

1. Se inicia el algoritmo. Usamos un símbolo de terminal.

2. Introducimos el valor de  $a$  y  $b$ . Como son datos que el programa necesita, utilizamos un símbolo de entrada de información.

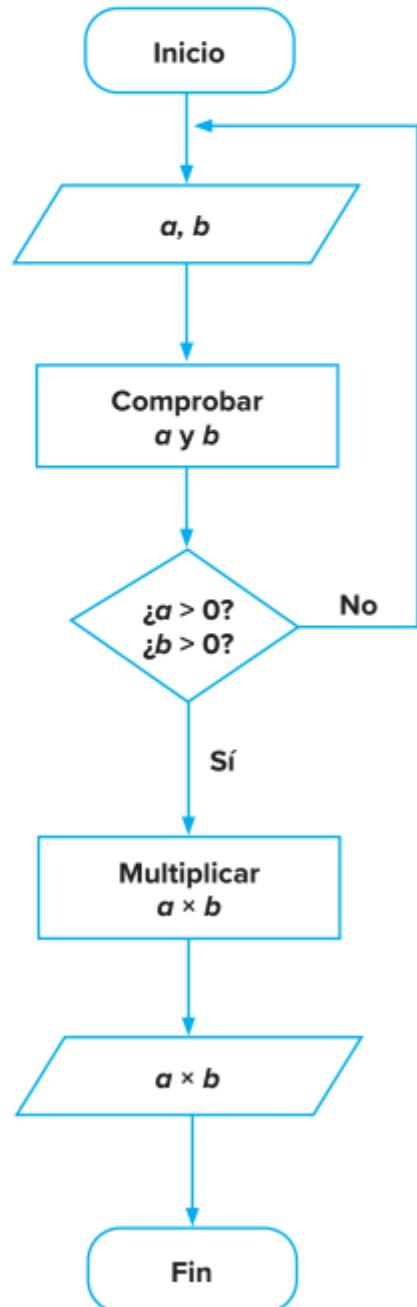
3. Comprobamos  $a$  y  $b$ . Como es una operación, utilizamos un símbolo de proceso.

4. En función del resultado obtenido al comparar  $a$  y  $b$  en el punto 3, el algoritmo sigue por un camino o por otro. Lo representamos con un símbolo de decisión.

5. Si  $a$  y  $b$  son mayores que 0, se multiplica  $a \times b$ . Como es una operación, la representamos con un símbolo de proceso.

6. El algoritmo ofrece el resultado de la multiplicación. Utilizamos el símbolo de salida de información.

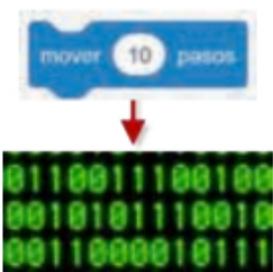
7. El algoritmo finaliza. Usamos otro símbolo de terminal.



## 4. Lenguajes de programación. Tipos

Ya has visto que los algoritmos están formados por un conjunto de acciones sencillas que dan respuesta a un problema concreto. Así, por ejemplo, el algoritmo de la suma nos permite obtener el resultado de esta operación entre varios números.

El proceso de resolución de problemas se agiliza y mejora si utilizamos los ordenadores para ayudarnos a desarrollar los algoritmos. Para ello, las instrucciones de los algoritmos se codifican utilizando lenguajes de programación, que son la forma en que los seres humanos pueden dar instrucciones a un equipo informático.



**Un lenguaje de programación** es un lenguaje que sirve para describir un conjunto de acciones que deben ser ejecutadas por las máquinas.

### EJEMPLO

Queremos que un gato que está en la pantalla se mueva una cierta distancia y luego diga “¡Hola!”.



Inicio

**Fase 1.** Diseñamos el algoritmo, descomponiendo todo el proceso en pequeños pasos y asignando en este caso las distancias y tiempos concretos:



**Fase 2.** Realizamos el programa, siguiendo el algoritmo predeterminado, con un lenguaje de programación (en este caso, Scratch).



Fin

**Fase 3.** El gato se mueve en la pantalla la distancia programada y dice “¡Hola!” durante el tiempo determinado.

### Tipos de lenguajes de programación

Hay diferentes criterios para clasificar los lenguajes de programación: según la **forma de ejecutarse** pueden ser compilados o interpretados; según la **forma de resolver los problemas**, los hay estructurados y orientados a objetos...

En este primer curso nos interesa que los diferencias según el **nivel al que se utilizan**, y en este caso podemos encontrar tres tipos de lenguajes:

- **Lenguaje máquina.** Es el lenguaje de programación que entiende directamente la computadora. Usa el alfabeto binario, es decir, el 0 y el 1.
- **Lenguajes de bajo nivel.** Se parecen más al lenguaje humano. El **lenguaje ensamblador** fue el primero que trató de sustituir el lenguaje máquina por uno de bajo nivel.
- **Lenguajes de alto nivel.** Están diseñados para que los programadores escriban y entiendan instrucciones lo más parecidas al lenguaje humano (normalmente, en inglés), lo cual hace que se necesite menos tiempo para aprender a programar (y por ello son los más utilizados).

## 5. Programación por bloques: Scratch

### Scratch

**Scratch** es un entorno de programación gráfico y gratuito desarrollado por un grupo de investigadores del Instituto Tecnológico de Massachusetts (MIT) bajo la dirección del Dr. Mitchel Resnick.

Sirve para crear juegos y animaciones, al mismo tiempo que se desarrollan habilidades de pensamiento lógico y se potencia la creatividad.

Podemos trabajar con una **versión descargada e instalada** en nuestro ordenador o bien con la **versión online** de la web <http://scratch.mit.edu>, que, si nos registramos, nos permitirá guardar en nuestro espacio virtual los proyectos que desarrollemos.

Como el propio menú superior de la web indica, Scratch está pensado para crear, explorar, comentar, ayudar y buscar proyectos compartidos.

La sintaxis de Scratch se basa en un conjunto de **bloques gráficos de programación** que se ensamblan para crear programas.

### Ventana de Scratch

Cuando abrimos el programa o accedemos a él a través de su web, nos encontramos la siguiente pantalla de programación:



### ¿Sabías que...?

La extensión de los archivos de Scratch es **.sb3**; la de los fondos creados con el programa, **.svg**; y la de los objetos, **.sprite3**.

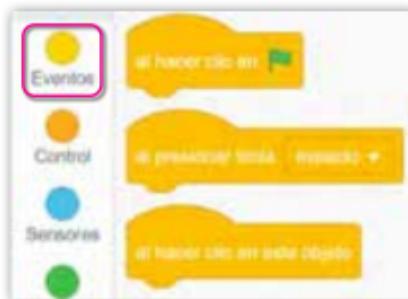


## Fundamentos de la programación por bloques

A continuación vamos a ver los **principales elementos** que se utilizan al programar por bloques. Entender cómo funcionan estos elementos es muy útil para hacer buenos programas, como veremos en las prácticas de Scratch de las páginas siguientes.

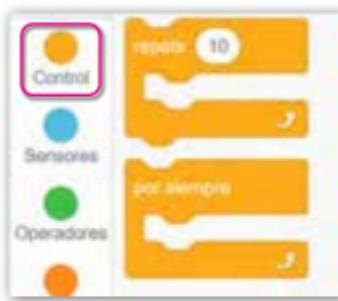
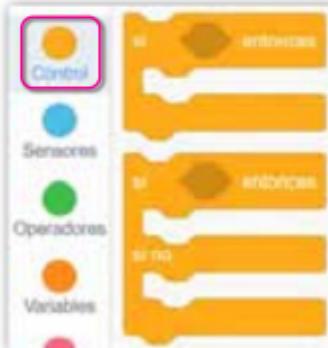
### Eventos

Los **eventos** son acciones que se usan para comenzar un programa o desencadenar una acción, como por ejemplo hacer clic en un objeto o presionar una tecla concreta.



### Estructuras de control

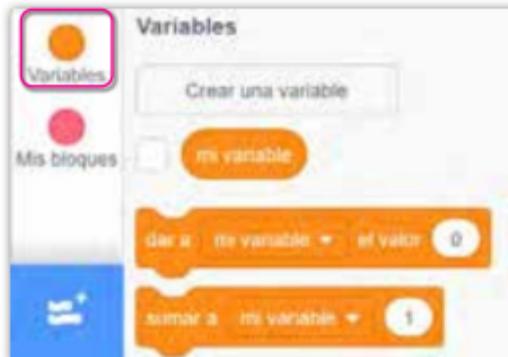
- **Bucles.** Un **bucle** es una secuencia de instrucciones en que se repite varias veces el mismo trozo de código mientras se cumpla una condición. Las instrucciones para crear bucles se encuentran en la categoría **Control**.
- **Condicionales.** Un **condicional** es una instrucción o un grupo de instrucciones que se pueden ejecutar o no, en función del valor de una condición.



### Variables

Una **variable** se puede entender como una caja en la que vamos almacenando datos. Por ejemplo, es habitual en los juegos que necesitemos almacenar el número de puntos o de preguntas acertadas; para ello necesitaremos usar variables.

Existen dos **tipos de variables**: las **globales**, que se pueden utilizar en cualquier objeto, y las **locales**, que sólo sirven para un objeto determinado.



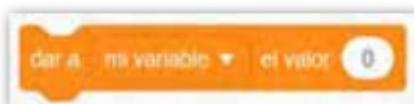
Al crear una nueva variable podemos elegir si será global o local.

## Operaciones con variables

En Scratch, además de utilizar las variables para las **operaciones aritméticas** básicas (suma, resta, multiplicación y división), tenemos una opción muy útil que nos sirve para **incrementar** o **decrementar** la variable. Por ejemplo, cuando en un juego conseguimos un punto, hacemos que se incremente automáticamente en 1 la variable **puntos**. Para ello utilizamos la instrucción **sumar a**.



Las variables también nos permiten **inicializar** un recuento, es decir, dar a la variable un valor inicial (que generalmente es 0). Por ejemplo, es habitual inicializar el número de puntos iniciales de un juego.



## Integración de gráficos y sonidos

Para utilizar gráficos y sonidos hay que añadir una extensión que se encuentra en la esquina inferior izquierda. Así podemos añadir extensiones como **Música**, **Lápiz**, etc.



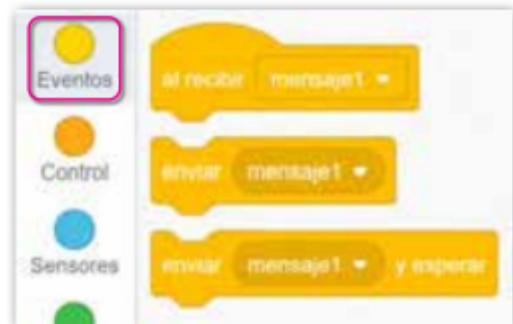
## Ejecución simultánea de varios objetos, clones y comunicación entre ellos

En muchas ocasiones necesitamos utilizar en nuestros programas varios objetos al mismo tiempo. Por eso es conveniente utilizar clones y aprender cómo se pueden comunicar entre sí los distintos objetos.

- **Clones.** Un **clon** es una copia idéntica de un objeto. Crear clones nos resulta útil cuando queremos que dos objetos sean iguales pero cada uno de ellos se comporte de forma diferente.



- **Comunicación entre objetos.** Para que los objetos se comuniquen entre sí utilizamos las instrucciones **enviar mensaje** y **al recibir mensaje**, de la categoría **Eventos**. Con ellas podemos hacer que el personaje 2 comience a hacer una serie de acciones cuando reciba un mensaje que le envía el personaje 1.



## 6. Análisis y validación de software

A lo largo del desarrollo de un programa informático es necesario realizar evaluaciones de su funcionalidad y verificar su adecuación como solución al problema planteado.

No todos los programas son igual de importantes, ni tampoco los fallos que puedan detectarse en ellos son de la misma relevancia. Así, por ejemplo, un programa informático que controle el proceso de producción de una vacuna o uno que monitorice a un paciente en un hospital son programas considerados críticos y deben ser sometidos a estrictos controles y procesos de verificación y validación antes de su puesta en funcionamiento.



**Los procesos de verificación** son llevados a cabo por los desarrolladores del programa y determinan la calidad del software a lo largo de todo su proceso de desarrollo. Con ellos se busca la detección y corrección temprana de posibles errores, lo cual permite evitar costes derivados de fallos del sistema.



**Los procesos de validación** determinan si el programa final cumple los requerimientos planteados por el cliente, incluyen pruebas de resultados y valoran la integración del software con el hardware en el que se implementará.

Ambos procesos garantizan la calidad del software creado.

### EVITA ERRORES EN TUS PROGRAMAS DE SCRATCH

The Scratch script for 'buzo1' contains several errors:

- Initializations:** It initializes variables 'buzo1' and 'buzo2' at the start of the script.
- Control Flow:** It has a loop that runs until it touches a 'Jellyfish' sprite.
- Variables:** Inside the loop, it changes variable 'buzo1' to 1 and then back to 0.
- Comments:** It includes a comment 'Iniciizo la variable'.
- Output:** It says 'Soy buzo 1 y he tocado a la medusa' for 2 seconds.
- Sound:** It plays a sound 'señalar'.

**Common mistakes highlighted:**

- Inicializa todas las variables al comienzo de cada programa.** (Initializes all variables at the beginning of each program.)
- Pon nombres significativos a los personajes, los escenarios y las variables.** (Give meaningful names to characters, scenes, and variables.)
- Revisa bien que no des órdenes contradictorias.** (Check well that you do not give contradictory orders.)
- Incluye comentarios que expliquen los procesos que se llevan a cabo.** (Include comments that explain the processes carried out.)
- Incluye instrucciones que muestren los valores que van tomando las variables a lo largo del programa, para poder revisar tu programa más fácilmente.** (Include instructions that show the values that variables take along the program, so you can easily review your program.)

Para Scratch, existe una herramienta online, llamada **Dr. Scratch**, con la que puedes evaluar tus programas (<http://drscratch.org>). Puedes subir uno que tengas descargado en tu ordenador o enlazar el que tengas abierto en la web de Scratch. Esta herramienta muestra la valoración del programa en relación con diversos aspectos del pensamiento computacional, como el pensamiento lógico, el control de flujo o la abstracción, entre otros.



Escanea este código para acceder a la web Dr. Scratch y evaluar alguno de tus programas.

## 7. Software libre y software propietario

Aunque puedan parecer prácticas lícitas, no está permitido instalar un programa en varios ordenadores con una única licencia, ni hacer copias de programas sin permiso del autor ni distribuir por la Red de forma masiva programas protegidos por copyright (por ejemplo, videojuegos).

Todos los programas informáticos están protegidos por la **ley de propiedad intelectual**, que reconoce al autor del software el derecho exclusivo a la explotación de su obra en cualquier forma, y en especial los derechos de reproducción, distribución, comunicación pública y transformación, que no podrán ser realizadas sin su autorización. Dicho de otro modo, es el autor de cada programa informático quien decide cómo se explota su obra.

Conforme a esta potestad, el software puede ser catalogado como software libre o como software propietario.

- El **software libre** incluye los programas informáticos que se distribuyen sin restricciones atendiendo a cuatro libertades fundamentales: libertad de uso, de distribución, de copia y de modificación.

Ejemplos de software libre son el paquete LibreOffice y el sistema operativo Linux.

En algunas ocasiones, los programas informáticos distribuidos como software libre están protegidos por copyleft (en oposición a copyright). La licencia **copyleft** exige que cualquier modificación hecha sobre un programa de software libre sea liberada para el resto de los usuarios con las mismas libertades que el programa original.

- El **software propietario (o privativo)** incluye los programas que se distribuyen con limitación en una o varias de las cuatro libertades fundamentales.

Los usuarios que adquieren un programa distribuido como software propietario tienen que comprobar qué libertad les otorga el autor sobre el programa adquirido.

Ejemplos de software propietario son el sistema operativo Windows, las herramientas de ofimática de Microsoft, la aplicación WhatsApp o el videojuego Fortnite.



**Las cuatro libertades fundamentales del software libre son:**

**1** Libertad de uso del programa con cualquier propósito

**2** Libertad de distribución de copias del programa a terceros

**3** Libertad de estudio y modificación del software

**4** Libertad de copia y distribución de copias modificadas

### ¿El software libre es gratuito y el software propietario es de pago?

No siempre. La mayoría del software libre se distribuye gratuitamente, pero hay ocasiones en las que esta distribución puede incluir algún extra por el que se cobra, como por ejemplo la instalación, la asistencia técnica o cualquier otra mejora de funcionalidad para el usuario.

Tampoco el software propietario es siempre de pago. Lo puedes comprobar revisando las aplicaciones gratuitas que utilizas en tu teléfono móvil habitualmente, como WhatsApp, Shazam, Adobe Scan, etc.

## Práctica 1. Resolución de problemas. Infografía con Canva

La resolución de problemas es un proceso cíclico: los pasos para resolverlos se repiten una y otra vez hasta encontrar la solución idónea.

Utilizando el pensamiento computacional, ¿qué debes hacer para resolver un problema?



### 1. Comprende el problema

Lee el enunciado del problema varias veces hasta conocer la incógnita, los datos y los condicionantes que tienes que respetar.



### 2. Establece un plan de acción

Descarta los datos innecesarios o irrelevantes y decide qué ecuaciones vas a utilizar. Si el problema es complicado, divídelo en partes más pequeñas fácilmente resolubles.



### 3. Desarrolla el plan. Obtén una solución

Sigue detalladamente el plan que has planteado. Simplifica siempre que sea posible y haz dibujos o diagramas para presentar las soluciones.



### 4. Revisa la solución obtenida

La solución al problema planteado tiene que ser completa, esto es, debe dar respuesta a todas las incógnitas. Por otra parte, no basta con llegar a una solución del problema, es necesario confirmar que la solución es la adecuada. Para confirmarlo, intenta llegar a ella por otro camino o estimar con antelación si es buena.



Fig. 2

En esta práctica vamos a crear una infografía en la que mostraremos los pasos de la resolución de problemas. Utilizaremos **Canva**, una aplicación muy útil para elaborar murales e infografías, de uso libre.

Para hacer esta práctica deberás tener el consentimiento de tus padres o tutores legales. Asimismo, recuerda leer detenidamente todas las condiciones de prestación de servicios.

1. Entra en [www.canva.com](http://www.canva.com). La primera vez que accedes a esta página, la aplicación te pide que te registres. Puedes hacerlo con tu cuenta educativa de correo.
2. Una vez que has accedido, observa que aparecen distintos diseños con los que crear tarjetas, pósters, presentaciones, etc. (figura 1). Haz clic en la opción **Infografía**.

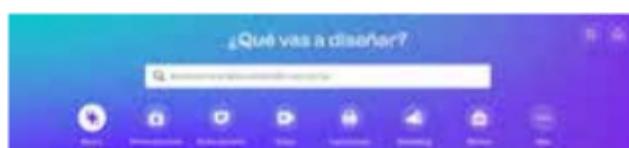


Fig. 1

3. Entre las plantillas que aparecen, escoge la que se llama **Infografía proceso dibujo a mano amarillo gris y negro** u otra que te guste a ti (figura 2).
4. Haz doble clic en el título de la infografía, "Infografía de procesos", para modificarlo, y escribe en su lugar "Pasos para resolver problemas".

5. Ahora, en cada uno de los globos de color amarillo, haz doble clic para poder modificar su contenido. Escribe en ellos las cuatro fases para resolver un problema que hemos visto antes.
6. En los globos blancos puedes incluir alguna información de interés relacionada con cada una de las fases de la resolución de problemas. No añadas textos muy largos, las infografías tienen que ser muy visuales.
7. Elimina el resto de los textos que aparecen en la plantilla y que no resulten de interés. Para eliminar un texto, haz clic sobre él, comprueba que se selecciona la caja del texto y después haz clic con el botón derecho sobre la caja del texto y pulsa **Eliminar** (figura 3).
8. Elimina también todos los dibujos o fotos que no quieras que aparezcan en la infografía. Sigue para ello las indicaciones del paso anterior.
9. Selecciona ahora un globo blanco haciendo clic sobre él. Verás que aparece una barra de herramientas en la parte superior para modificar el formato del texto (figura 4). Ponle como tamaño de letra **25** para que se vea mejor. Luego haz esto mismo en los demás globos blancos. Puedes modificar de la misma forma el color de los textos y el tipo de letra, si te parece que quedará todo mejor.

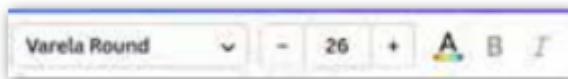


Fig. 4

10. En la barra de herramientas lateral de la aplicación, haz clic en **Más** y selecciona **Fotos**. Aquí puedes buscar fotos para tu infografía. Escribe “solución” en el buscador del menú **Fotos** (figura 5) y verás que el programa te ofrece muchas imágenes relacionadas. Escoge una que te guste y haz clic en ella. Comprueba que la imagen se sitúa sobre tu infografía.
11. Pincha la imagen y arrástrala para colocarla en la parte final de la infografía. Puedes ajustar su tamaño haciendo clic en los círculos de las esquinas y arrastrando, para hacerla mayor o menor, según necesites.
12. Tu infografía deberá quedar parecida a la de la figura 6.
13. Una vez terminada, ve a la barra de herramientas superior y haz clic en **Descargar** para bajarte la infografía en PDF a tu ordenador. Selecciona la opción **Descarga un borrador gratis con marca de agua**.
14. Accede al borrador descargado, haz clic sobre él con el botón derecho y cambia su nombre por **UD01\_P1\_nombreapellido**. Guarda el documento en tu carpeta de prácticas.



Fig. 3

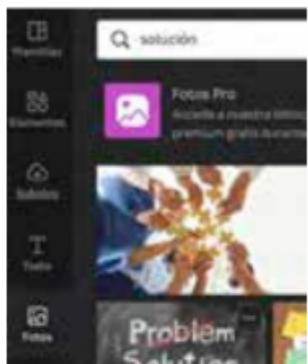


Fig. 5



Fig. 6

### DESAFÍO

- Crea con Canva un cartel en tamaño A4 para la clase de Programación, Inteligencia Artificial y Robótica. Puedes ponerle un fondo de color y usar letras vistosas para el nombre de la asignatura. Adorna el cartel con imágenes relacionadas con la asignatura. Luego descárgatelo y guárdalo con el nombre **UD01\_P1\_nombreapellido\_desafio1.pdf**.

## Práctica 2. ¿Cómo se hacen los diagramas de flujo?



En esta práctica vamos a aprender las reglas básicas para elaborar diagramas de flujo. Utilizaremos para ello el programa **LibreOffice Draw**.

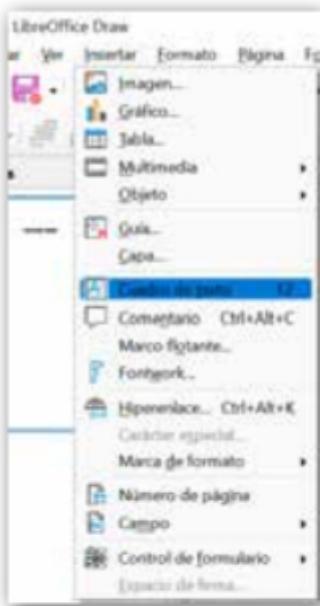


Fig. 7

1. Abre el programa yendo a **Inicio / Todos los programas / LibreOffice Draw**. Ve a **Archivo** y escoge **Guardar como**. Llama al archivo **UD01\_P2\_nombreadellido.odg**.
2. Ve al menú **Insertar** y elige **Cuadro de texto** (figura 7).
3. Haz clic en la parte superior de la hoja y escribe “¿Cómo se hacen los diagramas de flujo?”.
4. Pon a este título tamaño **18** y color **azul** (figura 8).

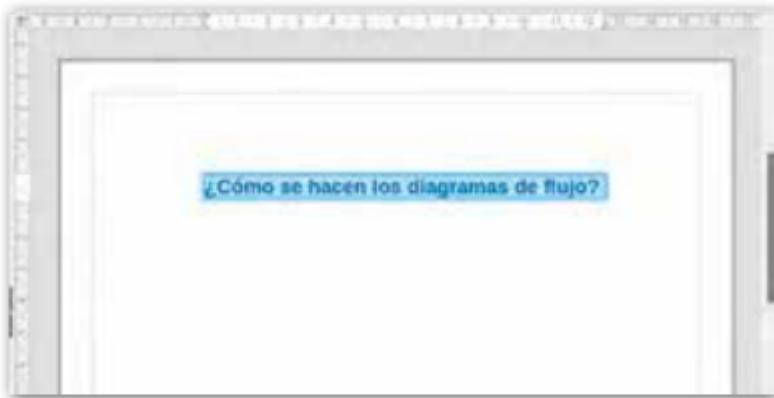


Fig. 8



Fig. 9

5. A continuación, inserta otro cuadro de texto y escribe en él la primera norma sobre cómo hacer diagramas de flujo: “1. Todo diagrama de flujo debe tener un principio y un fin”. Ponle al texto color **negro** y tamaño **12**.
6. En la barra lateral derecha, haz clic en el ícono **Galería** (si no ves en tu pantalla esa barra lateral, ve al menú **Ver** y elige **Galería**; de ese modo te aparecerá a la derecha lo que se muestra en la figura 9).
7. Selecciona **Diagrama de flujo**. Escoge el símbolo **Start/End** y arrástralo hasta la hoja, debajo del texto. A continuación, arrastra el símbolo de fin, que se llama **Terminator**.
8. De momento, tu documento debe quedar como en la figura 10:

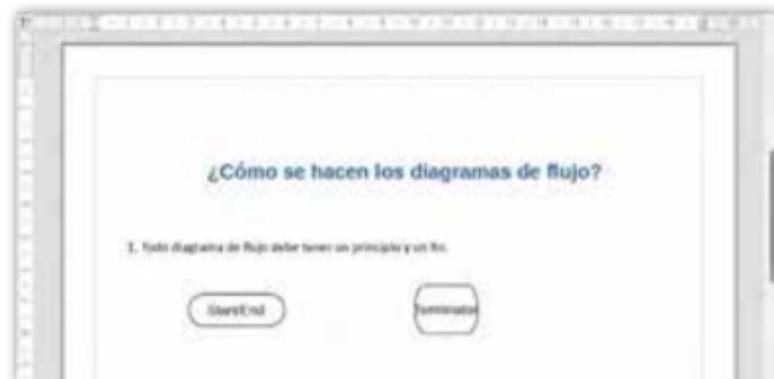


Fig. 10

9. Haz doble clic sobre el texto "Start/End" del primer símbolo y cámbialo por "Inicio". Cambia en el segundo símbolo la palabra "Terminator" por "Fin".
10. Inserta otro cuadro de texto y escribe la segunda norma sobre cómo hacer diagramas de flujo: "2. Las flechas deben ser horizontales o verticales y deben estar conectadas a un símbolo".
11. Inserta dos cuadros de texto. En uno escribe "BIEN", y en el otro, "MAL". A continuación, arrastra las flechas y símbolos necesarios para que te queden como en la figura 11:

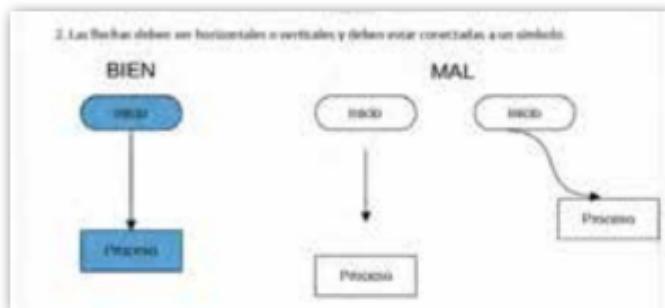


Fig. 11

12. Colorea el diagrama correcto utilizando la herramienta **Color de relleno** (el bote de pintura que te mostramos en la figura 12). Cambia de nuevo "Start/End" por "Inicio" en todos los bloques, así como "Process" por "Proceso".
13. Repite estos pasos para completar la hoja tal como aparece en la figura 13.

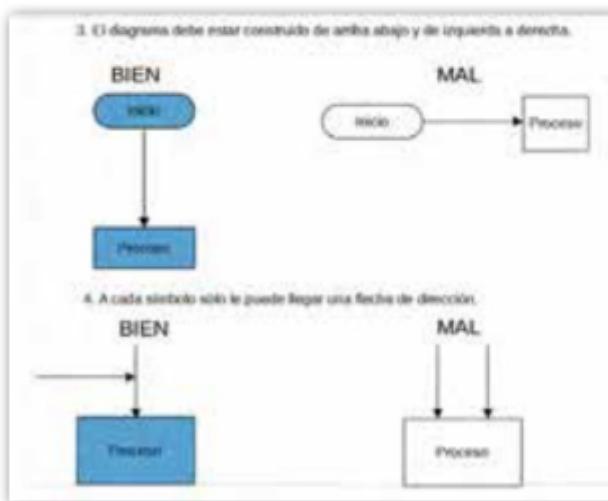


Fig. 13

14. Cuando hayas acabado, ve al menú **Archivo** y escoge **Guardar**. Después, ve de nuevo a **Archivo** y escoge **Exportar / Exportar a PDF**.



### DESAFÍO

- En un nuevo documento de Draw, y siguiendo el modelo de esta práctica, haz un trabajo en el que muestres todos los símbolos utilizados en los diagramas de flujo, con su nombre correspondiente (los hemos visto en la parte de teoría de la unidad). No olvides poner un título al principio del documento. Guarda el archivo como **UD01\_P2\_nombreadellido\_desafio1.odg** y expórtalo también a PDF.



Fig. 12



En tu espacio web tienes el **documento completo** de esta práctica, con el nombre **UD01\_P2\_modelo.pdf**.

## Práctica 3. Traducir una receta de cocina a un algoritmo

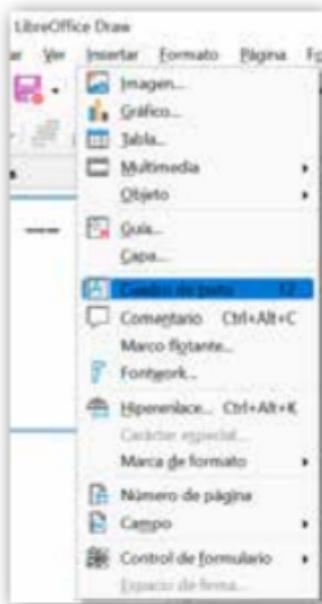


Fig. 14

Imagina que tienes una receta de cocina y quieres preparar ese plato. Lo mejor es que los pasos de la receta sean sencillos, precisos, finitos y secuenciados.

Vamos a practicar esto aquí a partir de una receta sencilla y además elaboraremos su diagrama de flujo.

Aquí tienes la receta:

### Receta para hacer un huevo pasado por agua

Cogemos un cazo pequeño. Lavamos el huevo con agua y lo metemos en el cazo. Ponemos agua fría en el cazo hasta que cubra por completo el huevo. Echamos sal en el agua para que no se agriete la cáscara. Ponemos el cazo en el fuego. Cuando el agua comience a hervir a borbotones, esperamos tres minutos, si nos gusta la clara poco cuajada, o cuatro minutos, si queremos que la clara esté cocida y la yema esté líquida. Removemos durante el primer minuto para que la yema del huevo se quede en el centro. Sacamos el huevo y lo ponemos en agua fría. ¡Ya está listo para comer!

1. Abre el programa LibreOffice Draw. Ve a Archivo y escoge **Guardar como**. Llama al archivo **UD01\_P3\_nombreadellido.odg**.
2. Ve al menú **Insertar** y elige **Cuadro de texto** (figura 14).
3. Haz clic en la parte superior de la hoja y escribe "Cómo hacer un huevo pasado por agua".
4. Para que no nos quede un diagrama de flujo muy grande, vamos a descomponerlo en dos: primero la preparación y después cómo cocinarlo.
5. Realiza las operaciones necesarias para completar los primeros elementos de la figura 15:

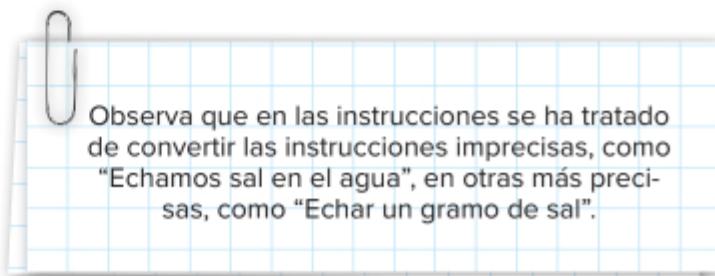
### Cómo hacer un huevo pasado por agua

#### 1. Algoritmo de preparación

Inicio

Fig. 15

6. Vamos a pensar ahora cuáles serían las tareas de preparación, leyendo la receta, así como su orden. Completa el primer diagrama con los pasos que consideres y siguiendo la receta; además, intenta que sean lo más precisos posible. En la figura 16 te proponemos una solución.



Observa que en las instrucciones se ha tratado de convertir las instrucciones imprecisas, como "Echamos sal en el agua", en otras más precisas, como "Echar un gramo de sal".



Fig. 16

7. Ve al menú **Página** y elige **Página nueva** (figura 17).
8. En la segunda hoja, ve al menú **Insertar** y elige **Cuadro de texto**. Haz clic en la parte superior de la hoja y escribe “2. Algoritmo de pasar el huevo por agua”.
9. Lee de nuevo la receta y ve pensando en los distintos pasos. Confecciona el diagrama de flujo correspondiente. Observa que vas a necesitar una estructura selectiva como la de la figura 18.

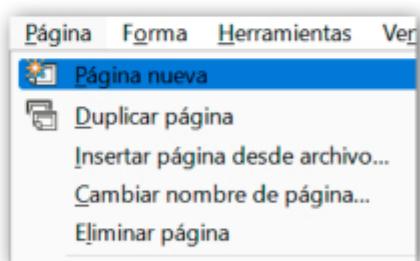


Fig. 17

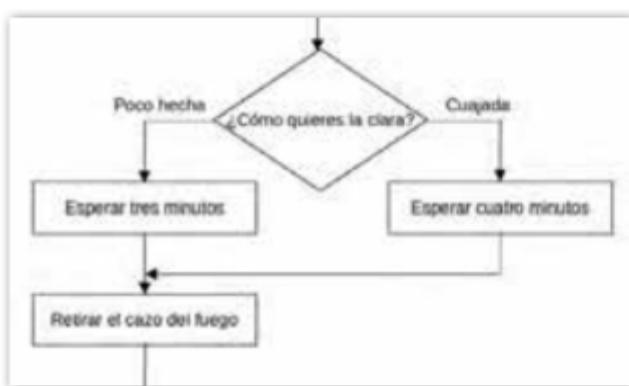


Fig. 18

10. En la figura 19 te proponemos una solución. Observa la precisión de las instrucciones y el orden en que se han colocado.
11. Cuando hayas acabado, ve al menú **Archivo** y escoge **Guardar**. Después, ve de nuevo a **Archivo** y escoge **Exportar / Exportar a PDF**.



En tu espacio web tienes el **documento completo** de esta práctica, con el nombre **UD01\_P3\_modelo.pdf**.

## 2. Algoritmo de pasar el huevo por agua

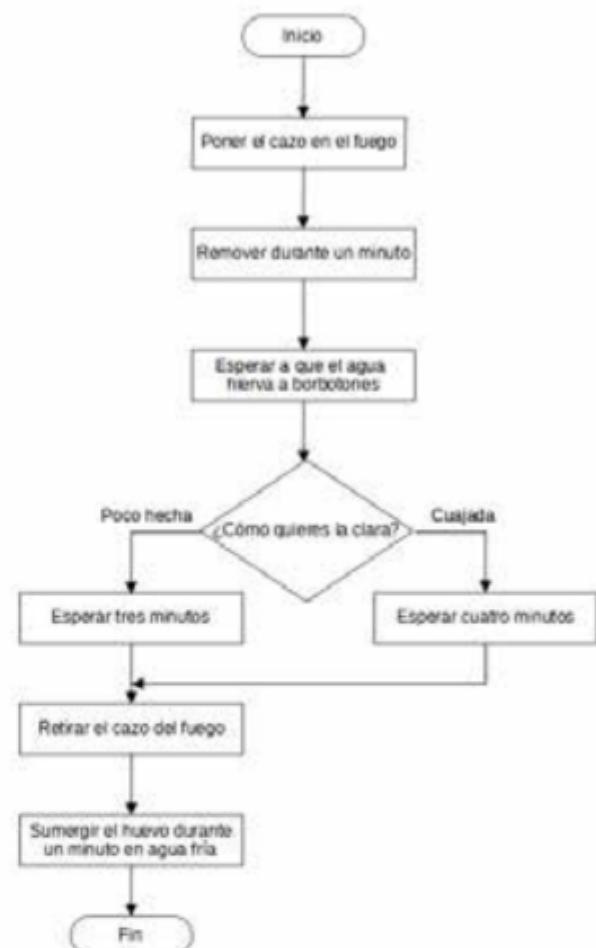


Fig. 19



### DESAFÍO

- A continuación tienes las instrucciones para hacer un huevo frito:

#### Receta desordenada para hacer un huevo frito

- Golpear el huevo contra la sartén hasta partir su cáscara
- Sacar el huevo de la sartén cuando esté al gusto
- Echar un poco de aceite en la sartén
- Esperar a que se caliente el aceite
- Lavarse las manos
- Poner la sartén en el fuego
- Encender el fuego
- Echar el huevo en el aceite
- Echar sal al gusto
- Remover el aceite hacia el huevo

Sigue estos pasos:

- Piensa cómo transformar cada una de estas instrucciones de forma que sean lo más precisas posible.
- Ordénalas de modo lógico.
- Abre un archivo de dibujo y elabora el diagrama de flujo con las instrucciones ordenadas. No hace falta que partas de cero, puedes reformar el diagrama que has hecho en esta práctica. Guárdalo como **UD01\_P3\_nombreapellido\_desafio1.odg** y expórtalo también a PDF.

## Práctica 4. Depurar algoritmos. Esquema de pasos con Padlet

Para hacer esta práctica deberás tener el consentimiento de tus padres o tutores legales.

En ocasiones, los algoritmos que proponemos como solución a un problema no son correctos, porque incluyen instrucciones poco precisas, pasos innecesarios o instrucciones repetidas.

En esta práctica vamos a crear con **Padlet** una lista de pasos ordenados para revisar los algoritmos que hayamos creado.

1. Abre el navegador y entra en <https://es.padlet.com>. Haz clic en **Registrarse** y regístrate con tu cuenta de correo educativa.
2. En la pantalla que aparece, pulsa **Hacer un padlet** (figura 20).
3. En la pantalla siguiente, pulsa **Seleccionar** en la opción **Lienzo**.
4. Haz doble clic sobre el título que te ha asignado Padlet y te aparecerá la pantalla de la figura 21, donde puedes cambiar el título y la descripción. Pon “Cómo depurar algoritmos” como título y “Pasos ordenados” como descripción. Modifica también el fondo de pantalla yendo a **Fondo de pantalla / Texturas y formas** y eligiendo el que prefieras.
5. Pulsa **Guardar** y luego **Cerrar**. Ahora haz doble clic en el papel tapiz de la pantalla, para empezar a publicar entradas. En **Asunto**, como título de la primera entrada, escribe “1. Leer el algoritmo” y en el espacio inferior pon “Enseña el algoritmo a una persona distinta de ti para que lo lea detenidamente” (figura 22). Haz clic en el icono de la **búsqueda de imágenes** y luego en **Web** y sube una imagen relacionada que encuentres en Internet. Puedes buscar en bancos de imágenes como [www.freepik.es](http://www.freepik.es), no importa que tengan marca de agua cuando las descargas. Haz clic en **Publicar**.
6. Repite el proceso indicado en el punto anterior para publicar dos entradas nuevas. Copia para cada una de ellas el título y la descripción que tienes en la figura 23.
7. Una vez creadas las tres entradas, haz clic en los tres puntos que puedes ver en la parte superior derecha de la primera entrada que has creado. En el menú que aparece, escoge **Conectarse a una publicación** y haz clic en el botón **Conectar** de la segunda entrada.
8. Repite el proceso anterior para conectar la entrada 2 con la 3. Y haz lo mismo para conectar la entrada 3 con la 1. Al unir las entradas así (figura 23), representamos un proceso cíclico como es el de la depuración de algoritmos, que se lleva a cabo varias veces hasta que ya no se detecten errores.
9. En la barra de herramientas superior, haz clic en **Compartir** y luego en **Copiar el enlace en el portapapeles**. Pega el enlace en un documento de texto y guárdalo como **UD01\_P4\_nombreapellido**.



### DESAFÍO

- Investiga sobre las licencias de software freeware, shareware, GPL y MPL y elabora otro lienzo de Padlet con información sobre ellas. Pega el enlace en un documento de texto y guárdalo como **UD01\_P4\_nombreapellido\_desafio1**.



+ HACER UN PADLET

Fig. 20



Fig. 21



Fig. 22

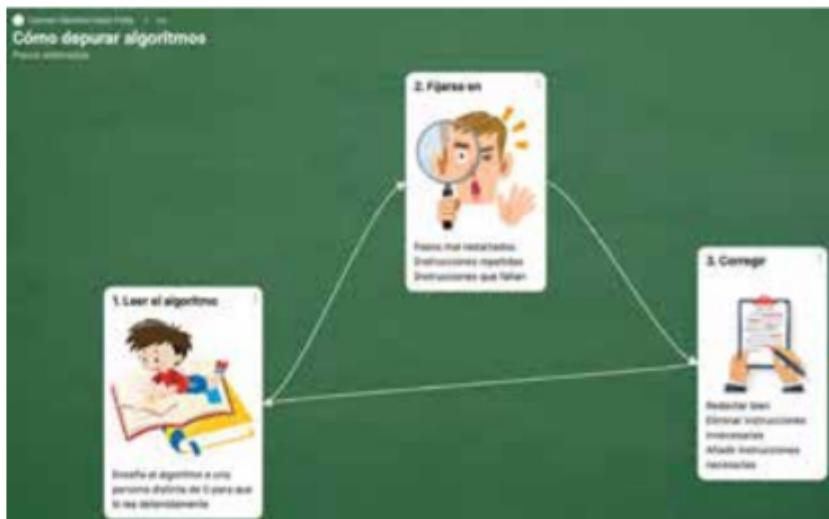


Fig. 23

## Práctica 5. Juego con algoritmos. La Hora del Código

Puedes encontrar muchos juegos para practicar lo aprendido en esta unidad y seguir aprendiendo a pensar como un informático.

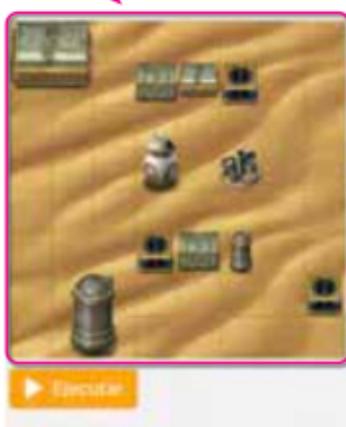
**La Hora del Código** es una iniciativa que surgió en el año 2013 con la idea de potenciar el pensamiento computacional en la escuela realizando actividades de programación durante una hora. La web de La Hora del Código incluye multitud de actividades para aprender a programar (para todos los niveles) utilizando algoritmos y comprobando su funcionamiento.

Para hacer esta práctica deberás tener el consentimiento de tus padres o tutores legales.

### Ejercicio 1. Star Wars

1. Accede a la web <https://hourofcode.com/es>. Haz clic en **Inténtalo** y se abrirá una nueva página en la que hay muchos juegos de práctica.
2. Busca el juego **Star Wars: crea una galaxia a través del código** y haz clic en su imagen y luego en **Inicio**. La URL de este juego es <https://code.org/starwars>.
3. Este juego te permite jugar de dos maneras: con bloques o con instrucciones de JavaScript. Elige **Bloques**.
4. Puedes visualizar el video introductorio para aprender cómo funciona el juego.
5. Trata de completar todos los niveles del juego. Presta atención.
6. Tras completar el último nivel, haz una captura de pantalla del programa que has realizado, haciendo clic en la tecla **Impr Pant** de tu teclado. Luego abre un documento nuevo en un procesador de texto y pega la captura. Guarda la práctica como **UD01\_P5\_E1\_nombreadellido**.

Una vez creado el programa con el conjunto de bloques, haz clic en **Ejecutar**. En esta zona comprobarás si el algoritmo indicado a BB-8 es correcto y consigue todos los objetivos.



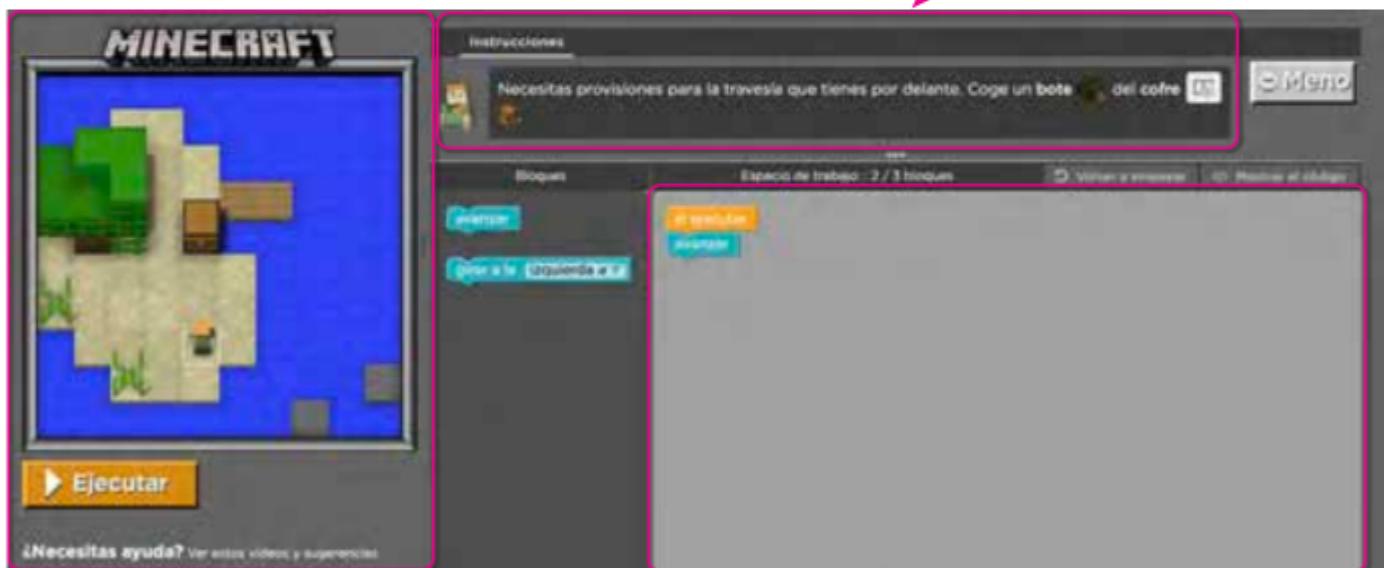
El personaje te indica qué tienes que conseguir en cada caso.

En esta parte de la imagen, arrastra los bloques de instrucciones en el orden en que quieras que BB-8 las ejecute. Si te equivocas, sólo tienes que seleccionar la instrucción equivocada y arrastrarla hasta el conjunto de instrucciones de la izquierda para eliminarla.

## Ejercicio 2. Minecraft

1. En esta ocasión vas a jugar a Minecraft. En la página de La Hora del Código, y de la misma forma que en el ejercicio anterior, entra en el juego **Voyage Aquatic de Minecraft**, que está dentro de **Hora del Código de Minecraft**. La URL de este juego es <https://studio.code.org/s/aquatic/lessons/1/levels/1>.
2. Visualiza el vídeo explicativo. En esta ocasión las instrucciones de juego te las explica un personaje de Minecraft.
3. Despues de ver el video, escoge qué personaje quieras ser: Steve o Alex. El funcionamiento de este juego es muy similar al anterior, pero los bloques de código son diferentes.
4. Tras completar el último nivel, haz una captura de pantalla del programa que has realizado, haciendo clic en la tecla **Impr Pant** de tu teclado. Luego abre un documento nuevo en un procesador de texto y pega la captura. Guarda la práctica como **UD01\_P5\_E2\_nombre-apellido**.

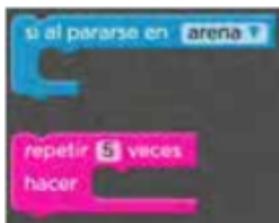
El personaje te indica qué tienes que conseguir en cada caso. En este juego te aparecerá Alex o Steve, según tu elección.



Comprueba en esta zona la exactitud del algoritmo que has realizado, haciendo clic en **Ejecutar**.

Arrastra los bloques de instrucciones en el orden en que quieras que sean ejecutados.

Este juego incluye bloques con instrucciones iterativas y selectivas en las que hay que anidar otras instrucciones. Por ejemplo:

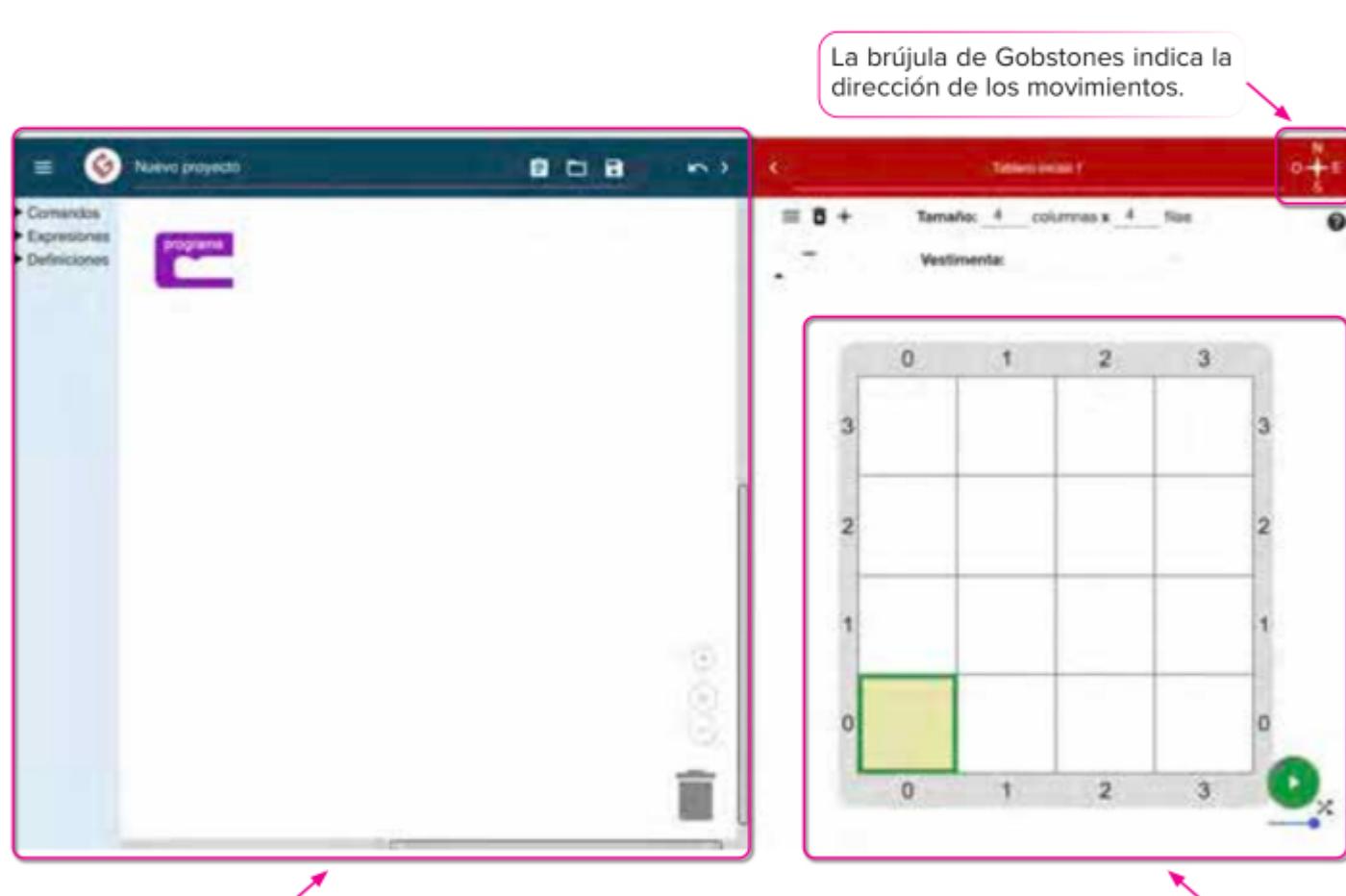


## Práctica 6. Tus primeros programas. Gobstones

**Gobstones** es un lenguaje de programación desarrollado por un equipo de la Universidad Nacional de Quilmes (Argentina) y diseñado para enseñar a programar. En esta práctica usaremos el entorno de programación júnior.

En Gobstones se programa para cuatro bolitas de estos colores: azul, rojo, verde y negro.

1. Accede a la web <http://gobstones.github.io> y haz clic en **¡Probá Gobstones Web ahora!** Elige a continuación **Gobstones Jr** para entrar en su entorno de programación.
2. Te aparecerá una ventana en la que abrir algún ejercicio que tengas guardado. Ciérrala para empezar el primer ejercicio y comprueba que accedes a una pantalla como la de la imagen.



En esta parte de la pantalla aparecen las instrucciones disponibles para trabajar en Gobstones. En la parte superior están los iconos que permiten abrir proyectos guardados y guardar los que hagas.

Para crear un programa debes arrastrar instrucciones desde la izquierda hasta la parte central y colocarlas unidas unas debajo de las otras. Si quieras eliminar una instrucción, debes arrastrarla hasta la papelera.

La brújula de Gobstones indica la dirección de los movimientos.

En el tablero de Gobstones vemos el resultado del programa realizado cuando pulsamos el ícono de reproducción de la esquina inferior derecha.

Si el programa no es adecuado para el problema que hay que resolver, aparecerá en pantalla un mensaje de error que simula una explosión.

### • Ejercicio 1. Situar cuatro bolas de color en la primera casilla del tablero

1. En la barra de menús de la izquierda, haz clic en **Comandos** y luego en **Comandos primitivos**. Selecciona el bloque **Poner** y anídale dentro del bloque **programa**.
2. En la misma barra de menús, haz clic en **Expresiones** y luego en **Literales**. Selecciona el bloque **Azul** y colócalo dentro de la instrucción **Poner**.
3. Sitúa el cursor sobre la instrucción **Poner** y haz clic con el botón derecho (figura 24). Selecciona **Duplicar** para crear una copia de esta instrucción. En la copia, haz clic en la flecha del desplegable que aparece junto a la palabra "Azul" y elige otro color (por ejemplo, **Verde**). Anida esta instrucción debajo de la que has hecho en el paso 2, dentro de **programa**.
4. Repite el paso 3 dos veces más hasta tener una instrucción **Poner** para el color **Rojo** y otra para el color **Negro**. Anídalas unas debajo de las otras (figura 25).
5. Haz clic en el botón de reproducción y comprueba que aparecen cuatro bolas, cada una de un color, en la celda inferior izquierda del tablero.
6. En la parte superior de la pantalla, cambia el título **Nuevo proyecto** por **UD01\_P6\_E1\_nombreadellido** y haz clic en el ícono **Guardar** . Se descargará un archivo que puedes guardar en tu carpeta de prácticas.



Fig. 24



Fig. 25

### • Ejercicio 2. Mover bolas por el tablero

Vamos a modificar el programa anterior para que se sitúe una bola de cada color en cada una de las cuatro esquinas del tablero.

1. Abre el programa que has hecho en el ejercicio anterior haciendo clic en el ícono **Abrir** de la barra de herramientas superior.
2. En el menú **Comandos / Comandos primitivos**, elige la instrucción **Mover**.
3. En el menú **Expresiones / Literales**, escoge el bloque **Norte** y síntalo dentro de la instrucción **Mover**. De esta forma indicamos el sentido de desplazamiento de las bolas.
4. Anida la instrucción **Mover Norte** que acabas de crear debajo del bloque **Poner Azul**.
5. Crea otras dos instrucciones **Mover Norte** y anídalas en el bloque **Poner Azul** para mover la bola azul hasta la esquina superior izquierda.
6. De la misma forma, pon tres instrucciones **Mover** debajo de la segunda instrucción **Poner**. En este caso, la dirección será **Este** (observa la brújula) para llevar hasta la esquina superior derecha la bola verde.
7. Finalmente, pon otras tres instrucciones **Mover** debajo de la tercera instrucción **Poner**. Como dirección, elige en este caso **Sur** para trasladar la bola negra hasta la esquina inferior derecha.
8. Tendrás un programa como el de la figura 26.



Fig. 26



Fig. 27

9. Haz clic en el botón de reproducción y comprueba que aparecen cuatro bolas, cada una de un color, en las cuatro esquinas del tablero.
10. Guarda el proyecto como **UD01\_P6\_E2\_nombreadellido** en tu carpeta de prácticas.

### Ejercicio 3. Instrucción “Repetir”

1. Abre el programa del ejercicio anterior desde la barra de herramientas superior.
2. Ahora vamos a mejorar el programa que has hecho en el ejercicio 2, utilizando comandos de repetición.
3. En el menú **Comandos / Repeticiones**, escoge el bloque **repetir ... veces** y arrástralo hasta la zona de programación de Gobstones.
4. En el menú **Expresiones / Literales**, escoge el bloque **0** y colócalo dentro de la instrucción **repetir**. Cambia el **0** por un **3**.
5. Anida dentro del bloque **repetir** una de las tres instrucciones **Mover** que has utilizado para desplazar la bola azul y elimina las otras dos.
6. Coloca la instrucción **repetir** debajo de la instrucción **Poner Azul**.
7. Repite los pasos necesarios para crear una instrucción **repetir** para cada bola y colócalas debajo de las instrucciones **Poner** correspondientes.
8. Tendrás un programa como el de la figura 27.
9. Haz clic en el botón de reproducción y comprueba que, al igual que en el ejercicio 2, aparecen las cuatro bolas, cada una de un color, en las cuatro esquinas del tablero.

10. Guarda el proyecto como **UD01\_P6\_E3\_nombreadellido** en tu carpeta de prácticas.

### Ejercicio 4. Definir procedimientos

Una utilidad del programa Gobstones es la de poder definir procedimientos propios para usarlos fácilmente en el programa. Así, en vez de volver a escribir un conjunto largo de instrucciones, podemos simplemente programar un procedimiento.

En este caso vamos a crear un procedimiento para colocar las bolas de colores en las esquinas del tablero, aprovechando el programa que has realizado en el ejercicio anterior.

1. Abre el programa del ejercicio anterior desde la barra de herramientas superior.
2. En el menú **Definiciones / Procedimientos**, escoge el bloque **Define que Hacer algo** y arrástralo hasta la zona de programación de Gobstones.
3. Anida todas las instrucciones del programa del ejercicio anterior (excepto la instrucción **programa**) dentro de la instrucción **Define que Hacer algo**.



Fig. 28

4. Cambia el texto "Hacer algo" por "Crear cuadros con bolas de colores".
5. Tendrás un programa como el de la figura 28.
6. Accede ahora al menú **Comandos / Mis procedimientos** y comprueba que aparece ahí el procedimiento que acabas de crear. Arrástralо hasta la instrucción **programa** (figura 29) y comprueba su funcionamiento haciendo clic en el botón de reproducción.
7. Guarda el proyecto como **UD01\_P6\_E4\_nombreapellido** en tu carpeta de prácticas.

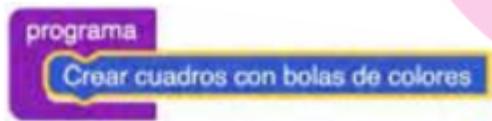


Fig. 29

### Ejercicio 5. Alternativas

1. Abre un proyecto nuevo desde la barra de herramienta superior.
2. Copia el programa de la figura 30 en la zona de programación de Gobstones. Para encontrar el bloque de instrucciones **si ...**, accede al menú **Comandos / Alternativas**.
3. Haz clic en el icono de reproducción y comprueba qué ocurre.
4. Guarda el proyecto como **UD01\_P6\_E5\_nombreapellido** en tu carpeta de prácticas.
5. Finalmente, contesta en tu cuaderno las siguientes preguntas:
  - a) ¿Cuántas bolitas azules indica el programa que deben aparecer?
  - b) ¿Por qué aparecen sólo dos bolitas azules?
  - c) Explica el funcionamiento del bloque de instrucciones **si ...** en este programa.

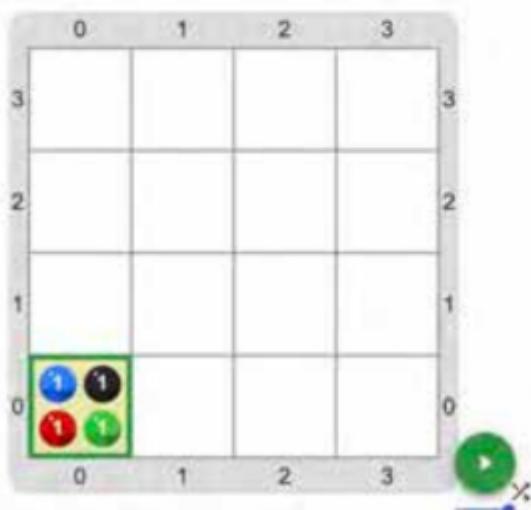


Fig. 31. El tablero de Gobstones, con las cuatro bolitas de colores en la celda inferior izquierda (0,0)



Fig. 30

### DESAFIOS

1. Crea con Gobstones un programa que coloque una bola azul en cada una de las celdas de una columna del tablero. Guarda el proyecto como **UD01\_P6\_nombreapellido\_desafio1** en tu carpeta de prácticas.
2. Crea con Gobstones un programa que coloque tres bolas verdes dibujando un triángulo en el tablero. Guarda el proyecto como **UD01\_P6\_nombreapellido\_desafio2** en tu carpeta de prácticas.

## Práctica 7. Programar un algoritmo. Me presento con Scratch



|                  |          |
|------------------|----------|
| NOMBRE           | Balubita |
| EDAD             | 150      |
| COLOR FAVORITO   | Azul     |
| MÚSICA PREFERIDA | Hip-hop  |

En la parte de teoría de la unidad hemos visto que utilizamos los algoritmos para resolver problemas y que los codificamos en programas para que el ordenador pueda procesar las soluciones. En esta práctica vamos a resolver un problema que se da con cada comienzo de curso: presentarte a tus compañeros y profesores.

### Ejercicio 1. Análisis

Para nuestra presentación utilizaremos el objeto **Starfish**, de la categoría **Animales**, y el fondo **beach malibu**. Los encontrarás en la biblioteca de objetos y la de fondos, respectivamente.

En primer lugar, pensamos cuatro aspectos que definen a nuestro objeto: su nombre y edad, su color favorito, sus gustos y aficiones o su música preferida.

Decidimos que nuestro objeto Starfish se llama Balubita, que tiene 150 años, que su color favorito es el azul y que su música preferida es el hip-hop.

### Ejercicio 2. Diseño

Ahora toca diseñar cómo llevar a cabo la presentación. ¿El proyecto se desarrollará secuencialmente? ¿Se deberá repetir alguna instrucción? ¿Será necesario que se cumpla alguna condición para que se activen movimiento, música o efectos sonoros? ¿Se moverá el personaje?

Balubita, para presentarse, quiere decir su nombre, edad y color favorito de forma secuencial. Además, quiere que, al decir su color, ella misma cambie de color y se vuelva azul. Y quiere moverse por la playa. Asimismo, a continuación, Balubita pedirá que hagamos clic en ella para que suene su música preferida.

### Ejercicio 3. Diagrama de flujo

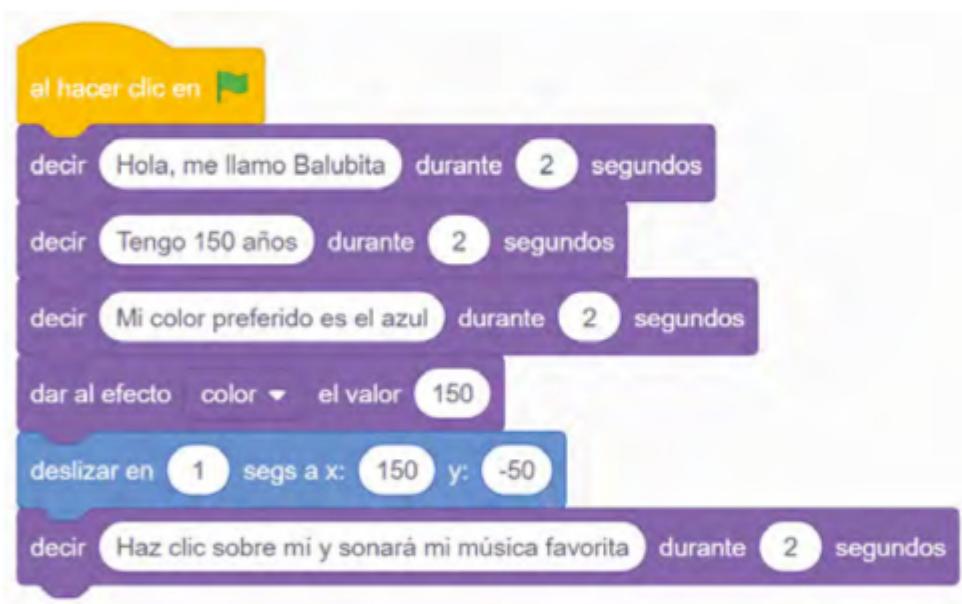
Al margen puedes ver el diagrama de flujo que indica el orden de las instrucciones. Corresponde a un algoritmo que combina las estructuras secuencial y selectiva.



#### Ejercicio 4. Programar en Scratch

Ahora codificaremos la presentación en el lenguaje de programación Scratch, siguiendo el diagrama de flujo.

1. Entra en <http://scratch.mit.edu> y haz clic en **Crear**. Inicia un nuevo proyecto yendo a **Archivo / Nuevo**. Llámalo **UD01\_P7\_nombre-apellido**.
2. Selecciona el personaje y el fondo indicados. Elimina el personaje **Objeto1** que viene por defecto.
3. Comienza la codificación con la instrucción **al hacer clic en bandera verde** y ve incluyendo cada una de las instrucciones necesarias.
4. Utiliza la instrucción **decir ... durante ... segundos**, de la categoría **Apariencia**, para incluir cada uno de los mensajes. Elige una duración de **2** segundos en cada mensaje.
5. Para cambiar el color, añade la instrucción **dar al efecto color el valor ...** y escribe **150** para que el personaje quede de color azul.
6. Anida después la instrucción **deslizar en 1 segs a x: 150 y: -50**, de la categoría **Movimiento**.



7. Incluye en otro programa, por separado, la instrucción **al hacer clic en este objeto**, de la categoría **Eventos**.
8. Anida en esta instrucción la orden **iniciar sonido ...**, de la categoría **Sonido**, y selecciona **Hip Hop** (este sonido se incluye en la biblioteca de sonidos de Scratch). Para encontrar este sonido, o cualquier otro de la biblioteca, te puede ser útil acceder a la pestaña **Sonidos** y escribir su nombre en el buscador.





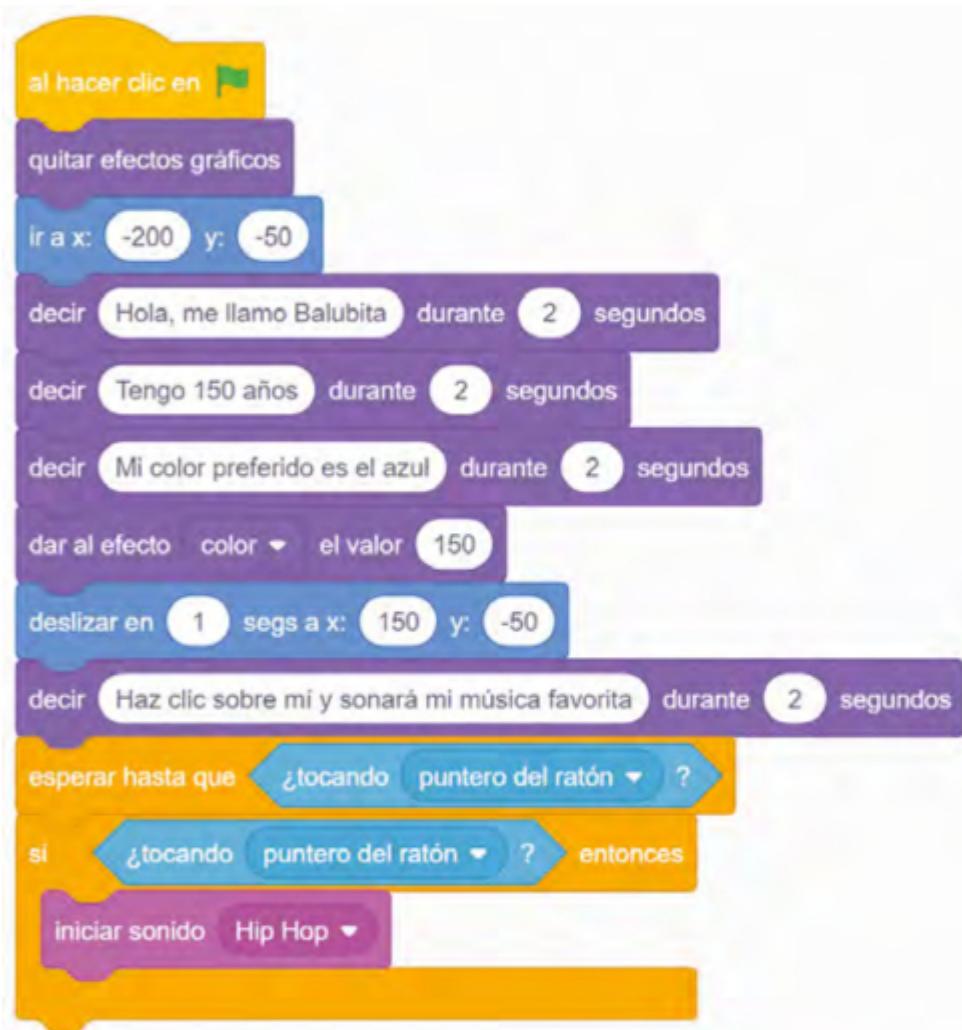
### Ejercicio 5. Mejorar la presentación

Puedes mejorar el programa anterior si te fijas en el diagrama de flujo. Como puedes observar, la parte final incluye la condición de que el sonido suene sólo si hacemos clic en el personaje.

1. Elimina el programa que se ejecuta al hacer clic en el objeto.
2. En el programa que no has borrado, anida la orden **esperar hasta que ...**, de la categoría **Control**, e incluye en ella la instrucción **¿tocando puntero del ratón?**, de la categoría **Sensores**. Así indicamos al personaje que debe esperar hasta que lo toquemos haciendo clic con el puntero del ratón.
3. Anida ahora la instrucción **si ... entonces**, de la categoría **Control**, e incluye en ella la instrucción **¿tocando puntero del ratón?**, de la categoría **Sensores**.
4. Por último, anida en el bucle **si ... entonces** la instrucción **iniciar sonido Hip Hop**, de la categoría **Sonido**.
5. Comprueba el resultado haciendo clic en la bandera verde.



6. Observa que el personaje se queda en la posición  $x = 150$ ,  $y = -50$  y de color azul. Para evitar esto y hacer que el programa comience con el personaje en su posición y condiciones iniciales, se incluyen habitualmente al comienzo del programa las condiciones de partida. Para ello, añade debajo de la instrucción **al hacer clic en bandera verde** las instrucciones **quitar efectos gráficos**, de la categoría **Apariencia**, e **ir a x: -200 y: -50**, de la categoría **Movimiento**. Así, el personaje se situará en esas coordenadas y se mostrará de su color original cada vez que reiniciemos el programa.



7. Prueba el programa. Descarga el proyecto **UD01\_P7\_nombreadellido.sb3** en tu ordenador yendo a **Archivo / Guardar en tu ordenador** y seleccionando la carpeta en la que guardas tus trabajos.

### DESAFÍO

- Modifica el programa para que la presentación se ajuste a ti mismo. Puedes incluir tu propia foto como personaje, y como fondo, un lugar en el que te guste estar. Guarda el proyecto como **UD01\_P7\_nombreadellido\_desafio1.sb3**.

## Práctica 8. Programar un algoritmo. El problema del acné



A tu edad, es fácil que ya conozcas al temido acné, esos granitos que empiezan a aparecer por tu cara y que resultan tan difíciles de quitar. También habrás oído decir que cuando crezcas irán desapareciendo, pero mientras ese momento llega, ¿qué hacer? La respuesta es fácil: cuidar tu alimentación, extremar la higiene en tu piel y acudir al médico si el problema es severo.

En esta práctica vamos a analizar el problema del acné y a darle la respuesta adecuada siguiendo algoritmos. Además, codificaremos en Scratch los algoritmos utilizados.

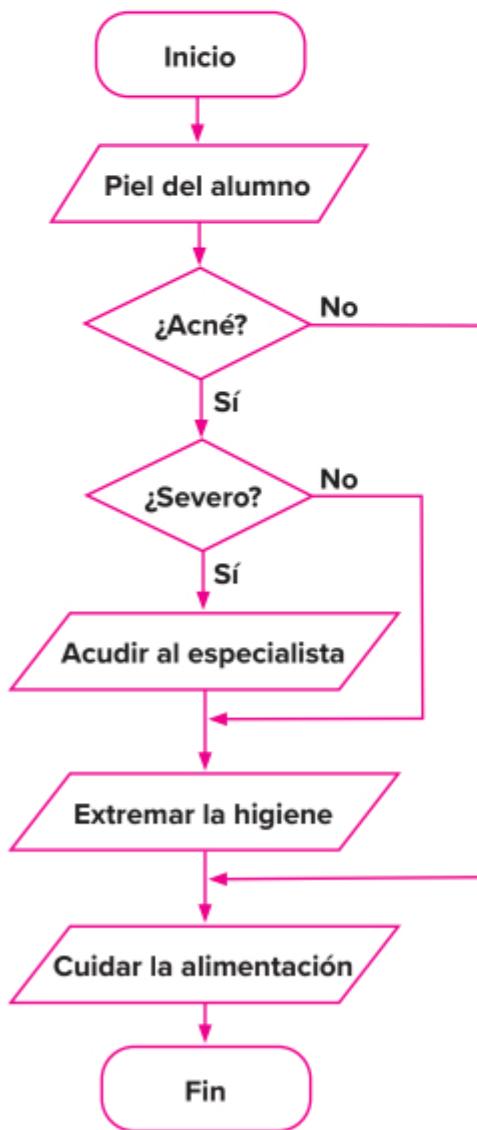
### Ejercicio 1. Análisis

Observa tu piel y la de tus compañeros. Seguramente habrá algunos que no tengan acné y otros que sí; entre éstos, alguno tendrá granitos aislados y otros tendrán mayor cantidad. Ante estas diferencias, el tratamiento para cada caso deberá ser distinto.

### Ejercicio 2. Diseño

Las situaciones que se pueden producir y su tratamiento serán las siguientes:

- Alumnos sin acné. En este caso será necesario prevenir su aparición. Para ello, es importante cuidar la alimentación evitando el abuso de grasas y azúcares.
- Alumnos con granitos puntuales en la piel. En este caso, bastará con cuidar la alimentación y extremar la higiene en las manos y en la piel para evitar que los granitos se diseminen por otras zonas.
- Alumnos con acné intenso o severo. En este caso, además de cuidar la alimentación y la higiene, es recomendable acudir al dermatólogo y que él nos recomiende un tratamiento corrector para este problema.



### Ejercicio 3. Diagrama de flujo

Al margen puedes ver el diagrama de flujo que indica el orden de las instrucciones. El algoritmo se repetirá para cada uno de los alumnos de la clase.

### Ejercicio 4. Programar en Scratch

Codifica ahora el problema del acné en el lenguaje de programación Scratch, siguiendo el diagrama de flujo.

1. Inicia un nuevo proyecto yendo a **Archivo / Nuevo**. Llámalo **UD01\_P8\_nombreapellido**.
2. Selecciona como personaje a **Avery** y selecciona como fondo **chalkboard**.
3. Comienza la codificación con la instrucción **al hacer clic en bandera verde** y ve incluyendo cada una de las instrucciones necesarias.
4. El programa va a analizar a cada uno de los alumnos de tu clase. Incluye la instrucción **repetir ...** e indica el número de alumnos que sois en clase. En el ejemplo hemos puesto **20**.

5. Anida la instrucción **decir Hola. Voy a resolver tu problema durante 2 segundos**, de la categoría **Apariencia**.
6. El personaje te pregunta a continuación tu nombre. Incluye para ello la instrucción **preguntar ¿Cómo te llamas? y esperar**, de la categoría **Sensores**.
7. Anida las instrucciones **decir Hola durante 1 segundos** y **decir respuesta durante 1 segundos**, de la categoría **Apariencia**. Fíjate: **respuesta** aparece dentro de la categoría **Sensores**, y, en ella, el programa almacena el nombre que escribas cuando te pregunte tu nombre.
8. Siguiendo el diagrama de flujo, la siguiente pregunta es si tienes o no acné. Por tanto, anida a continuación la instrucción **preguntar ¿Tienes acné? y esperar**, de la categoría **Sensores**. Hasta aquí, el programa debe quedar como en la figura 32.



Fig. 32

9. Ahora, siguiendo la estructura del diagrama de flujo, tienes que utilizar una estructura condicional que analice las respuestas. Anida el bucle **si ... entonces / si no**, de la categoría **Control**.
10. En la condición de la instrucción anterior, tienes que ver si se cumple que la respuesta sea "Sí", porque en ese caso habrá que ver si el acné es severo o no. Para ello, anida la condición utilizando el **comparador de igualdad** de la categoría **Operadores**, como puedes ver en la figura 33.

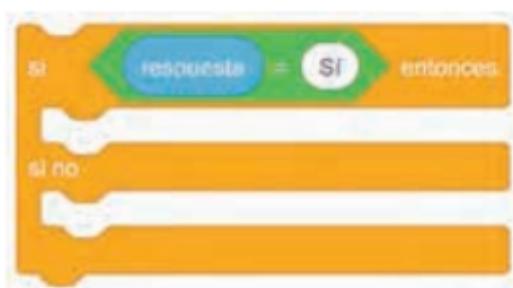


Fig. 33

### iCuidado con la ortografía!

En el bucle condicional, sólo se cumplirá la condición si la respuesta es "Sí". No puedes olvidar la tilde, porque en ese caso el programa entenderá que la condición no se cumple.



11. Anida dentro del bucle **sí ... entonces** la instrucción **preguntar ¿Es severo? y esperar**, de la categoría **Operadores**.
12. En función de la respuesta a esta pregunta, las indicaciones para corregir el acné serán unas u otras. Por tanto, es necesario anidar otro bucle condicional que analice si la respuesta es "Sí" o "No" a esta nueva pregunta. Incluye el bucle **sí ... entonces / si no** y codifica la condición igual que en el punto 10 (observa la figura 34).

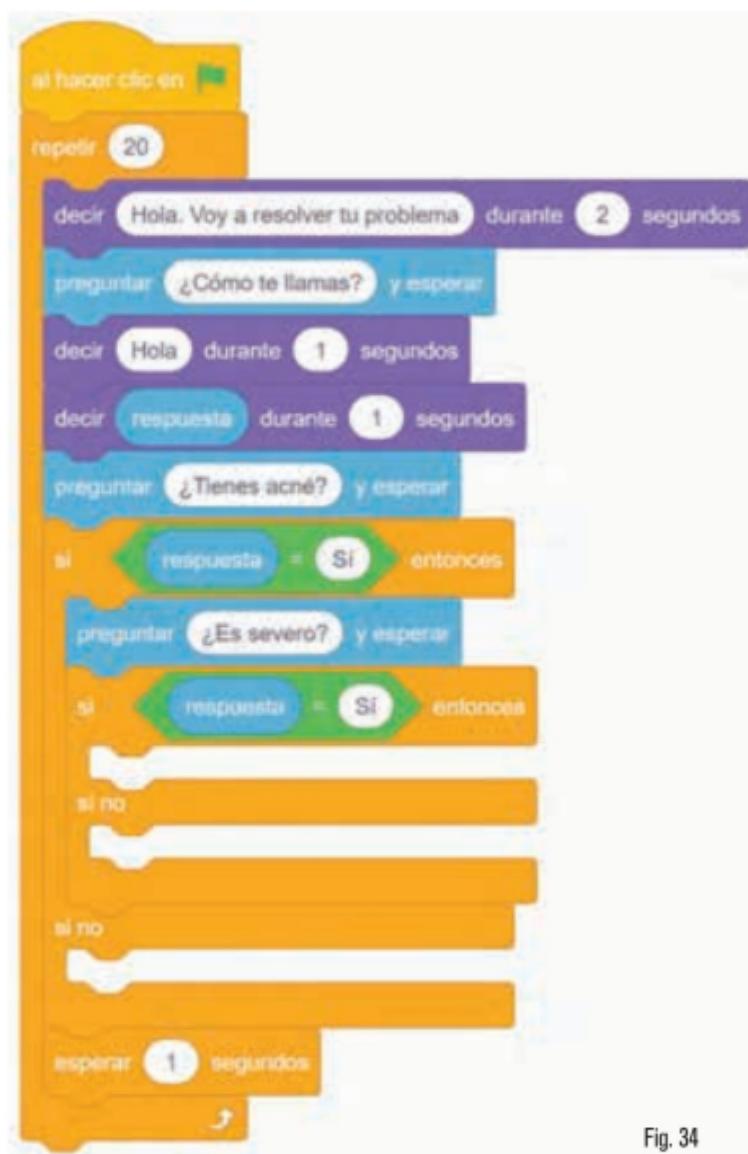


Fig. 34

13. Anida a continuación las instrucciones de las recomendaciones a seguir en el caso de que el acné sea severo (figura 35). Utiliza para ello la instrucción **dicir**, de la categoría **Apariencia**.
14. Debajo de cada **si no**, anida las instrucciones correspondientes (para el caso de que se tenga acné pero no sea severo y para el de que no se tenga acné). Utiliza nuevamente la instrucción **dicir**, de la categoría **Apariencia**.
15. Prueba el programa. Descarga el proyecto **UD01\_P8\_nombre-apellido.sb3** en tu ordenador yendo a **Archivo / Guardar en tu ordenador** y seleccionando la carpeta en que guardas tus trabajos.



### Cómo retardar el desarrollo de instrucciones iterativas

La última instrucción del programa es **esperar 1 segundos**, de la categoría **Control**. Esta instrucción permite que las repeticiones no sean inmediatas, sino que transcurre un segundo entre la valoración de cada uno de los alumnos.



Fig. 35

### DESAFIOS

1. Modifica el programa anterior para que, cuando le digas que no tienes acné, te responda "¡Enhорabuena!". Guarda el proyecto como **UD01\_P8\_nombreapellido\_desafío1.sb3**.
2. Basándote en esta práctica, codifica en Scratch un programa de diagnóstico que te diga si te estás alimentando correctamente. Si tomas refrescos azucarados diariamente, debe decirte que superas la cantidad de azúcar recomendada. Si no comes frutas y verduras diariamente, debe decirte que tienes que aumentar la cantidad de frutas y verduras en tu dieta. Si no tomas refrescos azucarados habitualmente y comes frutas y verduras diariamente, el programa debe felicitarte. Guarda el proyecto como **UD01\_P8\_nombreapellido\_desafío2.sb3**.

## Práctica 9. Programar un algoritmo. Mis ahorros



Otro de los problemas que a tu edad suelen aparecer es el de las finanzas. Sería muy útil contar con una pequeña calculadora mensual que te indique cuánto dinero te va quedando de tus ahorros, en función de tus ingresos y gastos.

En esta práctica, vas a aprender a desarrollar los algoritmos necesarios para poder codificar en Scratch tu calculadora mensual.

### Ejercicio 1. Análisis

El problema consiste en crear una calculadora que vaya restando los gastos que diariamente vayas introduciendo y sumando los ingresos que se produzcan. Además, para evitar que te quedes sin dinero, te avisará cuando en la hucha haya 50 euros o menos. Cuando haya más de 50 euros, la calculadora te dará un mensaje de ánimo para que continúes ahorrando.

### Ejercicio 2. Diseño

Para poder realizar la calculadora, necesitas trabajar con variables. En este caso son necesarias tres variables: ingresos, gastos y contenido de la hucha.

Utilizarás algoritmos de estructura secuencial, selectiva e iterativa.

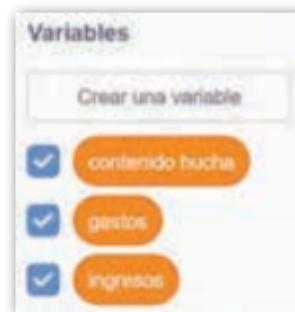
### Ejercicio 3. Diagrama de flujo

Al margen puedes ver el diagrama de flujo que indica el orden de las instrucciones. Los algoritmos se repetirán 30 veces, una por cada día del mes (considerando meses de 30 días).

### Ejercicio 4. Programar en Scratch

Codifica ahora la calculadora en el lenguaje de programación Scratch, siguiendo el diagrama de flujo.

- Inicia un nuevo proyecto yendo a **Archivo / Nuevo**. Llámalo **UD01\_P9\_nombreapellido**.
- En primer lugar, debes definir las variables que vas a utilizar. Accede a la categoría **Variables** y haz clic en **Crear una variable**. Crea las variables **contenido hucha**, **gastos** e **ingresos**.



- Inicia ahora el programa con la instrucción **al hacer clic en bandera verde**.

4. Fija el valor inicial de las variables. En este caso, supón que partes de un contenido en la hucha de 150 euros. Selecciona las instrucciones **dar a contenido hucha el valor 150**, **dar a gastos el valor 0** y **dar a ingresos el valor 0**, de la categoría **Variables**.
5. A partir de aquí, todas las instrucciones se repetirán durante los 30 días de cálculo. Para ello, utiliza una instrucción iterativa. Selecciona el bloque **repetir 30**, de la categoría **Control**.
6. Anida a continuación la instrucción **preguntar ¿Has tenido ingresos? y esperar**, de la categoría **Sensores**.
7. Ante la pregunta planteada, es necesario incluir ahora un bloque de estructura selectiva. Anida un bucle **sí ... entonces / si no**, de la categoría **Control**.
8. La condición que hay que verificar es si la respuesta a la pregunta anterior es “Sí” o no lo es. Para ello utiliza un **comparador de igualdad** de la categoría **Operadores**, como puedes ver en la figura 36.



Fig. 36

9. Si la respuesta es “Sí”, necesitas saber qué ingresos has tenido. Incluye a continuación la instrucción **preguntar ¿Cuánto? y esperar**, de la categoría **Sensores**.
10. Anida a continuación la instrucción **sumar a ingresos respuesta**, de la categoría **Variables**. De esta manera, el valor de la variable **ingresos** aumentará en el valor que se introduzca como ingreso cada vez.
11. Anida también la instrucción **dar a contenido hucha el valor contenido hucha + respuesta**, de la categoría **Variables**. Para codificar esta instrucción necesitas un **operador de suma** de la categoría **Operadores** y el valor de **respuesta** de la categoría **Sensores**. Observa la figura 37.



Fig. 37

- 12.** Si la respuesta a la pregunta de si has tenido ingresos (punto 6) es “No”, entonces es necesario saber si ha habido gastos. Para ello, en el bloque si ... entonces / si no, anida bajo la rama si no la instrucción preguntar ¿Has tenido gastos? y esperar, de la categoría **Sensores**.
- 13.** Anida a continuación un bloque si ... entonces para verificar la respuesta, como has hecho en el punto 7, e incluye la instrucción preguntar ¿Cuánto? y esperar, de la categoría **Sensores**. Anida las instrucciones para modificar los valores de las variables **gastos** y **contenido hucha**. Los pasos son similares a los detallados en las instrucciones de los puntos 9 y 10. Fíjate en la figura 38.

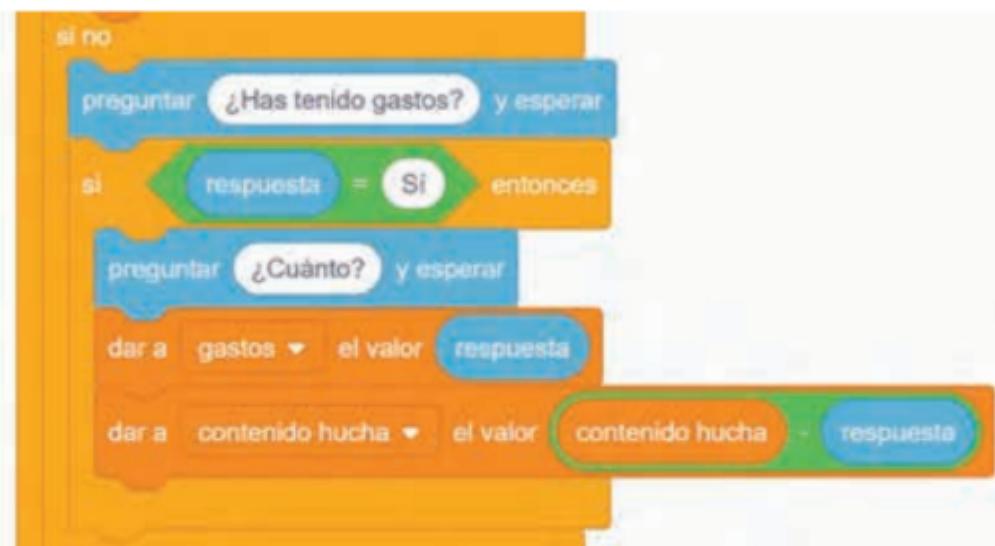


Fig. 38

- 14.** A continuación, fuera del bloque si ... entonces / si no, incluye una instrucción para que el personaje te diga el contenido de la hucha. Usa para ello la instrucción dicir, de la categoría **Apariencia**, junto con la instrucción unir, de la categoría **Operadores**. La instrucción debe quedar así (figura 39):



Fig. 39

- 15.** Finalmente, sólo queda valorar si el contenido de la hucha es mayor que 50 euros o no. Utiliza para ello un nuevo bloque si ... entonces / si no y las instrucciones dicir Vas bien durante 2 segundos y dicir Hay que ahorrar durante 2 segundos, de la categoría **Apariencia**; la calculadora dirá una cosa u otra en función del contenido de la hucha.

16. El programa debe quedar como en la figura 40.

17. Prueba el programa. Descarga el proyecto **UD01\_P9\_nombreeapellido.sb3** en tu ordenador yendo a **Archivo / Guardar en tu ordenador** y seleccionando la carpeta en que guardas tus trabajos.

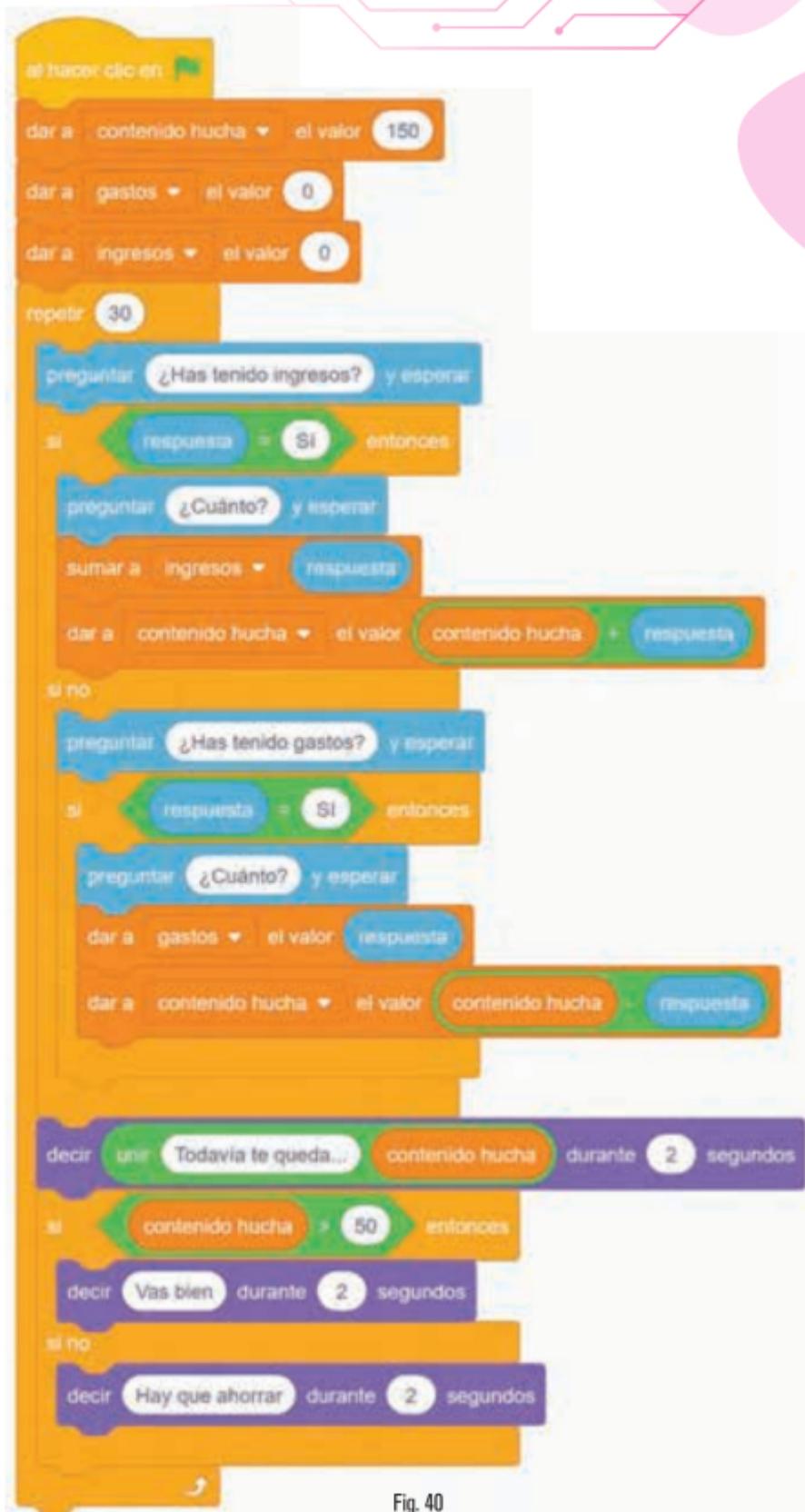


Fig. 40

### DESAFÍO

- Modifica el programa anterior para añadir como condición nueva que, cuando el contenido de la hucha sea menor de 20, el programa muestre el mensaje “¡No despilfarres tanto!”. Guarda el proyecto como **UD01\_P9\_nombreeapellido\_desafío1.sb3**.

## Práctica 10. Programar un videojuego. *El elefante hambriento*



Vamos a crear un juego en el que un elefante hambriento tiene que llegar hasta su comida pero para ello debe antes trabajar y ser capaz de tocar una pelota cinco veces. Para hacer este programa necesitaremos que se vaya almacenando en algún sitio el número de veces que va tocando la pelota, y para ello usaremos una variable.



Fig. 41



Fig. 42

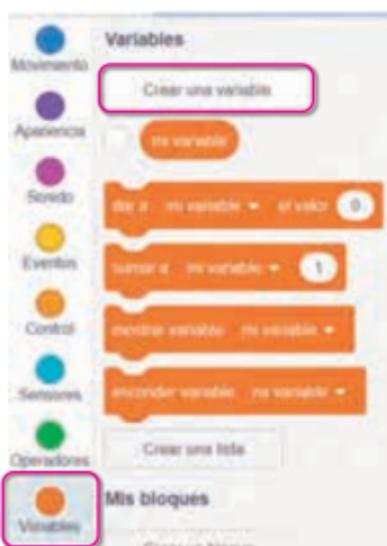


Fig. 43

### Nombrar las variables

Recuerda que, para que tus programas sean de buena calidad, tienes que darles a las variables nombres que sean significativos y que hagan referencia a la función que desempeñan en el programa. En este caso, como se trata de un videojuego, necesitaremos crear una variable llamada **puntos**.

### Ejercicio 1. Realizar la interfaz

- Inicia un nuevo proyecto yendo a **Archivo / Nuevo**. Llámalo **UD01\_P10\_nombreapellido**. Elimina el personaje **Objeto1** que viene por defecto.
- Vamos a crear un escenario como el que se muestra en la figura 41. Para añadir el fondo, ve a **Elige un fondo** y escoge **Stripes**.
- A continuación, crearemos los objetos necesarios. Para ello, en **Elige un objeto**, escoge los objetos **Elephant**, **Ball** y **Fruit Salad**. Después, colócalos como en la figura 41.
- Sitúate en el elefante y, en **Dirección**, elige **Izquierda/Derecha**, como indica la figura 42. Esto sirve para que el objeto no se ponga patas arriba cuando gire, pero sí se dará la vuelta cuando ande hacia la izquierda.

### Ejercicio 2. Crear una variable

- Selecciona la categoría **Variables** y luego pulsa **Crear una variable** (figura 43). En la ventana emergente escribe “puntos” como nombre de la variable y pulsa **Aceptar**.

### Bloques de variables

Hasta que no crees la variable, ésta no te aparecerá en los bloques de instrucciones correspondientes.

- Como hemos visto en la parte de teoría de la unidad, para poder revisar más fácilmente los programas es de gran ayuda poder ver el valor que va tomando una variable en el desarrollo. Así, pues, comprueba que la variable **puntos** está seleccionada (con un tic a su izquierda) para que aparezcan los puntos en el escenario.

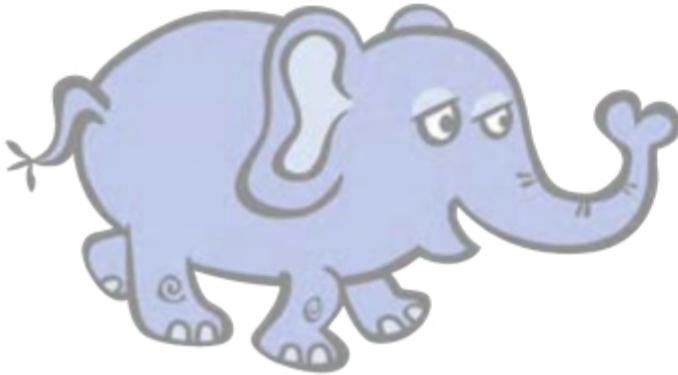
### Ejercicio 3. Programar la pelota

La variable puede ser usada tanto en la programación del elefante como en la de la pelota. En este caso la usaremos en la programación de la pelota.

1. Selecciona la pelota y realiza el programa que tienes en la figura 44.
2. En primer lugar, reiniciamos la variable con la instrucción dar a puntos el valor 0, para que, cada vez que empiece el juego, la puntuación vuelva a cero.
3. Para que te aparezca la variable **puntos**, debes desplegar el menú de la instrucción dar a ... el valor ..., ya que es ahí donde aparecen todas las variables una vez creadas.
4. La instrucción ir a posición aleatoria permite que la pelota vaya rebotando por toda la pantalla. Si la quitas, la pelota irá solamente moviéndose en una línea de la pantalla.
5. La instrucción por siempre nos permite el movimiento continuo de la pelota.
6. La instrucción si ¿tocando Elephant? entonces, junto con los bloques siguientes, sirve para que la variable se incremente en uno cada vez que el elefante toque la pelota y para que la pelota parta de una posición aleatoria cada vez que toque al elefante. Si no pusieramos la instrucción ir a posición aleatoria, la pelota se quedaría tocando al elefante y sumaría puntos continuamente (puedes hacer la prueba de quitarla y lo verás).
7. La instrucción sumar a puntos 1 nos permite ir incrementando en uno la variable **puntos**, para ir contando el número de veces que el elefante toca la pelota.
8. La instrucción si puntos = 5 entonces se construye arrastrando dentro del bloque si ... entonces el bloque verde correspondiente (de la categoría **Operadores**) y, por último, arrastrando la variable **puntos** (de la categoría **Variables**) sobre el hueco del operador.
9. La instrucción detener este programa sirve para que, cuando la variable alcance el valor de 5 puntos, la pelota se pare y permita al elefante ir a por su comida.



Fig. 44





### Reiniciar las variables y el aspecto al empezar el juego

Observa que el juego acaba con el elefante en el disfraz **elephant-b** y junto al bol de fruta. Por eso es importante colocar el juego siempre justo después de la instrucción **al hacer clic en bandera verde**.

De la misma forma, es importante reiniciar las variables, como hemos hecho al inicio del programa de la pelota dándole valor 0.

### Instrucción "detener"

Observa que la instrucción **detener ...** puede configurarse para detener un solo programa, como en el caso de la pelota, o para detener todos los programas, como en el caso del elefante. Esta instrucción es necesaria siempre que estemos usando bloques como **por siempre**.

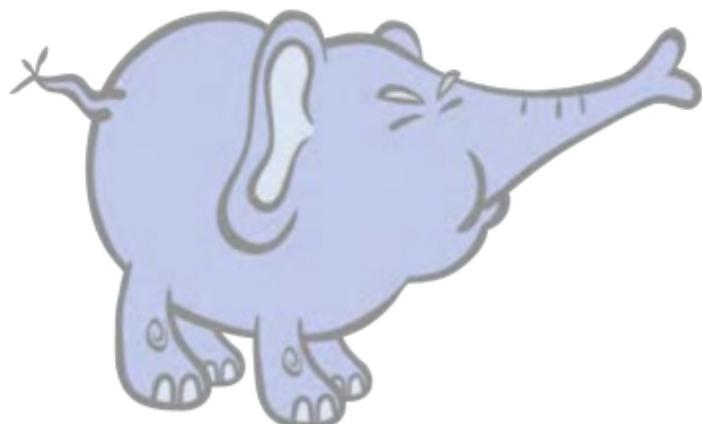


Fig. 45. Programación del elefante

### Ejercicio 4. Programar el elefante

1. Selecciona el elefante y realiza el programa de la figura 45.
2. Lo primero que hacemos es hacer que el programa, cada vez que empiece, coloque al elefante en el sitio correcto y con el disfraz correcto. Para ello usamos la instrucción **ir a x: -164 y: -108**, con la que el elefante partirá siempre de la esquina inferior izquierda, y luego añadimos la instrucción **cambiar disfraz a ...**.
3. A continuación, hacemos el elefante más pequeño para todo el programa con la instrucción **fijar tamaño al 50%**. Otra opción habría sido transformarlo en la solapa **Disfraces**, pero entonces deberíamos transformar todos los disfraces que vamos a usar y dejarlos al mismo tamaño, por lo que es mejor la opción que hemos elegido.
4. El bloque **por siempre** nos permite que el elefante se mueva cada vez que se pulse la tecla correspondiente y no sólo una vez.
5. Observa cómo hemos conseguido el movimiento del elefante utilizando cuatro bloques **si ... entonces**, en lugar de hacer cuatro programas que empiecen por la instrucción **al presionar tecla ...** (que es lo que hicimos en la práctica 5 para el astronauta). Ambas opciones son válidas.
6. El siguiente bloque de instrucciones es un bloque especial, que incluye dos instrucciones: **si ... entonces** y **si no**. Si el elefante quiere comer antes de haber tocado la pelota cinco veces (es decir, siendo la variable **puntos** menor que 5), saldrá un aviso de que no puede comer todavía y se detendrá el juego. Si no, es que el elefante ha conseguido los cinco toques (entonces, la variable vale en ese momento 5 puntos) y ya puede comer.
7. Entonces, el elefante dice “¡Por fin!” durante un segundo y luego cambia de disfraz tres veces, para lo cual usamos el bloque **repetir**. Y, por último, detenemos todos los programas con **detener todos** para que se pare el juego.

### Ejercicio 5. Prueba el proyecto que has realizado

Comprueba que el programa funciona correctamente y luego descarga el proyecto **UD01\_P10\_nombreapellido.sb3** en tu ordenador yendo a **Archivo / Guardar en tu ordenador**.



#### DESAFIOS

1. Modifica el programa aumentando la velocidad de la bola para que el juego sea más difícil. Guarda el proyecto como **UD01\_P10\_nombreapellido\_desafio1.sb3**.
2. Haz que el juego sea más difícil aún aumentando el número de puntos necesarios para que el elefante pueda ir a por su comida. Guarda el proyecto como **UD01\_P10\_nombreapellido\_desafio2.sb3**.
3. Completa el juego con un contador de tiempo. Guarda el proyecto como **UD01\_P10\_nombreapellido\_desafio3.sb3**.

## Práctica 11. Extensiones de Scratch. Integración de gráficos y sonidos

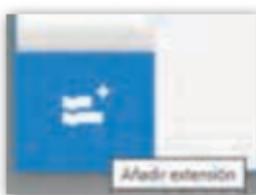


Fig. 46

### Vocabulario

El **hardware** es la parte física del ordenador. Scratch permite programar varios dispositivos, como por ejemplo una webcam o la placa micro:bit.

El **software** son los programas, como los que hacemos con Scratch.

- Inicia un nuevo proyecto yendo a **Archivo / Nuevo**. No hace falta que le pongas nombre al proyecto, no lo vamos a guardar.
- Debajo de las categorías de bloques, en la esquina inferior izquierda, tienes el botón **Añadir extensión** (figura 46). Púlsalo y verás que te aparecen las extensiones de Scratch (figura 47), que son una serie de funcionalidades que te permiten programar tanto software como hardware.



Fig. 47

- Selecciona las siguientes extensiones de software: **Música**, **Lápiz**, **Texto a voz** y **Traducir**. Verás que se incorporan a la columna de categorías de bloques. Observa los bloques que contienen estas extensiones, pruébalos y trata de imaginar qué podrías hacer con cada uno de ellos. Estas extensiones son muy útiles, nos sirven para lo siguiente:
  - Música**: para tocar instrumentos
  - Lápiz**: para dibujar con los objetos
  - Texto a voz**: para hacer que tus proyectos hablen
  - Traducir**: para traducir textos a muchos idiomas
- A continuación, usando un procesador de texto, redacta un documento en el que expongas todo lo que has imaginado sobre los bloques de extensiones de software. Guarda el documento como **UD01\_P11\_nombreadellido**.
- Ahora selecciona la extensión de hardware **Sensor de vídeo**, para interaccionar con la webcam. Observa los bloques que contiene esta extensión, pruébalos y trata de imaginar qué podrías hacer con ellos. A continuación, escríbelo también en tu documento, y luego guárdalo y ciérralo.
- Por último, prueba la extensión **micro:bit** (figura 48) y observa sus bloques (figura 49). Esta extensión sirve para enlazar programas de Scratch con la tarjeta micro:bit (que aprenderás a usar en la unidad de robótica).



Fig. 48

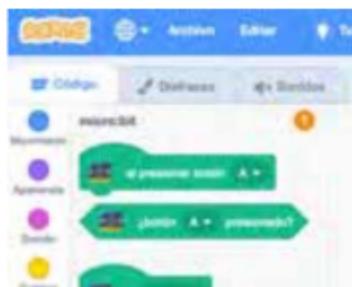


Fig. 49

## Práctica 12. Polígonos de colores. Subalgoritmos en Scratch



En ocasiones, para facilitar la evaluación del software y evitar que los programas sean demasiado largos utilizamos subalgoritmos o procedimientos, que son pequeños programas a los que llama un programa principal para su ejecución.

### Ejercicio 1. Dibujar un polígono en Scratch

- Inicia un nuevo proyecto yendo a **Archivo / Nuevo**. Llámalo **UD01\_P12\_1\_nombreadellido**.
- Haz clic en **Añadir extensión** y elige **Lápiz**.
- Copia el programa de la figura 50 para el personaje que aparece por defecto.

El programa permite dibujar un polígono de los lados que le indiquemos.

La instrucción **mover 360 / respuesta pasos** ajusta el tamaño de los lados del polígono en función del número de lados que indiquemos.

La instrucción **girar 360 / respuesta grados** hace que el personaje gire en cada movimiento los grados necesarios para dibujar el polígono.

- Prueba el programa con distintos números de lados.



Fig. 50

### Ejercicio 2. Crear un subalgoritmo

Vamos a modificar el programa anterior para crear un subalgoritmo que se pueda utilizar desde el programa principal.

- Accede al menú **Mis bloques** y haz clic en **Crear un bloque**. Escribe como nombre **dibujar polígonos** (figura 51).
- Comprueba que ha aparecido en tu espacio de programación un bloque nuevo, **definir dibujar polígonos** (figura 52), similar a los bloques de inicio del menú **Eventos**. Anida debajo de dicho bloque todo el programa anterior, excepto la instrucción **al hacer clic en bandera verde**.



Fig. 52



Fig. 51

- Debajo del bloque **al hacer clic en bandera verde** (que ha quedado en el área de código), anida las instrucciones **dicir Vamos a dibujar polígonos por 2 segundos**, de la categoría **Apariencia**, y **dibujar polígonos**, de la categoría **Mis bloques**.



Fig. 53



Fig. 54

Fíjate en que este procedimiento también te va a permitir dibujar polígonos. Es muy similar al que has realizado en el ejercicio 1 de esta práctica pero en este caso los polígonos se dibujan en función de dos parámetros que hemos llamado **color** y **lados**.

El parámetro **color** permite definir el color del lápiz con el que se dibujarán los polígonos.

El parámetro **lados** permite determinar el número de lados que tendrá cada polígono.

Ambos parámetros se definen en el programa principal.

4. En tu área de código has de tener un programa que comienza con la instrucción al hacer clic en bandera verde (figura 53) y un subalgoritmo o procedimiento de ese programa que comienza con la instrucción definir dibujar polígonos.
5. Prueba el proyecto haciendo clic en la bandera verde.
6. Descarga el proyecto **UD01\_P12\_1\_nombreadellido.sb3** en tu ordenador yendo a **Archivo / Guardar en tu ordenador**.

### Ejercicio 3. Subalgoritmos con parámetros

En Scratch, los procedimientos pueden incluir diferentes parámetros que permiten el intercambio de datos entre el programa principal y el subalgoritmo. Los parámetros pueden ser de tres tipos: un número o texto, una expresión booleana o una etiqueta.

En este caso, vamos a crear un subalgoritmo con tres parámetros de tipo numérico.

1. Inicia un nuevo proyecto yendo a **Archivo / Nuevo**. Llámalo **UD01\_P12\_2\_nombreadellido**.
2. Accede al menú **Mis bloques** y crea un bloque llamado **dibujar polígono**. Añade en él dos parámetros de tipo número llamados **color** y **lados** (figura 54).
3. Para el personaje, copia el procedimiento de la figura 55:



Fig. 55

4. Copia ahora el programa principal siguiente (figura 56):



Fig. 56

Fíjate en que los parámetros **color** y **lados** se definen en este programa. Cada uno de ellos se define como un número al azar entre unos valores determinados: el parámetro **color** se establecerá entre los valores 150 y 250, y el parámetro **lados** se establecerá entre los valores 3 y 14 para dibujar polígonos de hasta 14 lados como máximo.

### Comentarios

Para hacer aclaraciones sobre el funcionamiento de un programa, puedes incluir comentarios en él.

Por ejemplo, sitúate sobre la primera instrucción del subalgoritmo que has creado. Haz clic con el botón derecho y escoge **Añadir comentario**. Teclea “Este procedimiento dibuja polígonos de colores aleatorios”.



### Ejercicio 5. Prueba el proyecto que has realizado

1. Prueba el programa haciendo clic en la bandera verde. Haz clic en cualquier parte del escenario para ver los polígonos que se van dibujando.
2. Descarga el proyecto **UD01\_P12\_2\_nombreadellido.sb3** en tu ordenador yendo a **Archivo / Guardar en tu ordenador**.



### DESAFÍOS

1. Fijándote en el ejercicio 1 de esta práctica, modifica el programa que has realizado en la práctica 7 de esta unidad definiendo el procedimiento de presentación. El procedimiento debe ser llamado desde el programa principal. Guarda el proyecto como **UD01\_P12\_nombreadellido\_desafio1.sb3**.
2. Fijándote en el ejercicio 3 de esta práctica, modifica el programa que has realizado en la práctica 7 de esta unidad definiendo el procedimiento de presentación con dos parámetros llamados **edad** y **color preferido**. En el programa principal se definirá el valor de estos parámetros y se llamará al procedimiento de presentación. Guarda el proyecto como **UD01\_P12\_nombreadellido\_desafio2.sb3**.

# ACTIVIDADES de refuerzo

## Autocorrección y calificación online

Las actividades de este apartado se pueden responder desde la plataforma digital de BlinkLearning. Todas ellas son de autocorrección y la calificación queda registrada automáticamente en la plataforma.

## Pensamiento computacional

### 1. Completa las frases con las palabras que te ofrecemos:

**Palabras:** computacional, creativas, ordenador, problemas, soluciones

- En ocasiones, trabajar como lo haría un (...) nos puede ayudar a resolver problemas de la vida diaria y desarrollar (...) sencillas y (...) para problemas habituales.
- Esta forma de trabajo se denomina *pensamiento* (...) y consiste en abordar los (...) planteados como si fuéramos científicos informáticos.

### 2. Ordena del 1 al 5 los siguientes pasos, que describen el pensamiento computacional:

- Desarrollamos un modelo que pueda ser una solución.
- Hacemos deducciones, planteamos hipótesis, imaginamos situaciones... Utilizamos el pensamiento abstracto.
- Identificamos el aspecto esencial de un problema (datos, condicionantes y restricciones).
- La solución obtenida debe poder ser ejecutada por un sistema informático.
- Simplificamos los elementos de un problema y lo dividimos en otros más sencillos.

### 3. Une cada palabra con su definición.

|                |   |
|----------------|---|
| 1) Patrón      | a) Revisar los pasos y procedimientos que forman un algoritmo para que no haya errores, instrucciones poco precisas o innecesarias ni pasos duplicados.                       |
| 2) Información | b) Conjunto de características que se repiten de forma determinada en los objetos; puede tratarse de repeticiones de colores, simetrías, cambios de forma o de posición, etc. |
| 3) Depurar     | c) Conjunto de datos que nos aportan un significado de interés para un problema planteado.  |

### 4. El método para solucionar problemas comprende cuatro fases. Une cada una de ellas con su explicación.

| FASES                                     | EXPLICACIONES  |
|---|--|
| 1) Comprende el problema                  | a) Descarta los datos innecesarios o irrelevantes y decide qué ecuaciones vas a utilizar. Si el problema es complicado, divide el problema en partes más pequeñas fácilmente resolubles.   |
| 2) Establece un plan de acción            | b) Lee el enunciado del problema varias veces hasta conocer la incógnita, los datos y los condicionantes que tienes que respetar.  |
| 3) Desarrolla el plan. Obtén una solución | c) La solución al problema planteado tiene que ser completa, esto es, debe dar respuesta a todas las incógnitas. Por otra parte, no basta con llegar a una solución del problema, es necesario confirmar que la solución es la adecuada. Para confirmarlo, intenta llegar a ella por otro camino o estimar con antelación si es buena. |
| 4) Revisa la solución obtenida            | d) Sigue detalladamente el plan que has planteado. Simplifica siempre que sea posible y haz dibujos o diagramas para presentar las soluciones.   |

● 5. A lo largo de la unidad hemos aprendido que los datos son fundamentales en el pensamiento computacional. Indica a qué tipo de datos se refiere cada una de estas dos definiciones:

- a) Son representados mediante cifras. Pueden ser enteros (que permiten contar elementos) o decimales (que permiten expresar partes no completas de cantidades).
- b) Están formados por todo tipo de caracteres: cifras, letras, símbolos, etc. Ejemplos: las matrículas de los coches, las direcciones de correo electrónico y el número del NIF.

## Algoritmos

● 6. Une cada palabra con su definición.

|   |                      |
|---|----------------------|
| 1) Secuencia ordenada de pasos que resuelven un problema en un tiempo finito  | a) Diagrama de flujo |
| 2) Serie o sucesión de cosas que guardan entre sí cierta relación   | b) Terminal          |
| 3) Acción de repetir un proceso para alcanzar un objetivo o resultado   | c) Proceso           |
| 4) Acción de elegir una o más cosas entre varias  | d) Decisión          |
| 5) Gráfico que muestra la secuencia de las acciones que se han de realizar  | e) Secuencia         |
| 6) Símbolo que, en un diagrama de flujo, representa el comienzo o el fin del desarrollo de un algoritmo   | f) Selección         |
| 7) Símbolo que, en un diagrama de flujo, permite representar cada una de las acciones que hay que realizar para desarrollar un algoritmo  | g) Algoritmo         |
| 8) Símbolo que, en un diagrama de flujo, se utiliza cuando es necesario decidir entre dos o más opciones y señala el camino que habrá que seguir según cuál sea la opción elegida | h) Iteración         |

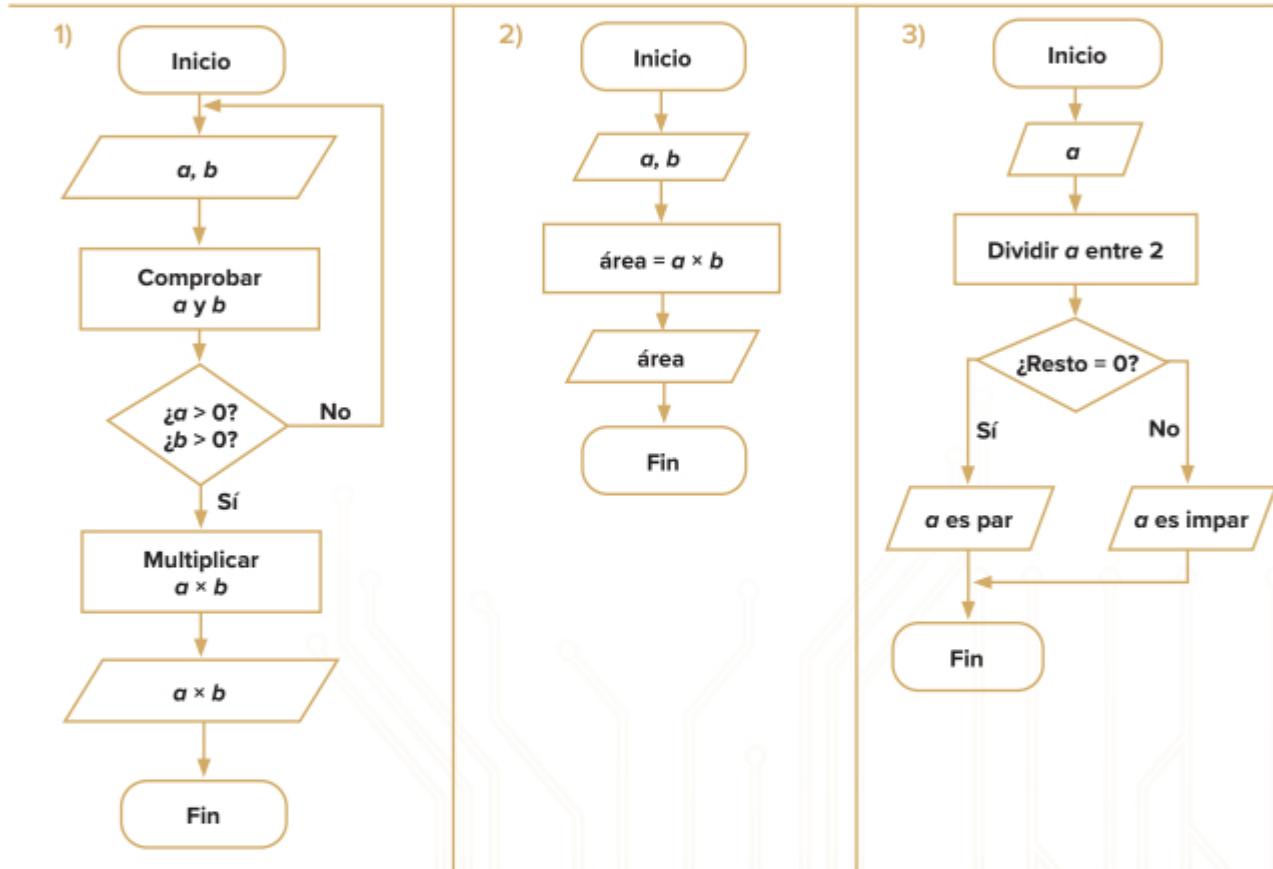
● 7. Une cada símbolo de los diagramas de flujo con la función que representa.

|  |   |
|--|---|
| 1)  | a) Terminal. Representa el comienzo o el fin del desarrollo de un algoritmo.  |
| 2)  | b) Proceso. Permite representar cada una de las acciones que hay que realizar para desarrollar el algoritmo.  |
| 3)  | c) Entrada o salida de información. Se utiliza cuando es necesaria información (datos adicionales para desarrollar el algoritmo) o se presentan datos o resultados. |
| 4)  | d) Decisión. Se utiliza cuando es necesario decidir entre dos o más opciones y señala el camino que habrá que seguir según cuál sea la opción elegida.              |
| 5)  | e) Línea de flujo. Señala el orden en que se desarrollan las acciones en el algoritmo.  |

8. Di si las siguientes afirmaciones sobre tipos de algoritmos son verdaderas o falsas:

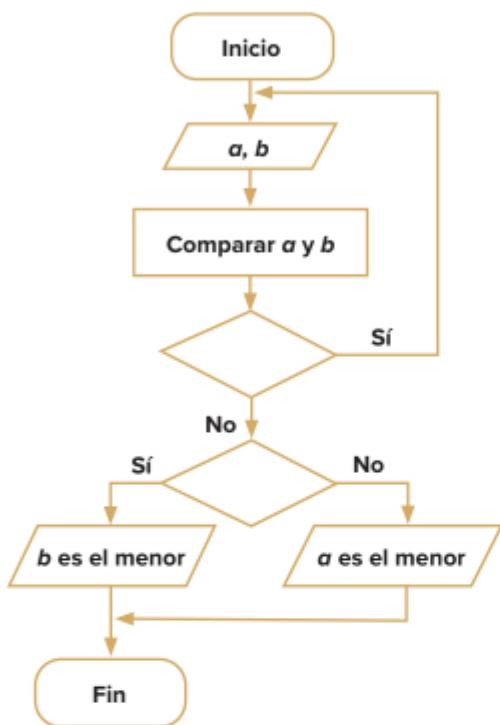
- a) En los algoritmos de estructura secuencial, las instrucciones se van cumpliendo en el orden en que aparecen.
- b) En los algoritmos de estructura secuencial, las instrucciones se llevan a cabo sólo si se cumple una determinada condición.
- c) Los algoritmos de estructura selectiva incluyen instrucciones que se repiten varias veces.

9. ¿Para qué nos sirve cada uno de los siguientes diagramas de flujo?



- a) Calcular el área de un rectángulo
- b) Calcular el perímetro de un rectángulo
- c) Multiplicar dos números
- d) Multiplicar dos números, siempre que ambos sean mayores que 0
- e) Saber si un número es par o impar
- f) Saber si un número es divisible entre dos o entre tres

10. El siguiente diagrama de flujo incompleto sirve para comparar dos valores distintos y saber cuál es el menor de ellos. ¿Cuál es la pregunta en cada una de las dos decisiones que faltan?



- a) La primera es  $a > b$ ? y la segunda es  $a < b$ ?
- b) La primera es  $a = b$ ? y la segunda es  $a > b$ ?
- c) Las dos preguntas son  $a > b$ ?
- d) La primera es  $a > b$ ? y la segunda es  $a = b$ ?

11. Teniendo en cuenta las normas sobre cómo hacer diagramas de flujo, indica para cada imagen si es correcta o incorrecta.

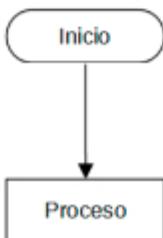
a)



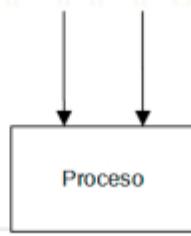
b)



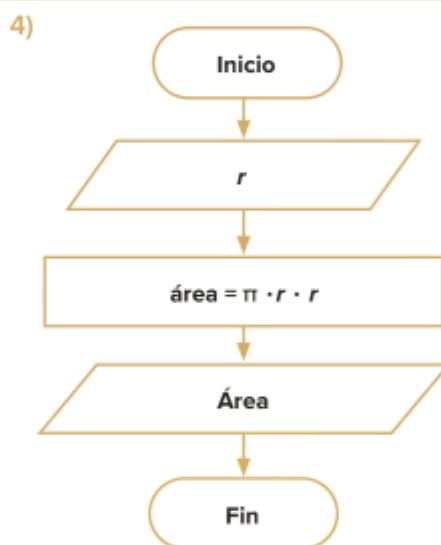
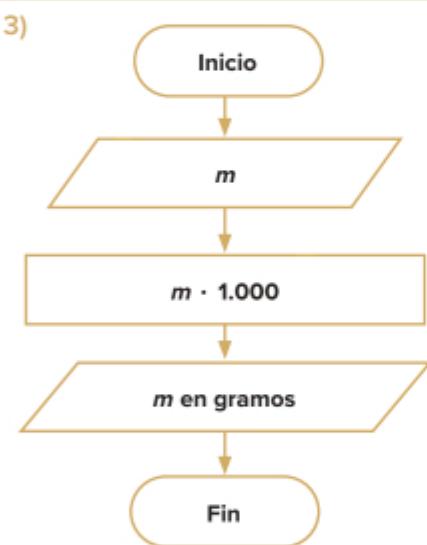
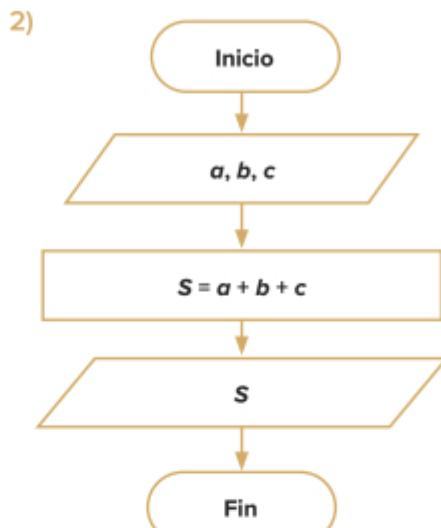
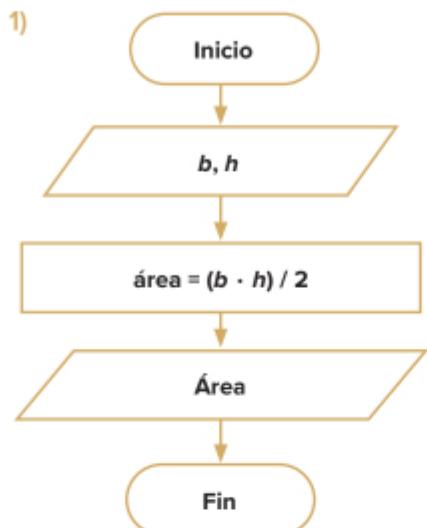
c)



d)



12. ¿Para qué nos sirve cada uno de los siguientes diagramas de flujo?



- a) Calcular el área de un triángulo de base  $b$  y altura  $h$
- b) Pasar una masa  $m$  de kilogramos a gramos
- c) Sumar tres números  $a, b$  y  $c$
- d) Calcular el área de un círculo de radio  $r$

## Programación

◆ 13. Selecciona la respuesta correcta en cada caso:

1) Son instrucciones de la categoría **Control** en Scratch:

- a) detener todos y operaciones
- b) repetir y apariencia
- c) por siempre y movimiento
- d) por siempre y esperar hasta que

2) Son instrucciones de la categoría **Movimiento** en Scratch:

- a) girar 15 grados
- b) mover 10 pasos
- c) apuntar en dirección 90
- d) Todas las respuestas anteriores son correctas.

3) En la categoría **Apariencia** encontramos bloques que nos permiten...

- a) Cambiar de un disfraz a otro.
- b) Apuntar en dirección 90.
- c) Crear variables.
- d) Ninguna de las respuestas anteriores es correcta.

4) En la categoría **Operadores** encontramos bloques que nos permiten...

- a) Realizar operaciones aritméticas.
- b) Unir textos.
- c) Realizar operaciones lógicas.
- d) Todas las respuestas anteriores son correctas.

◆ 14. De los siguientes programas, ¿cuál simulará mejor que un personaje ande?

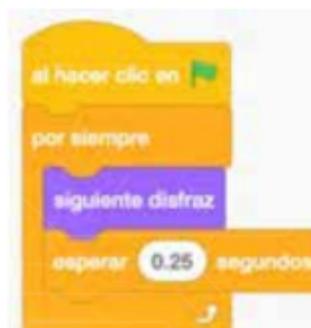
a)



b)



c)

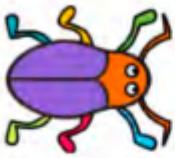


d)



15. Queremos que el escarabajo se mueva describiendo un triángulo. ¿Cuál es el bloque de instrucciones correcto?

a)



```

    when green flag clicked
      [move 100 steps
       turn 120 degrees
       move 100 steps
       turn 120 degrees
       move 100 steps]
  
```

b)

```

    when green flag clicked
      [move 100 steps
       turn 60 degrees
       move 100 steps
       turn 60 degrees
       move 100 steps]
  
```

c)

```

    when green flag clicked
      [move 100 steps
       turn 120 degrees
       move 100 steps
       turn 120 degrees
       move 100 steps]
  
```

16. Para el siguiente diagrama de bloques, señala si las afirmaciones son verdaderas o falsas.



- Hay una variable creada, llamada **Número final**, que va almacenando el valor de la suma.
- El programa suma cuatro números que va escribiendo el jugador.
- El programa finaliza mostrando en pantalla la suma de los números durante 2 segundos.

17. Elige una palabra o grupo de palabras, entre las propuestas, de modo que sea correcta cada una de las afirmaciones siguientes:

- a) El bloque (...) es una estructura iterativa, que permite repetir un bloque de instrucciones hasta que finalice el programa. Si queremos que lo haga un número finito de veces, podemos utilizar el bloque (...) o bien el de repetir hasta que ....

|                    |                |           |                              |                    |
|--------------------|----------------|-----------|------------------------------|--------------------|
| <u>por siempre</u> | <u>esperar</u> | <u>si</u> | <u>esperar hasta que ...</u> | <u>repetir ...</u> |
|--------------------|----------------|-----------|------------------------------|--------------------|

- b) El bloque (...) es una estructura selectiva, que permite llevar a cabo una serie de instrucciones sólo si pasa algo.

|                    |                |                        |                              |                              |
|--------------------|----------------|------------------------|------------------------------|------------------------------|
| <u>por siempre</u> | <u>esperar</u> | <u>si ... entonces</u> | <u>esperar hasta que ...</u> | <u>repetir hasta que ...</u> |
|--------------------|----------------|------------------------|------------------------------|------------------------------|

- c) Hasta que no crees la variable, ésta no te (...) en los bloques de instrucciones correspondientes.

|              |          |         |           |        |
|--------------|----------|---------|-----------|--------|
| desaparecerá | cambiará | variará | aparecerá | sumará |
|--------------|----------|---------|-----------|--------|

- d) Es importante (...) las variables cuando comienza un programa, que significa darle el valor 0. Si no, se quedará el valor que tenía la variable cuando jugaste la última vez.

|       |         |        |            |           |
|-------|---------|--------|------------|-----------|
| sumar | guardar | restar | actualizar | reiniciar |
|-------|---------|--------|------------|-----------|

- e) Para hacer aclaraciones sobre el funcionamiento de un programa puedes incluir (...) en él.

|               |           |             |           |         |
|---------------|-----------|-------------|-----------|---------|
| instrucciones | variables | comentarios | preguntas | bloques |
|---------------|-----------|-------------|-----------|---------|

- f) Las (...) de Scratch son una serie de funcionalidades que te permiten programar tanto hardware como software.

|          |               |           |             |           |
|----------|---------------|-----------|-------------|-----------|
| llamadas | instrucciones | variables | extensiones | preguntas |
|----------|---------------|-----------|-------------|-----------|

#### 18. Completa las frases con las palabras que te ofrecemos:

Palabras: calidad, desarrolladores, integración, requerimientos, validación, verificación

- a) Los procesos de (...) son llevados a cabo por los (...) de un programa y determinan la (...) del software a lo largo de todo el proceso de desarrollo de este.

- b) Los procesos de (...) determinan si el programa final cumple los (...) planteados por el cliente, incluyen pruebas de resultados y valoran la (...) del software con el hardware en el que se implementará.

#### 19. Indica para cada uno de los programas y aplicaciones siguientes si se distribuye como software libre o como software propietario:

- LibreOffice
- GIMP
- WhatsApp
- TikTok
- Adobe Reader

- Microsoft Office
- Google Drive
- Spotify
- Mozilla Firefox
- Netflix