3. Programación

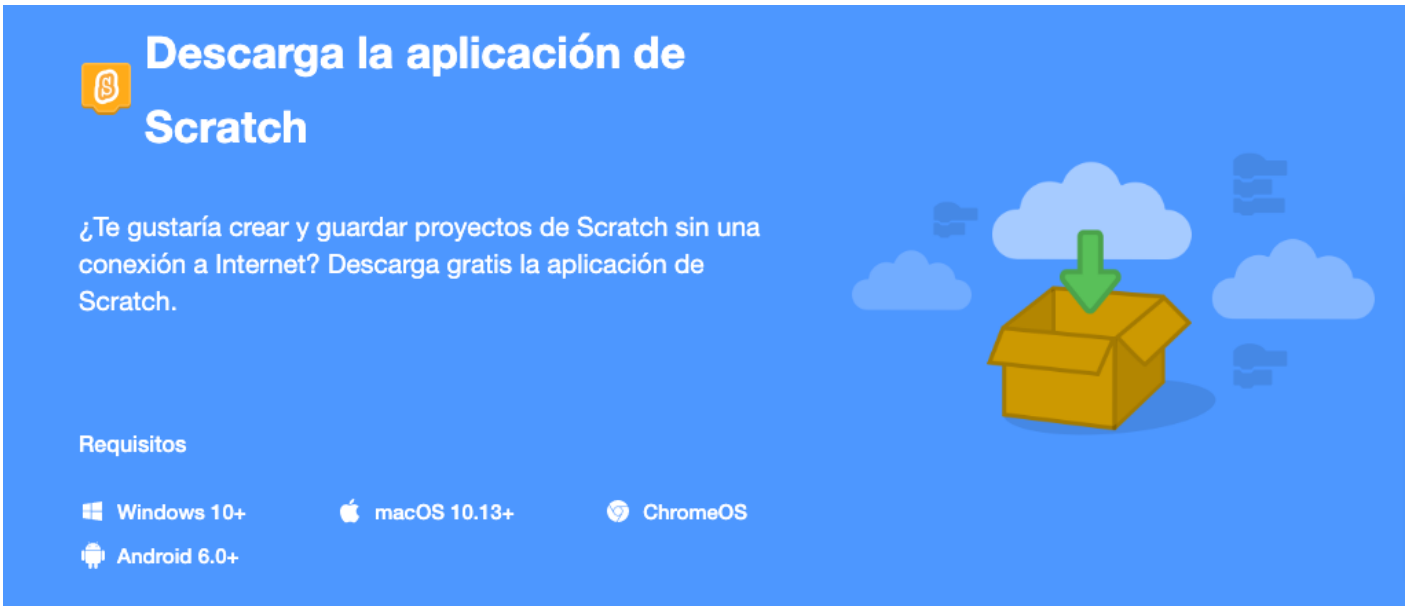
Scratch - Entorno de programación por bloques

Scratch es un entorno de programación de programación por bloques muy visual que fue diseñado por el MIT (Massachusetts Institute of Technology), especialmente orientado a que los niños puedan aprender a programar, creando videojuegos, arte o contando historias que puedan afianzar las habilidades del pensamiento computacional así como estimular su creatividad.

Está muy basado en el sistema de bloques encajables al estilo de LEGO que pueden ser arrastradas a la pantalla de creación de código, lo cual lo convierte en un entorno de programación sencillo e intuitivo ya que tienes al alcance de la vista todas las instrucciones que puedes utilizar, agrupadas por temáticas y cada uno con su propio código de colores.

Para programar en Scratch, tendremos que ir a su página y a partir de aquí tendremos dos opciones:

- crearnos una cuenta y crear todos nuestros programas en la zona de creación que tienen online. De esta forma podremos crear nuestro propio repositorio de programas en Scratch y accesibles desde cualquier lugar con el simple hecho de hacer login con tu cuenta
- descargar el programa como una aplicación más de tu sistema operativo y trabajar en modo offline. Existen versiones para los principales sistemas operativos. Para el caso de linux, y en concreto Vitalinux, la versión offline está disponible en Vitalinux Play con el nombre Scratux. Podéis encontrar instrucciones concretas aquí.



En la página Web de Scratch hay muchos otros recursos para ayudar a aprender Scratch:

- proyectos ya realizados que pueden inspirar tus propias ideas
- tutoriales para empezar desde cero

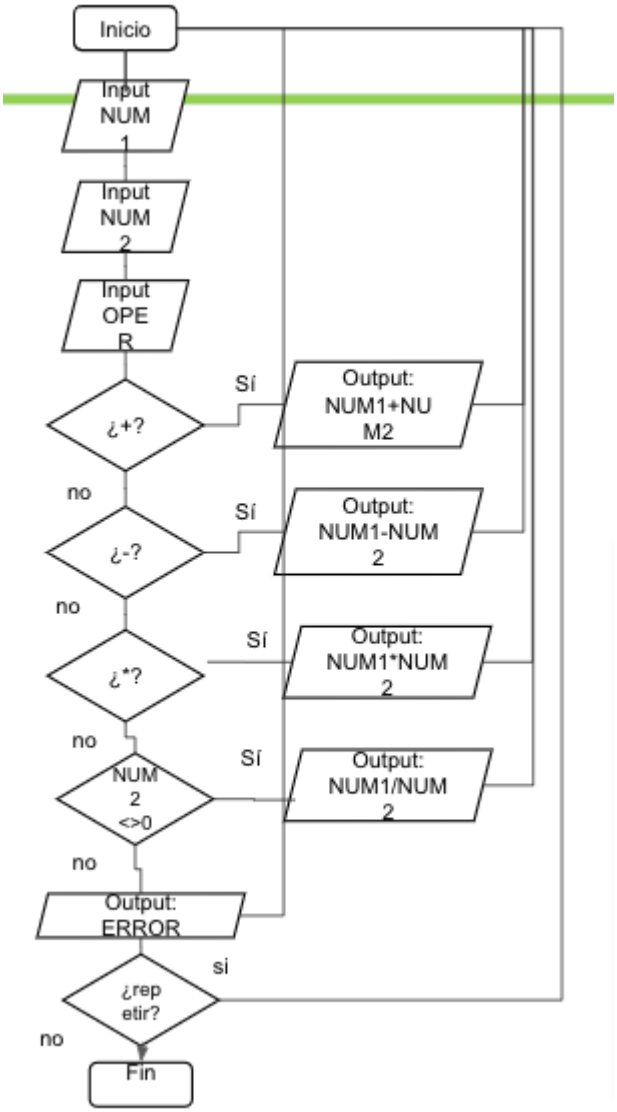
Para conocer más sobre Scratch puedes acudir a los siguientes enlaces, pertenecientes al libro Enseña Pensamiento Computacional con Scratch de **CATEDU**

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Situación de aprendizaje 5. Programando la calculadora básica

Vamos a programar el ejercicio planteado en los diagramas de flujo sobre cómo programar una calculadora básica. Recordamos que solo puede haber las 4 operaciones básicas (+ - * /) y que tenemos que controlar que el segundo número no sea un cero si se ha elegido la división como operación. Tras mostrar el resultado, preguntaremos si quiere seguir jugando.

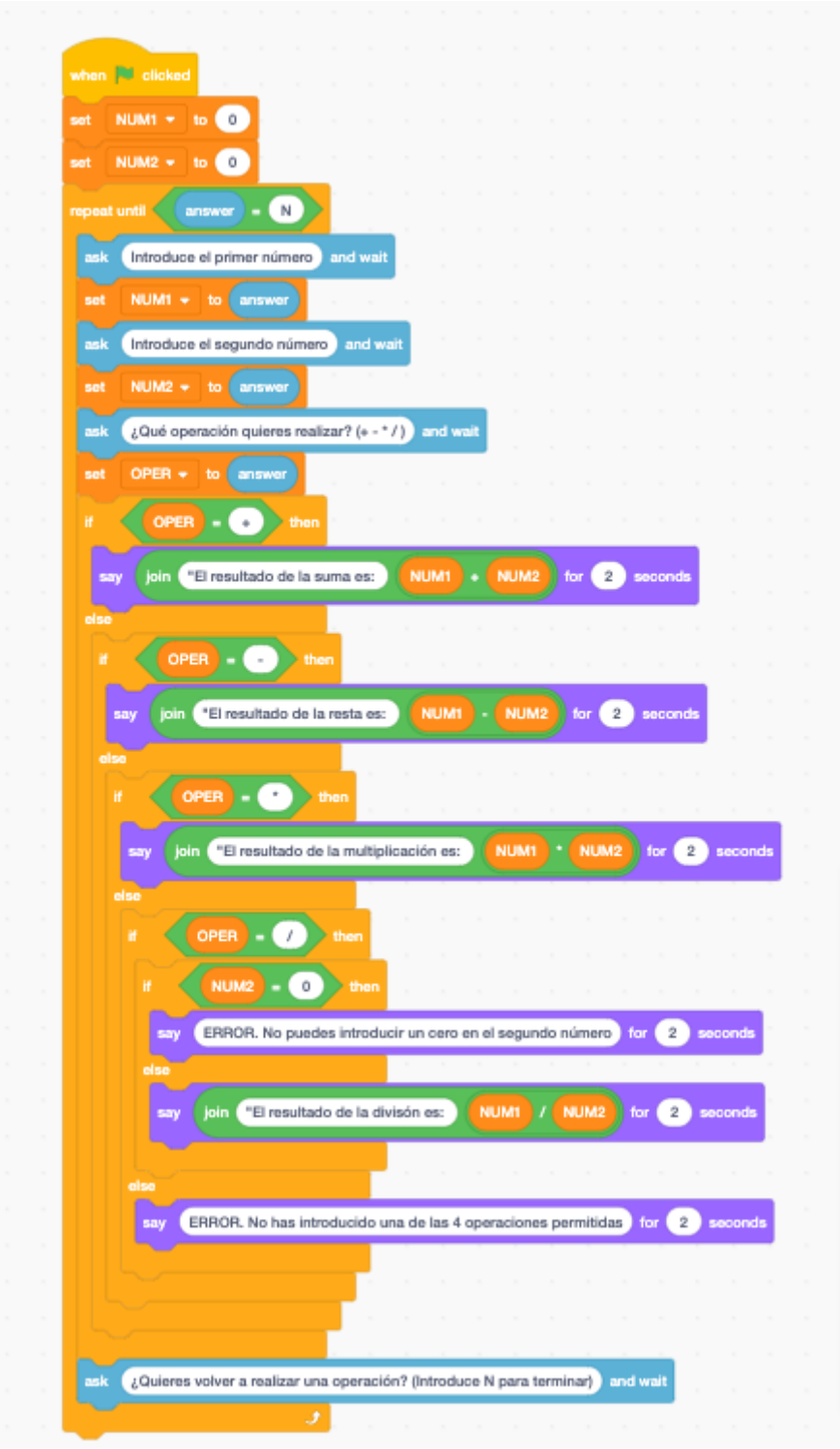
Como primera solución poco rigurosa, al detectar que el segundo número es un cero o si no introduce una de las 4 operaciones deseadas, simplemente sacará un mensaje de error y a continuación preguntará si quiere seguir jugando. El diagrama de flujo sería algo parecido a lo siguiente:



La experiencia de juego debería ser como la siguiente:

https://scratch.mit.edu/projects/743853472/embed

Esta primera resolución del problema tendría un código fuente parecido al siguiente:



Si lo que queremos es asegurarnos que el segundo número no sea un cero en una división y programar la solución para que la ejecución del programa no continue hasta que se introduzca algo diferente a un cero, habría que cambiar el código por lo siguiente:

Y la experiencia de juego sería así:

https://scratch.mit.edu/projects/743866321/embed

EJERCICIO EXTRA PROPUESTO:

Modificar el código propuesto para que el programa solo pueda aceptar como válidas una de las 4 operaciones deseadas y en el caso de la división el programa no continua hasta que el segundo dígito sea distinto de un cero. El programa tampoco continua hasta que se introduzca una operación válida

La experiencia de juego del programa quedaría de la siguiente forma:

https://scratch.mit.edu/projects/743869177/embed

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Situación de aprendizaje 6. Programando las tablas de multiplicar

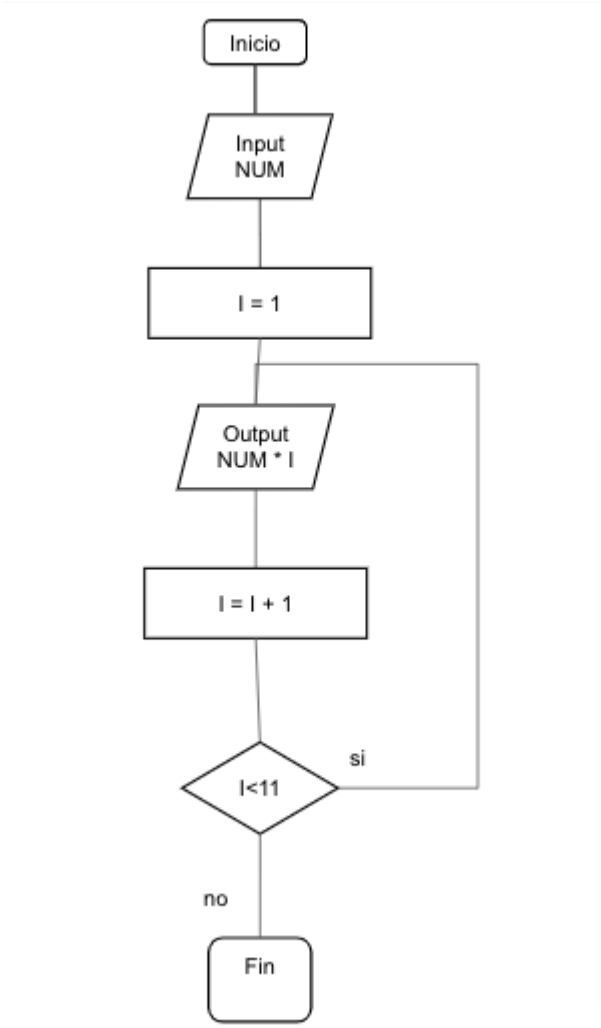
Esta práctica nos viene genial para practicar el uso de bucles junto con variables que se van a tener que ir incrementando para lograr nuestros objetivos.

Podemos plantear dos ejercicios referidos a las tablas de multiplicar:

1. Pedir el número de la tabla de multiplicar que queramos sacar por pantalla. Lo podemos complicar para que el usuario solo pueda meter números del 1 al 10 ambos inclusive y también para que introduzca S o N como únicas respuestas posibles ante la pregunta de si quiere seguir jugando.
2. Crear un programa que automáticamente nos visualice en pantalla las 10 tablas de multiplicar. Este ejercicio nos viene muy bien para practicar el concepto de meter un bucle dentro de otro, algo que al alumnado novel en programación le cuesta entender un poco al principio

Opción 1. Mostrando una tabla de multiplicar elegida por el usuario

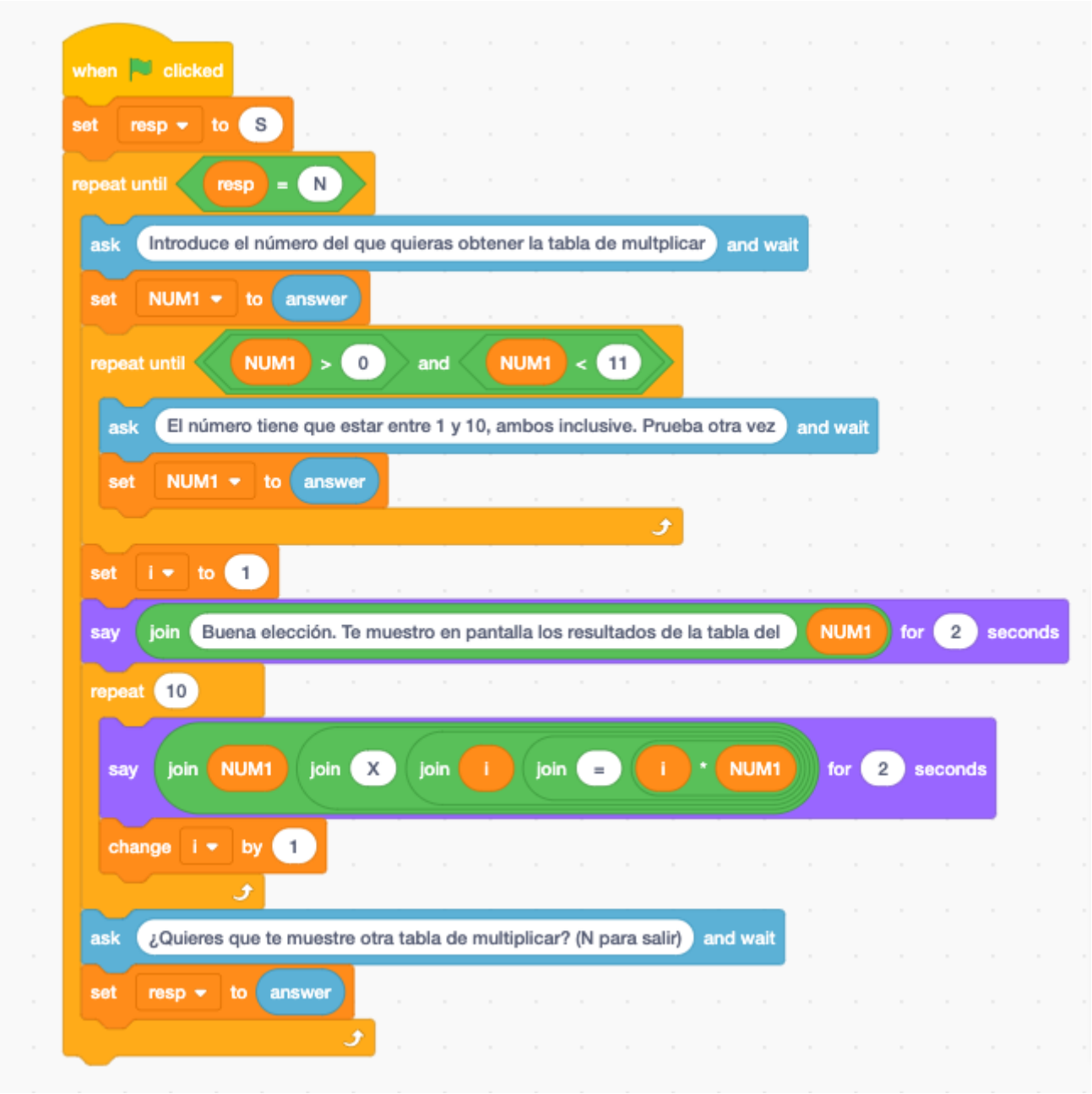
El diagrama de flujo que resuelve este problema sería algo parecido a lo siguiente:



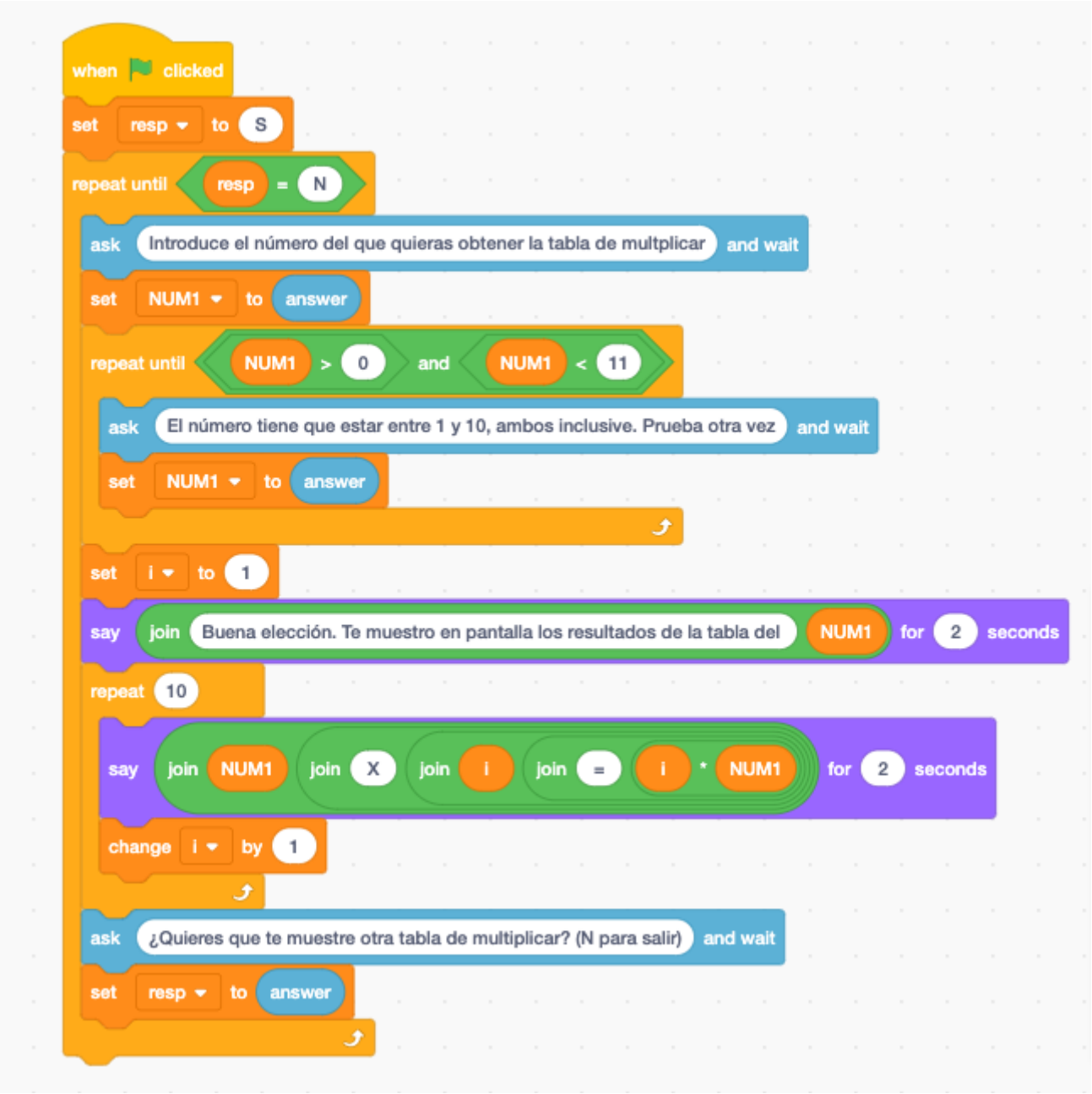
Se recomienda siempre que el alumnado no intente programar el ejercicio en su totalidad, sino que vayan identificando las cosas que tienen que hacer, comiencen por el problema a resolver y posteriormente las particularidades. En este caso, se recomienda que los pasos a la hora de afrontar el algoritmo sean los siguientes:

1. Programar la lógica referente a pedir un número, visualizar su tabla de multiplicar y finalmente preguntar si quiere seguir jugando, todo ello dentro de un bucle principal.
2. Programar el requisito de que el número esté comprendido entre 1 y 10, ambos inclusive
3. Programar que el usuario solo pueda introducir una S o una N ante la pregunta de si quiere seguir jugando.

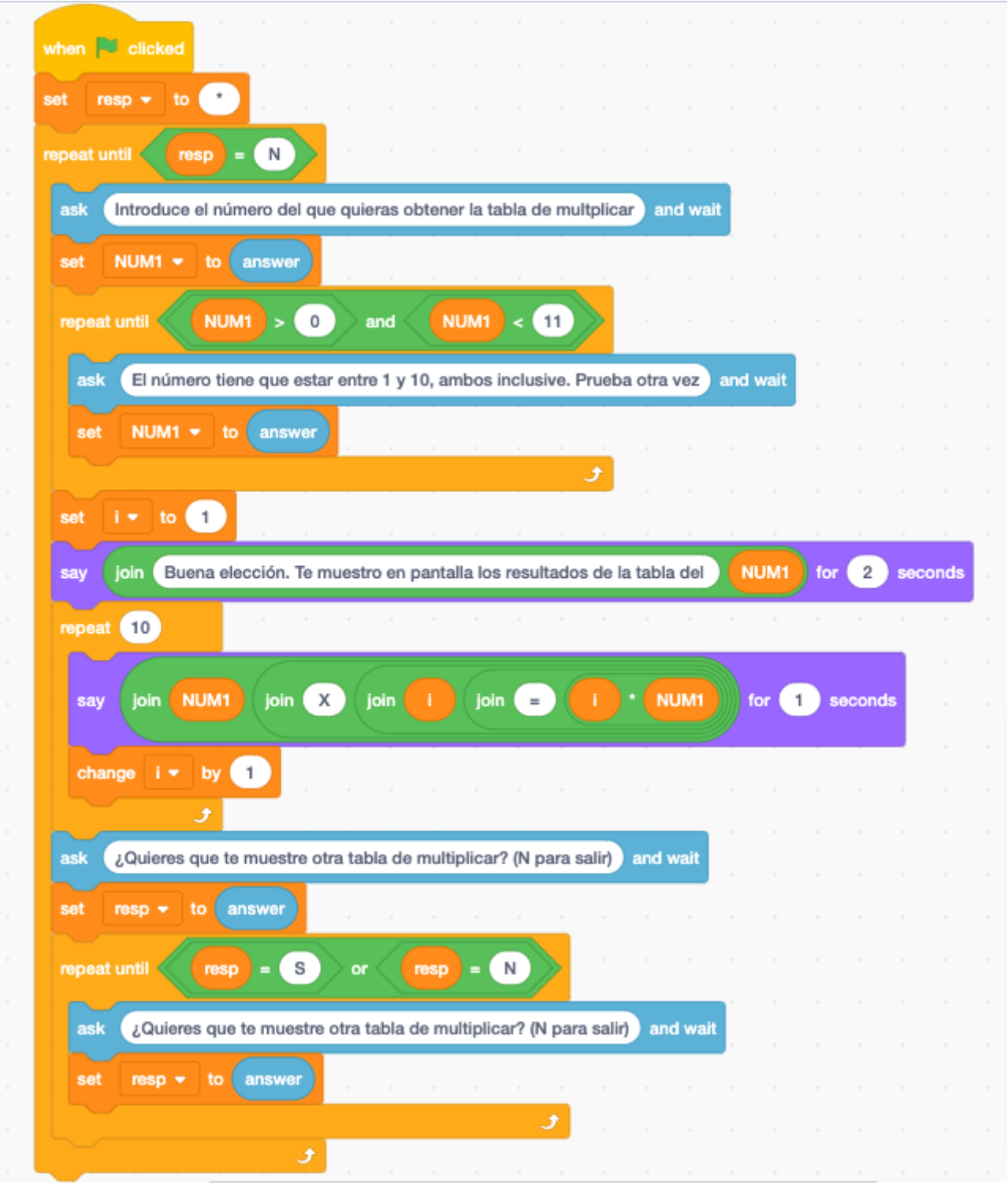
El programa en su fase 1 tendría un código parecido al siguiente:



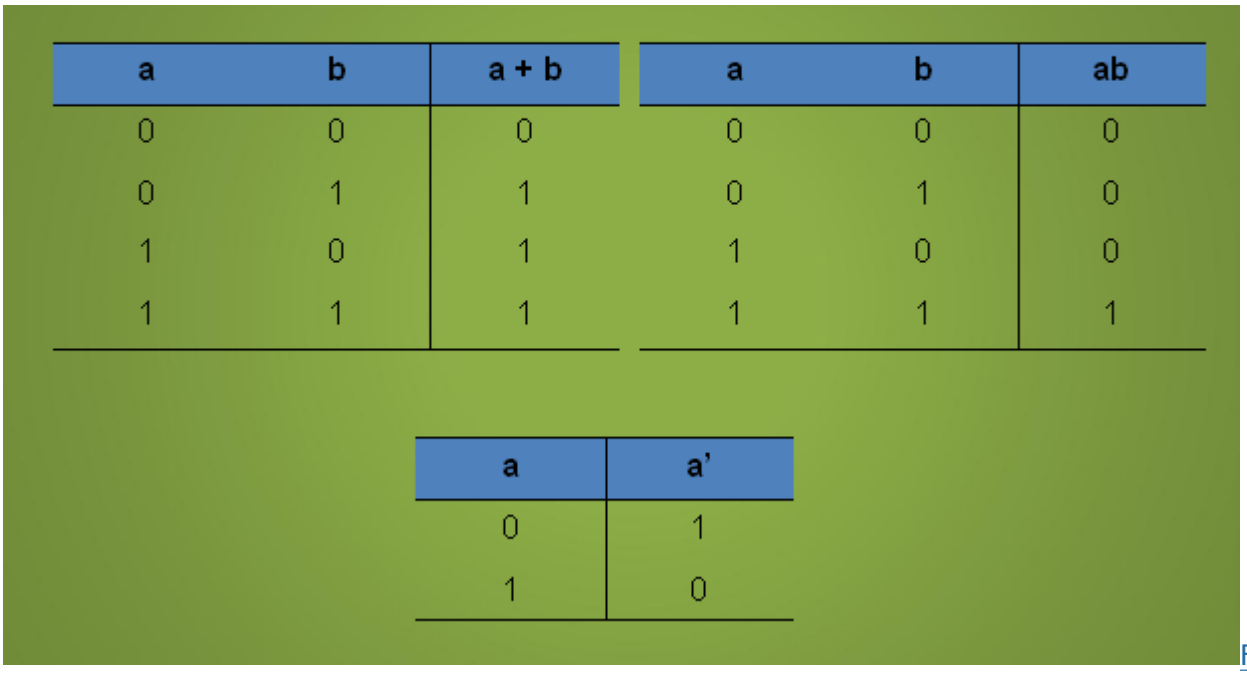
Cuando ya sabemos que nuestro programa funciona en líneas generales, podríamos incluir la programación de la limitación de que hasta que el usuario no introduzco un número entre 1 y 10 ambos inclusive, el programa no continua:



Finalmente, introducimos la programación de que la respuesta final del usuario solo puede ser S o N. Es el mismo tipo de bucle que en la anterior limitación:



Un aspecto importante a destacar es que si bien es cierto que en la programación de ambas limitaciones hemos utilizado el mismo tipo de bucle con dos condiciones a tener en cuenta, en el primero hemos utilizado una unión de esas dos condiciones con el bucle **AND**, puesto que **necesitábamos que ambas se cumplieran**, y en el segundo hemos utilizado un operador **OR** porque **nos bastaba con que se cumpliera una de las dos**. Estaría bien enseñar a los alumnos (si es que aún no lo saben) como son las tablas lógicas del AND y del OR.



a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	ab
0	0	0
0	1	0
1	0	0
1	1	1

a	a'
0	1
1	0

Fuente:

https://www.monografias.com/trabajos104/simplificacion-circuitos-logicos-algebra-conmutacion/img4.png

Este ejercicio, aunque parezca sencillo en su formulación, contiene en su programación unos cuantos aspectos básicos en el aprendizaje de programación que deben quedar bien asentadas para futuros aprendizajes más profundos.

Finalmente, la experiencia de juego sería algo parecida a lo siguiente:

https://scratch.mit.edu/projects/746099667/embed

Propuesta Extra Opción 2. Visualizando todas las tablas de multiplicar del 1 al 10

¿Te atreves a realizar este programa por tu cuenta? Y la experiencia de juego sería como se puede ver a continuación:

https://scratch.mit.edu/projects/746109344/embed

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Situación de aprendizaje 7. Programando el juego de adivina el número secreto

Vamos a realizar un programa donde la máquina va a generar números aleatorios y nosotros lo tenemos que adivinar. Condicionantes del programa:

- El número aleatorio deberá estar en un intervalo de 0 a 100, ambos inclusive.
- El programa no para hasta que lo acertemos
- Tenemos que dar pistas al usuario. Si el número introducido es mayor o menor que el número secreto, deberemos decírselo
- Habrá que mostrar también el mensaje de que has acertado el número
- Habrá que decirle al usuario cuántos intentos le ha costado adivinar el número
- Una vez acertado el número, habrá que preguntar al usuario si quiere seguir jugando y en caso afirmativo deberá crearse un nuevo número aleatorio

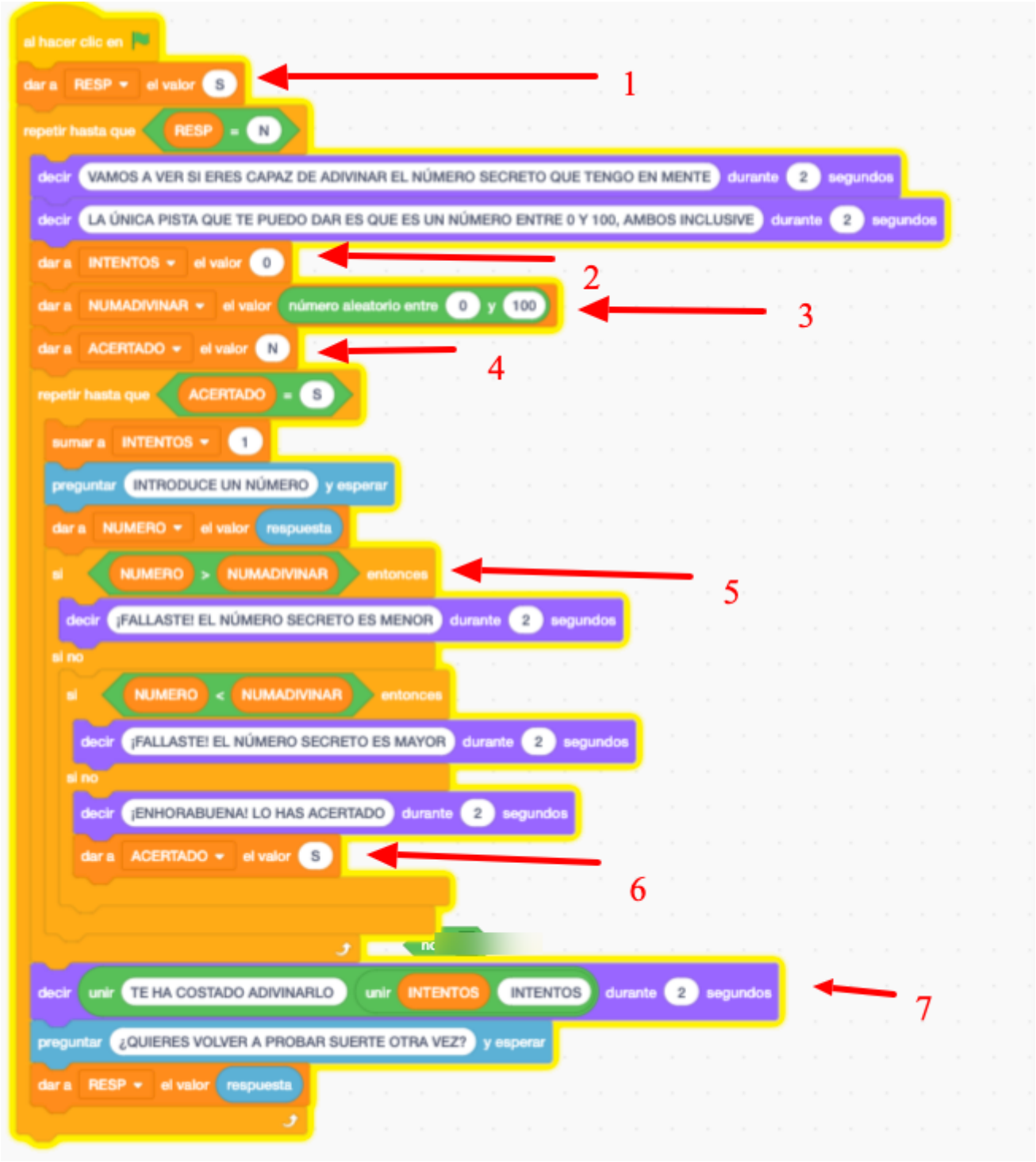
Para crear un número aleatorio tenemos la instrucción PICK RANDOM (inglés) o NÚMERO ALEATORIO ENTRE (español)

SOLUCIÓN COMENTADA

A continuación la solución al programa con una serie de aclaraciones:

1. Hay que iniciar la variable RESP (variable para controlar si queremos seguir jugando o no) con un valor inicial que al menos nos permita entrar la primera vez
2. Antes de comenzar a intentar adivinar el número, tendremos que resetear el contador de INTENTOS a 0
3. Instrucción para generar el número aleatorio
4. Resetear la variable ACERTADO a N, ya que la pondremos a S cuando hayamos acertado

5. Comenzamos con los tres IF para preguntar si es mayor, menor o hemos acertado y dar información al usuario
6. Poner a S la variable ACERTADO en caso de acertar, para romper el bucle de seguir intentándolo
7. Sacar el mensaje al usuario para decirle cuántos intentos ha tardado en adivinar el número. Esto lo haremos al acabar el bucle anterior



Y a continuación la experiencia del juego:

https://scratch.mit.edu/projects/747509834/embed

PROPUESTA DE MEJORA

Introduce las siguientes mejoras al programa:

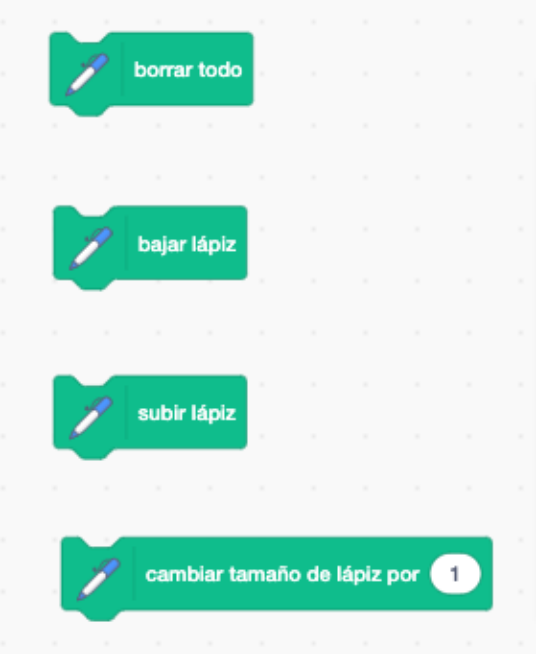
- Ofrecer solo 5 intentos para adivinar el número secreto. El programa acabará bien porque acierta el usuario o bien porque se le acaban los intentos
- Forzar al usuario a que introduzca exclusivamente S o N para seguir jugando o no

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Situación de aprendizaje 8. Art coding

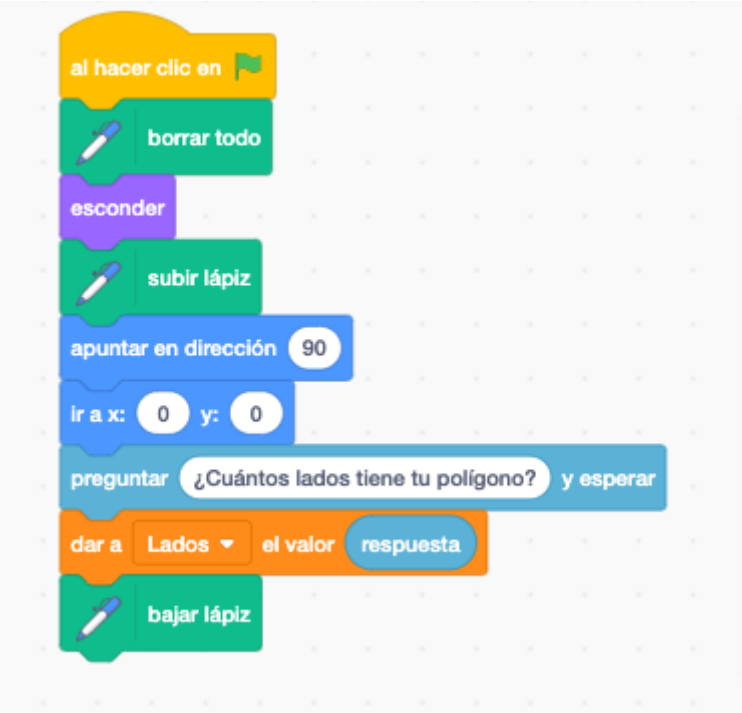
En este programa vamos a dibujar figuras geométricas que se van a ir desplazando por la pantalla para crear formas geométricas que van cambiando de colores.

Para realizar este programa, en primer lugar tenemos que sacar cómo dibujar una figura geométrica. Para ello utilizaremos un grupo concreto de instrucciones del **bloque LÁPIZ:**



Las instrucciones **BAJAR LÁPIZ** y **SUBIR LÁPIZ** hacen que la herramienta de pintar mientras nos movemos aparezca y desaparezca. Por lo tanto, si nos movemos en el escenario y tenemos puesta la instrucción de **BAJAR LÁPIZ**, se irá pintando una línea al mismo tiempo que nos movemos.

Lo primero que deberíamos realizar en nuestro programa es preguntarle al usuario cuántos lados tiene la figura geométrica que desea visualizar en pantalla. Dicho resultado lo guardaremos en una variable porque será la clave para poder generalizar la solución a cualquier número de lados. Hasta este momento, nuestro código debería tener algo parecido como lo siguiente:



Aclaraciones en este punto:

- La instrucción **ESCONDER** la utilizamos para hacer invisible el SPRITE y de esta manera la línea salgo más clara
- La instrucción **APUNTAR EN DIRECCIÓN 90** nos sirve inicialmente para que su movimiento sea recto y dirección hacia la derecha
- La instrucción **IR A X 0 Y 0** nos sirve para situar al SPRITE en el centro del mapa de coordenadas

Las instrucciones básicas para dibujar una figura geométrica serían:

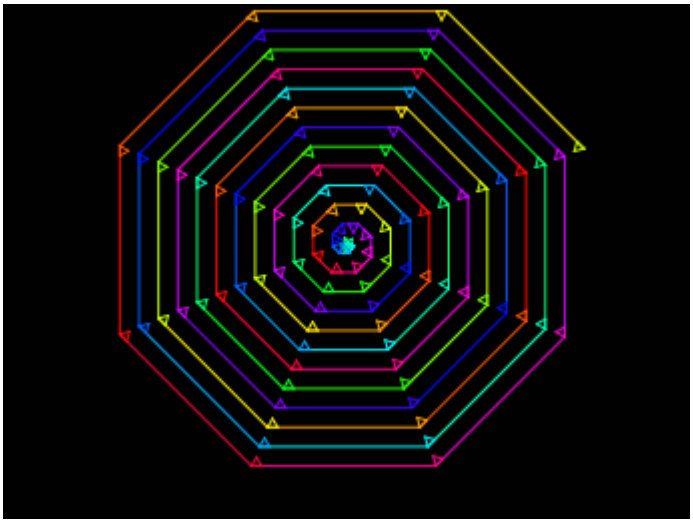
- **GIRAR x GRADOS.** El valor de **x** va a estar directamente relacionado con el número de lados que queremos pintar. Si es un triángulo, para dibujarlo perfecto, habrá que aplicar la fórmula de 360/3, y por lo tanto en cada giro estaremos girando 120 grados, antes de volver a pintar
- **MOVER x PASOS**. El valor de **x** tiene que ver aquí con el tamaño del triángulo que queramos pintar. Un valor de x pequeño, por ejemplo, 5, nos dibujaría un triángulo pequeñito, y un valor de 20, un triángulo más grande.

Como en este curso lo que nos interesa es trabajar las habilidades del pensamiento, tendríamos que buscar una solución que nos sirva para generalizar nuestras instrucciones de programación para que nos sirva para cualquier número de lados. Y para ello tendríamos que ejecutar un bucle que nos sirva para tal propósito, teniendo en cuenta que se deberá de repetir tantas veces como **LADOS** hayamos elegido, y que en su interior tendrá las dos instrucciones indicadas anteriormente. Concretamente, la instrucción del giro, para generalizar, tendremos que incluir la

variables **LADOS** como divisor del número 360.

Debería ser un bucle como el siguiente:

En este punto del programa ya habremos sido capaces de dibujar una figura geométrica con el número de lados que a nosotros nos haya interesado. A partir de aquí vamos a complicarlo un poco más. Vamos a intentar hacer algo como lo siguiente:



En este caso, se ha pintado un triángulo con un tamaño pequeño (5 pasos de movimiento). Nuestro objetivo será ir pintando esta figura geométrica de tal manera que vaya llenando toda nuestra pantalla. Para ello serán necesarias las siguientes acciones:

- Primero pintar nuestra figura
- A continuación hacer un giro de 45 grados
- A continuación desplazarnos una serie de pasos
- Cambiar el color del lápiz
- Que todo esto se repita el número de veces que nos interese, teniendo en cuenta que las dimensiones del escenario en Scratch tienen un límite y si lo superamos nuestra figura se empezará a deformar.

Para ir realizando esta figura que va aumentando su tamaño, la clave va a estar en ir aumentando el número de pasos que damos tras pinta nuestra figura. Para ello, podemos **crear una variable con valor inicial por ejemplo de 1, y cuyo valor vaya aumentando en 1** después de que haya sido pintada la figura. Todo esto debería estar incluido dentro de un bucle **POR SIEMPRE.** Quedaría algo como se puede ver a continuación:

El programa tendría una ejecución parecido a lo siguiente:

https://scratch.mit.edu/projects/747122106/embed

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU