Ejercicios resueltos

Secuenciación de ejercicios

A la hora de plantear ejercicios con nuestro alumnado, con cada concepto nuevo que introduzcamos deberemos respetar una secuencia lógica que vaya poniendo en juego habilidades cognitivas de menor a mayor complejidad, según nos indica la **Taxonomía de Bloom** revisada por Anderson y Krathwohl. Para ello recomendamos:

1. Partir de programas realizados en los que tengan que analizar sus elementos, predecir su comportamiento, comentarlos, encontrar errores, etc... (CONOCER, COMPRENDER)
2. Continuar proponiendo modificaciones a programas proporcionados para personalizarlos (APLICAR)
3. Comparar programas propuestos para solucionar un mismo problema atendiendo a diferentes criterios: funcionalidad, complejidad, cantidad de código empleado... (ANALIZAR, EVALUAR)
4. Por último crear programas sencillos desde cero partiendo de problemas reales. (CREAR)

Proponemos a continuación algunos ejemplos para verlo de forma más concreta.

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

Ejercicios de conocimiento y comprensión

EJEMPLO 1

ENUNCIADO:

Dado el siguiente código:

```
Algoritmo EJEMPLO1
    Definir num1 Como real;
    Definir num2 Como real;
    Escribir "Escribe el primer número:"
    Leer num1;
    Escribir "Escribe el segundo número:"
    Leer num2;
    Si num1<num2 Entonces
        Escribir num2, " es menor que ", num1;
    SiNo
        Si num1>num2 Entonces
            Escribir num1, " es mayor que ", num2;
        SiNo
            Escribir num1, " es igual que ", num2
        Fin Si
    Fin Si
FinAlgoritmo
```

a) Explica para qué sirve este programa.

b) Detecta si existe algún error en el código y justifica por qué.

c) Realiza el diagrama de flujo del mismo.

d) Cópialo en un fichero de PseInt y comprueba si has contestado correctamente los apartados a), b) y c)

e) Introduce los comentarios necesarios para su mejor comprensión.

SOLUCION

a) Es un sencillo programa para comparar dos números introducidos por el usuario.

b) Al reproducirlo con PseInt se detectan dos errores:

```
Algoritmo EJEMPLO1
    Definir num1 Como real;
    Definir num2 Como real;
    Escribir "Escribe el primer número:"
    Leer num1;
    Escribir "Escribe el segundo número:"
    Leer num2;
    Si num1<num2 Entonces
        Escribir num2, " es menor que ", num1;
    SiNo
        Si num1>num2 Entonces
            Escribir num1, " es mayor que ", num2;
        SiNo
            Escribir num1, " es igual que ", num2
        Fin Si
    Fin Si
FinAlgoritmo
```

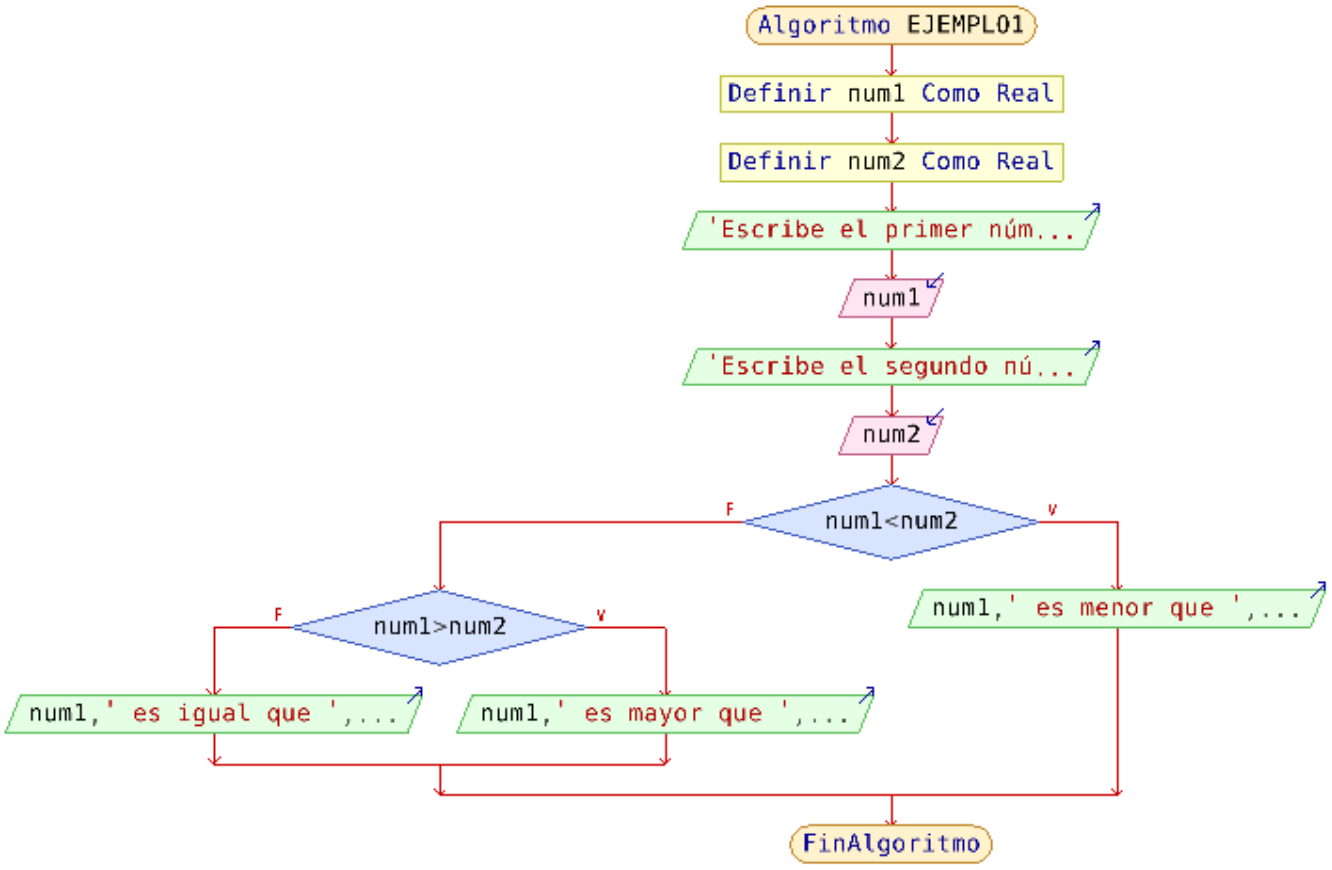
1

2

1. Esto es erróneo, debería poner num1 es menor que num2 para ser cierta la condición de la línea anterior.

2. Falta un punto y coma al final

c) El diagrama de flujo es:



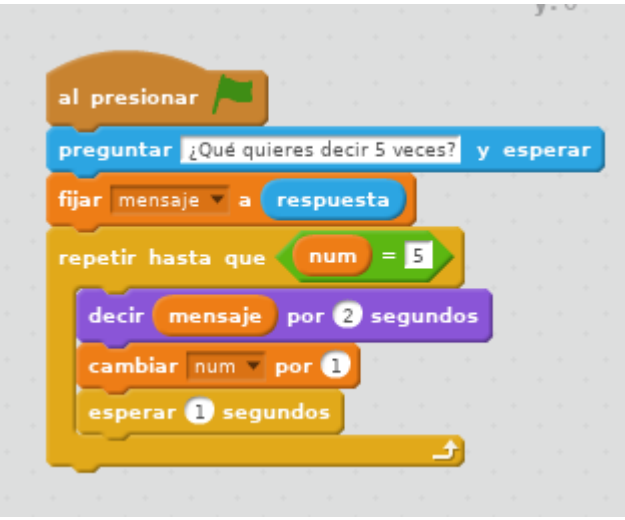
e) Un ejemplo de este programa comentado podría ser:

```
Algoritmo EJEMPLO1
    //defino las variables implicadas
    Definir num1 Como real;
    Definir num2 Como real;

    //Se solicitan los datos
    Escribir "Escribe el primer número:"
    Leer num1;

    Escribir "Escribe el segundo número:"
    Leer num2;

    //realizo la operación de comparación y muestro el resultado
    Si num1<num2 Entonces
        Escribir num1, " es menor que ", num2;
    SiNo
        Si num1>num2 Entonces
            Escribir num1, " es mayor que ", num2;
        SiNo
            Escribir num1, " es igual que ", num2;
        Fin Si
    Fin Si

FinAlgoritmo
```

EJEMPLO 2

ENUNCIADO:

Dado el siguiente diagrama de bloques



a) Dí qué realiza este programa.

b) Identifica en el dibujo qué elementos pertenecen a Entradas, Salidas, Datos u Operaciones.

c) ¿Detectas algún error o imprecisión?

d) Realiza el diagrama de flujo del mismo.

e) Reproduce el programa en Scratch y comprueba tus respuestas anteriores.

f) OPCIONAL: ¿Se te ocurre alguna forma más sencilla de reescribir este programa?

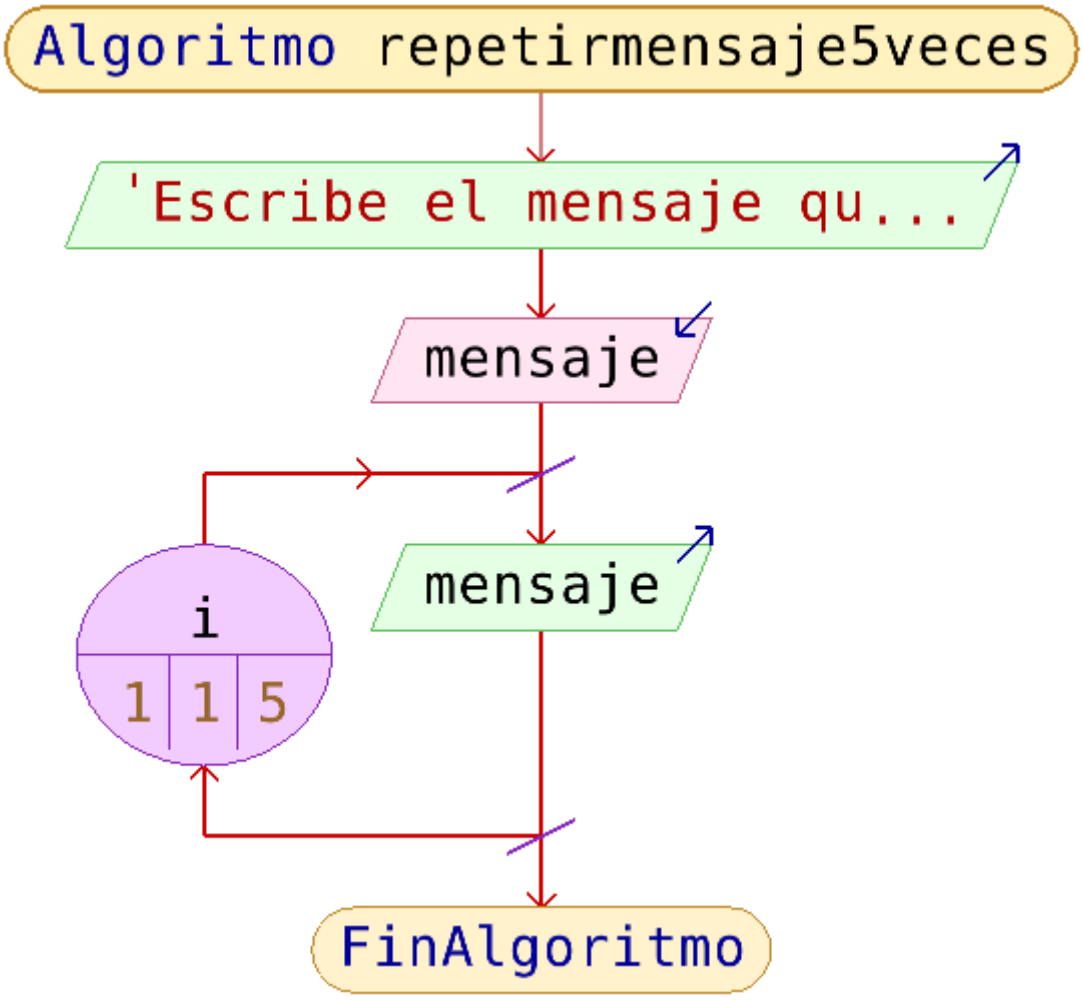
SOLUCION:

a) Es un programa que muestra en pantalla un mensaje elegido por el usuario 5 veces, durante 2 segundos y dejando un espacio entre un mensaje y otro de 1 segundo.

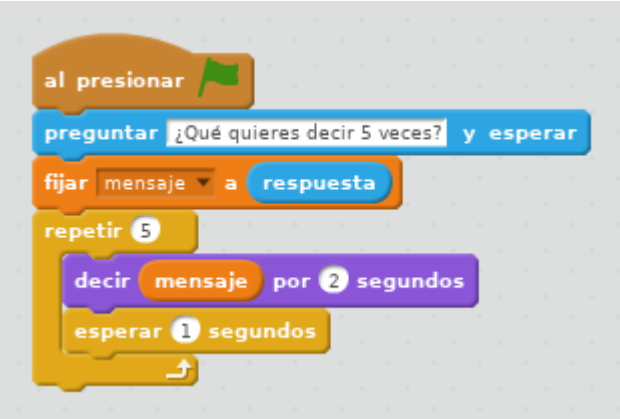
b)

c) El error es que la variable num no tiene un valor inicial predefinido, aunque sería más una imprecisión puesto que si no se asigna valor inicial, se asume el valor 0 que es el que nosotros queríamos poner.

d) Diagrama de flujo:



f) Si realmente se quiere repetir el mensaje 5 veces y no un número determinado por el usuario bastaría con este esquema.



El código inicial solo tiene sentido si el número de veces que se repite también quiere ser personalizado por la persona usuaria.

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

Ejercicios de aplicación

EJEMPLO 1:

ENUNCIADO:

Dado el siguiente código,

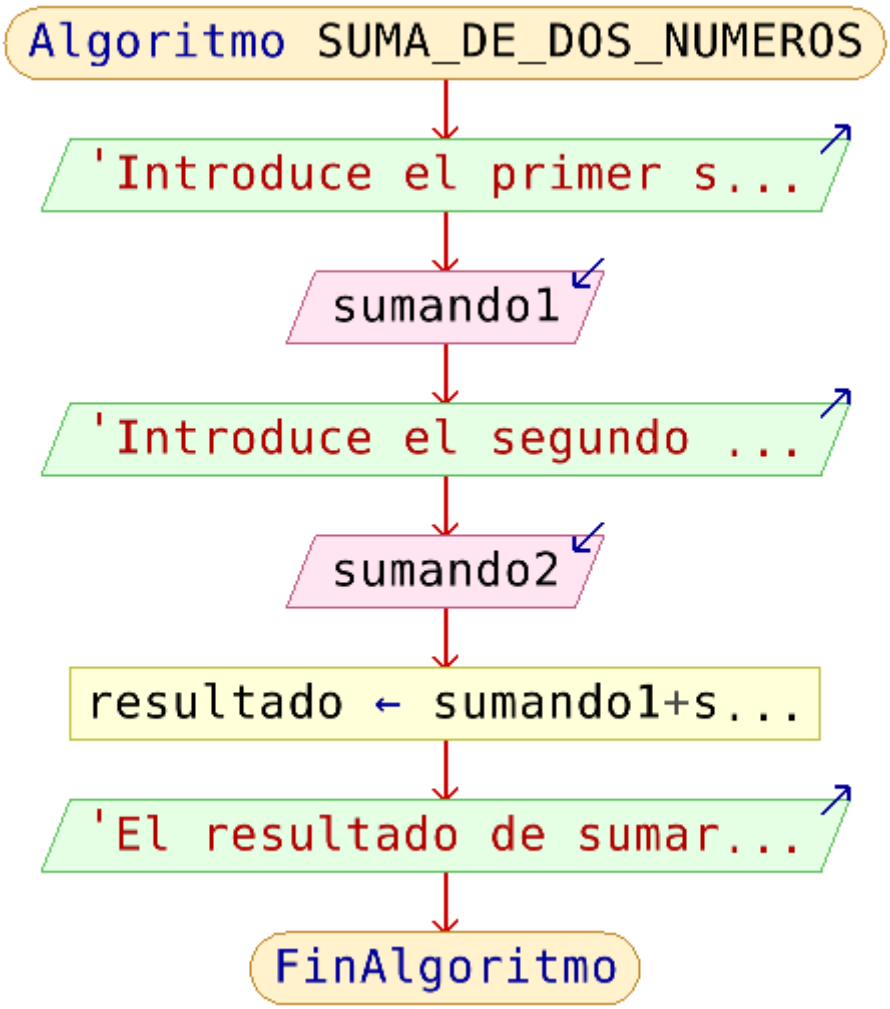
```
Algoritmo SUMA_DE_DOS_NUMEROS
    Escribir "Introduce el primer sumando: "
    Leer sumando1;
    Escribir "Introduce el segundo sumando: "
    Leer sumando2;
    resultado<- sumando1+sumando2;
    Escribir "El resultado de sumar ",sumando1," y ", sumando2, " es ",resultado;
FinAlgoritmo
```

a) Realiza su diagrama de flujo

b) Realiza tres programas similares que realicen la RESTA, PRODUCTO y DIVISIÓN de DOS NÚMEROS dados.

SOLUCIÓN:

a)



b) ALGORITMO RESTA

```
Algoritmo RESTA_DE_DOS_NUMEROS
    Escribir "Introduce el minuendo: "
    Leer minuendo;
    Escribir "Introduce el sustraendo: "
    Leer sustraendo;
    resultado<- minuendo-sustraendo;
    Escribir "El resultado de restar ", minuendo, " menos ",sustraendo, " es: ", resultado;
FinAlgoritmo
```

ALGORITMO PRODUCTO

```
Algoritmo PRODUCTO_DE_DOS_NUMEROS
    Escribir "Introduce el factor1: "
    Leer factor1;
    Escribir "Introduce el factor2: "
    Leer factor2;
    resultado<- factor1*factor2;
    Escribir "El resultado de multiplicar ", factor1, " por ",factor2, " es: ", resultado;
FinAlgoritmo
```

ALGORITMO DIVISIÓN

```
Algoritmo DIVISION_DE_DOS_NUMEROS
    Escribir "Introduce el dividendo: "
    Leer dividendo;
    Escribir "Introduce el divisor: "
    Leer divisor;
    resultado<- dividendo/divisor;
    Escribir "El resultado de dividir ", dividendo, " entre ",divisor, " es: ", resultado;
FinAlgoritmo
```

EJEMPLO 2:

ENUNCIADO

A partir del siguiente diagrama de bloques

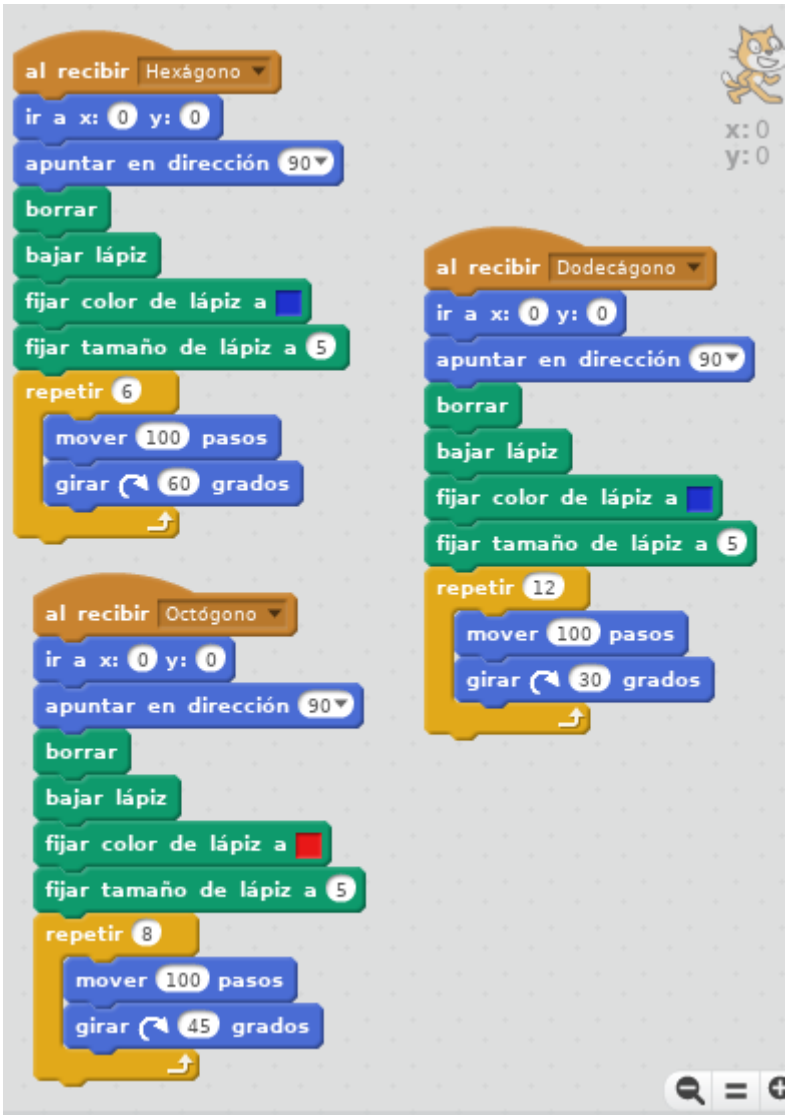
a) Explica qué hace este programa

b) Realiza otros tantos programas que permitan dibujar un hexágono, un octógono y un dodecágono. Elige un color y un grosor deseado y añade las mejoras que quieras.

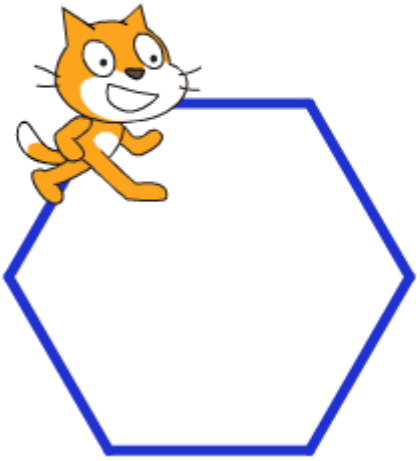
SOLUCIÓN

a) Dibuja un pentágono regular con el color y grosor predeterminado por el programa.

b) Posible código a implementar.



Como mejoras aparte de fijar el color y el tamaño, puede ser el marcar el lugar de inicio del dibujo, e incluso la orientación para asegurarse que siempre se dibuja en el centro del lienzo.



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

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
GOBIERNO DE ARAGON

Ejercicios de análisis y evaluación

EJEMPLO 1

ENUNCIADO:

Compara los siguientes procedimientos para realizar la misma acción

```
Funcion DIVISION_DE_DOS_NUMEROS_opc3 ()
    Escribir "Introduce el dividendo: "
    Leer dividendo;
    Escribir "Introduce el divisor: "
    Leer divisor;
    resultado<- dividendo/divisor;
    Escribir "El resultado de dividir ", dividendo, " entre ",divisor, " es: ", resultado;
Fin Funcion

Funcion DIVISION_DE_DOS_NUMEROS_opc2 (dividendo,divisor)
    resultado<- dividendo/divisor;
    Escribir "El resultado de dividir ", dividendo, " entre ",divisor, " es: ", resultado;
Fin Funcion

Funcion resultado <- DIVISION_DE_DOS_NUMEROS_opc1 ( )
    Escribir "Introduce el dividendo: "
    Leer dividendo;
    Escribir "Introduce el divisor: "
    Leer divisor;
    resultado<- dividendo/divisor;
Fin Funcion
```

Escribe en cada caso como los utilizarías en un programa que a partir de dos números nos devolviera su cociente.

SOLUCIÓN

- Opción 1: Es una función que devuelve un valor, por lo que dentro del programa no se puede invocar por separado sino siempre dentro de una expresión.

```
Funcion resultado <- DIVISION_DE_DOS_NUMEROS_opc1 ( )
    Escribir "Introduce el dividendo: "
    Leer dividendo;
    Escribir "Introduce el divisor: "
    Leer divisor;
    resultado<- dividendo/divisor;
Fin Funcion


Algoritmo DIVISION_DE_DOS_NUMEROS_OPCION1
    res<- DIVISION_DE_DOS_NUMEROS_opc1 ( );
    Escribir "El resultado es: ",res;
FinAlgoritmo
```

Opción 2: Es una función que no devuelve un valor (procedimiento) por lo tanto puede ser invocada por separado, si bien necesita que se le pasen dos parámetros para realizarse, que el programa habrá tenido que proporcionar antes.

```
Funcion DIVISION_DE_DOS_NUMEROS_opc2 (dividendo,divisor)
    resultado<- dividendo/divisor;
    Escribir "El resultado de dividir ", dividendo, " entre ",divisor, " es: ", resultado;
Fin Funcion

Algoritmo DIVISION_DE_DOS_NUMEROS_OPCION2
    Escribir "Introduce el dividendo: "
    Leer dividendo;
    Escribir "Introduce el divisor: "
    Leer divisor;
    DIVISION_DE_DOS_NUMEROS_opc2(dividendo,divisor);
FinAlgoritmo
```

- Opción 3: Es un procedimiento que no tiene parámetros, por lo que los datos habrán de ser recabados por el propio procedimiento.

```
Funcion DIVISION_DE_DOS_NUMEROS_opc3 ()
    Escribir "Introduce el dividendo: "
    Leer dividendo;
    Escribir "Introduce el divisor: "
    Leer divisor;
    resultado<- dividendo/divisor;
    Escribir "El resultado de dividir ", dividendo, " entre ",divisor, " es: ", resultado;
Fin Funcion

Algoritmo DIVISION_DE_DOS_NUMEROS_OPCION3
    DIVISION_DE_DOS_NUMEROS_opc3();
FinAlgoritmo
```

La elección entre unas u otras vendrá determinada por lo que sea más conveniente en el programa. La opción 3 es la que resulta más conveniente cuando una misma expresión se va a utilizar repetidas veces a lo largo del código, mientras que la opción 1 puede resultar conveniente si ese resultado quisiéramos introducirlo en posteriores instrucciones para tomar decisiones o realizar otras operaciones.

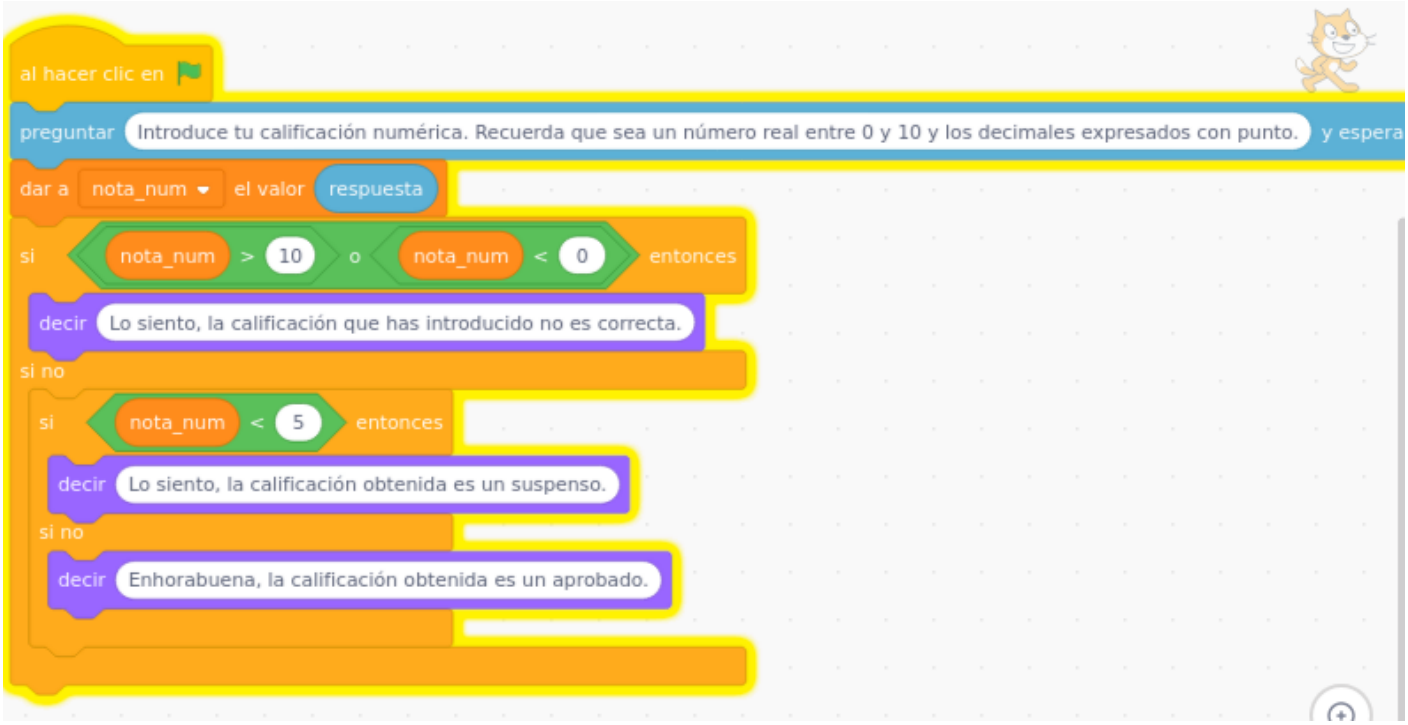
EJEMPLO2

ENUNCIADO

Compara los siguientes programas.

Programa 1

Programa 2



a) ¿Hacen lo mismo?

b) Justifica sus diferencias y explica cuál utilizarías tú y por qué.

SOLUCIÓN

a) Efectivamente hacen lo mismo. Es un programa que al introducir una calificación numérica evalúa que esté en el rango correcto y si no, devuelve un mensaje de error, y si está en el rango correcto evalúa si corresponde a un APROBADO o un SUSPENSO. Es el programa descrito en el último ejemplo del capítulo anterior.

b) La diferencia es:

- El programa 1 **primero evalúa si la nota introducida está en el intervalo correcto** mediante un operador Y, (ha de ser mayor O igual que cero Y menor O igual que 10) y en ese caso devuelve el mensaje de SUSPENSO si es menor que 5 y aprobado en el otro caso. Si no está en el intervalo devuelve un mensaje de error. Como en Scratch no existe el operador <= hay que ir concatenando operadores Y y O y acaba resultando un código complejo.
- El programa 2 **primero evalúa si la nota introducida está fuera del intervalo correcto** (cosa que sucede para valores estrictamente menores que 0 o mayores que 10) y en ese caso devuelve el mensaje de error, y en el contrario anida otro condicional donde devuelve el mensaje de SUSPENSO si es menor que 5 y aprobado en el otro caso.

En el **programa 2** el código es más sencillo y por lo tanto es la opción más recomendable.

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

Ejercicios de creación

EJEMPLO 1

ENUNCIADO

A partir de los ejercicios vistos anteriormente realiza un programa que simule ser una pequeña calculadora con las cuatro operaciones básicas (sumar, restar, multiplicar y dividir):

- Que muestre un menú inicial donde se elija la operación que se va a realizar.
- Si alguien escoge una opción no contemplada debe volver a mostrar las opciones iniciales.
- Que solicite los números implicados en cada operación y muestre el resultado.
- Que de opción a reiniciar el programa una vez realizada una operación.

SOLUCIÓN

Seguimos los mismos pasos vistos en el capítulo anterior en la creación de un programa.

Pasos 1 y 2: Análisis y diagrama de flujo del programa *Calculadora.*

Los elementos implicados son

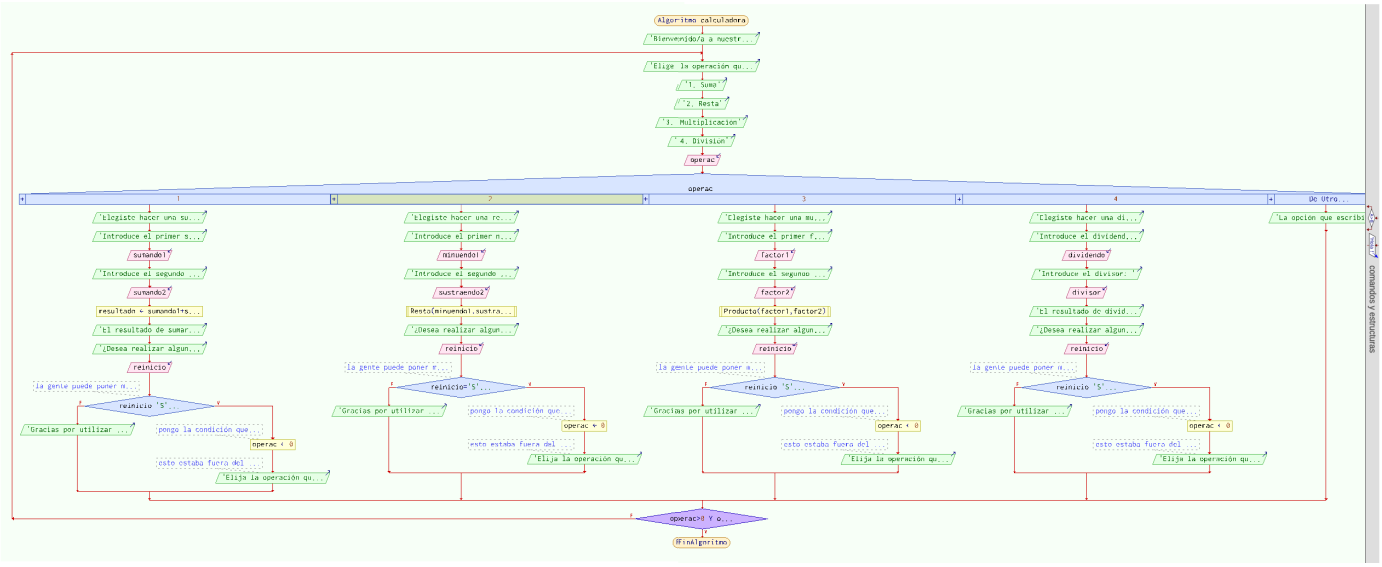
- **Salidas**: Mostrar menú inicial, preguntar por números implicados, mostrar resultado.
- **Entradas**: Operación a realizar (operac), números implicados (sumando1, sumando2....), reinicio.
- **Almacenamiento de datos**: operacion a realizar (operac), números implicados, resultado, reinicio (S/N)
- **Operaciones**: bucle y condicional (SEGUN, SI-ENTONCES, REPETIR hasta que), operaciones lógicas (comparaciones, Y, O), operaciones aritméticas.

En cuanto a la estructura del programa se propone realizar funciones parciales para cada una de las operaciones.

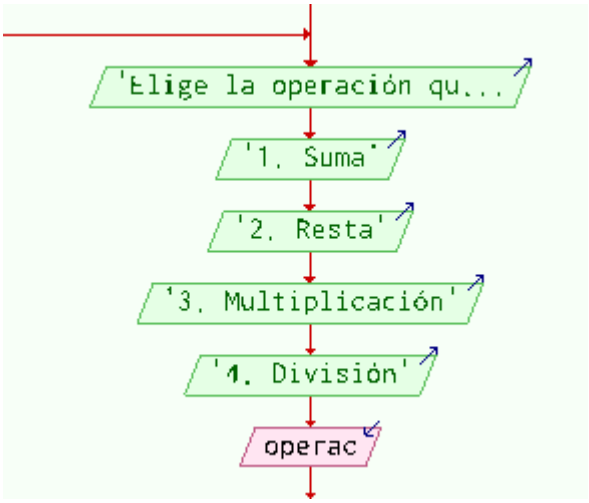
Para mostrar el menú se propone la estructura SEGUN ya que las opciones son discretas (4 operaciones) dentro de una estructura REPETIR hasta que la opción seleccionada sea la correcta.

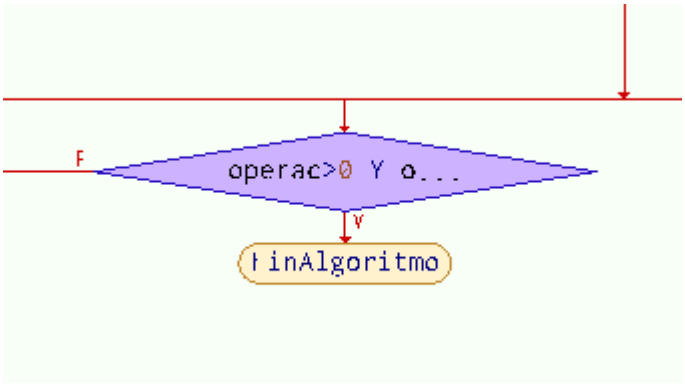
El reinicio se realiza al finalizar cada operación, preguntando y si la respuesta es afirmativa, modificar la condición que volvía a mostrar el menú inicial (operac).

Diagrama de flujo:



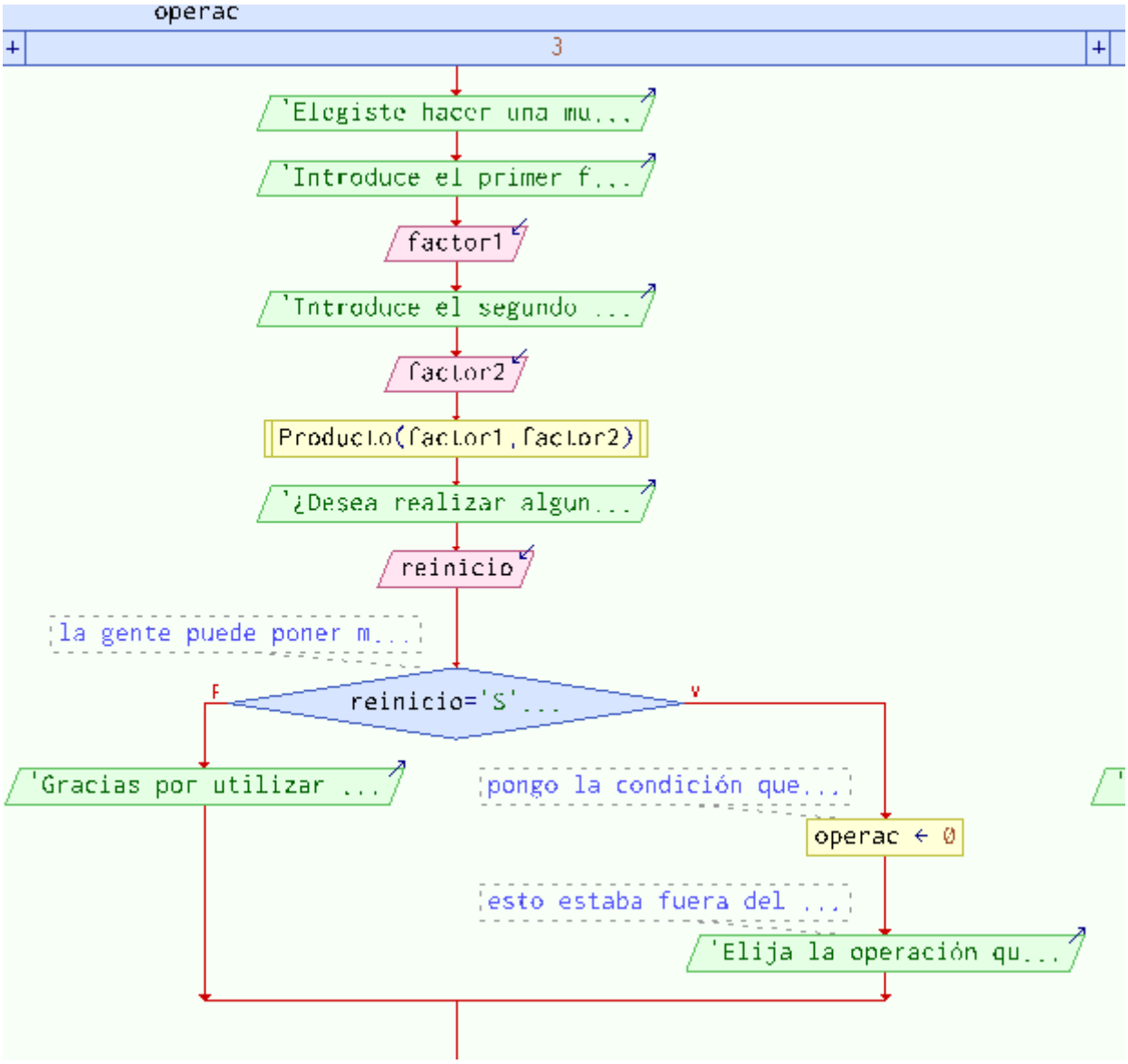
Las dimensiones del diagrama no permiten distinguirlo bien, pero sí notar que hay una estructura REPETIR HASTA QUE la variable operac tenga un valor comprendido entre 1 y 4, que muestra el mensaje inicial con las operaciones a elegir.





Dentro de ella una estructura SEGÚN que abre 5 opciones posibles, la última la de que se haya elegido un número distinto del 1 al 4 que vuelve a iterar el proceso, y las otras 4 una por cada operación.

También que cada una de las 4 operaciones termina con un bloque condicional para preguntar por el reinicio, y en caso de querer reiniciar, le da a la variable operac un valor diferente de 1,2, 3 y 4 lo que reactiva el bucle repetir hasta que.



Pasos 3, 4 y 5: Codificación, compilación y verificación del programa *Calculadora* con *PSeInt.*

Empezamos por definir las funciones. La suma la haremos directamente dentro del algoritmo principal para ver la diferencia.

```
Funcion resultado <- Division ( a,b )
    resultado<- a/b;
Fin Funcion

Funcion Resta ( a,b )
    resultado<-a-b;
    Escribir "El resultado de restar ",a," y ", b, " es ",resultado;
Fin Funcion

Funcion Producto ( a,b )
    resultado<-a*b;
    Escribir "El resultado de multiplicar ",a," por ", b, " es ",resultado;
Fin Funcion
```

El algoritmo principal quedaría de la siguiente forma:

```
Algoritmo calculadora
    Escribir "Bienvenido/a a nuestra calculadora"
    Repetir
        Escribir "Elige la operación que quieres realizar";
        Escribir "1. Suma";
        Escribir "2. Resta"
        Escribir "3. Multiplicación";
        Escribir "4. División";
        Leer operac;
        Segun operac Hacer
            1:
                Escribir "Elegiste hacer una suma.";
                Escribir "Introduce el primer sumando: "
                Leer sumando1;
                Escribir "Introduce el segundo sumando: "
                Leer sumando2;
                resultado← sumando1+sumando2;
                Escribir "El resultado de sumar ",sumando1," y ", sumando2, " es ",resultado;
                Escribir "¿Desea realizar alguna otra operación?(S/N)"
                Leer reinicio;
                Si reinicio='S'y reinicio='s' Entonces //la gente puede poner mayúsculas o minúsculas
                    operac←0; //pongo la condición que me vuelve a meter dentro del bucle repetir
                    Escribir "Elija la operación que desee realizar" //esto estaba fuera del bucle repetir, hay que ponerlo
                SiNo
                    Escribir "Gracias por utilizar el programa";
                Fin Si
```

```
        2:
            Escribir "Elegiste hacer una resta.";
            Escribir "Introduce el primer numero: "
            Leer minuendo1;
            Escribir "Introduce el segundo numero: "
            Leer sustraendo2;
            Resta(minuendo1, sustraendo2);
            Escribir "¿Desea realizar alguna otra operación?(S/N)"
            Leer reinicio;
            Si reinicio='S'∨ reinicio='s' Entonces //la gente puede poner mayúsculas o minúsculas
                operac←0; //pongo la condición que me vuelve a meter dentro del bucle repetir
                Escribir "Elija la operación que desee realizar" //esto estaba fuera del bucle repetir, hay que ponerlo
            SiNo
                Escribir "Gracias por utilizar el programa";
            Fin Si
        3:
            Escribir "Elegiste hacer una multiplicación.";
            Escribir "Introduce el primer factor: "
            Leer factor1;
            Escribir "Introduce el segundo factor: "
            Leer factor2;
            Producto(factor1,factor2);
            Escribir "¿Desea realizar alguna otra operación?(S/N)"
            Leer reinicio;
            Si reinicio='S'∨ reinicio='s' Entonces //la gente puede poner mayúsculas o minúsculas
                operac←0; //pongo la condición que me vuelve a meter dentro del bucle repetir
                Escribir "Elija la operación que desee realizar" //esto estaba fuera del bucle repetir, hay que ponerlo
            SiNo
                Escribir "Gracias por utilizar el programa";
            Fin Si

        4:
            Escribir "Elegiste hacer una división.";
            Escribir "Introduce el dividendo: "
            Leer dividendo;
            Escribir "Introduce el divisor: "
            Leer divisor;
            Escribir "El resultado de dividir ",dividendo," entre ", divisor, " es ",Division(dividendo,divisor);
            Escribir "¿Desea realizar alguna otra operación?(S/N)"
            Leer reinicio;
            Si reinicio='S'∨ reinicio='s' Entonces //la gente puede poner mayúsculas o minúsculas
                operac←0; //pongo la condición que me vuelve a meter dentro del bucle repetir
                Escribir "Elija la operación que desee realizar" //esto estaba fuera del bucle repetir, hay que ponerlo
            SiNo
                Escribir "Gracias por utilizar el programa";
            Fin Si
        De Otro Modo:
            Escribir "La opción que escribiste no es correcta. Elige otra."
    Fin Segun

Hasta Que operac>0 ∧ operac<5;


FinAlgoritmo
```

EJEMPLO2

ENUNCIADO

Crea un programa en Scratch para aprender las tablas de multiplicar. Las condiciones del programa son:

- Al inicio un personaje pregunta si quieres participar o no. Tiene que seguir preguntando hasta que se le diga que sí.
- Después preguntará del 1 al 10 qué tabla de multiplicar quiere practicar.
- A continuación mostrará en pantalla las operaciones de la tabla de multiplicar del número introducido.
- Al terminar se despedirá.

SOLUCIÓN

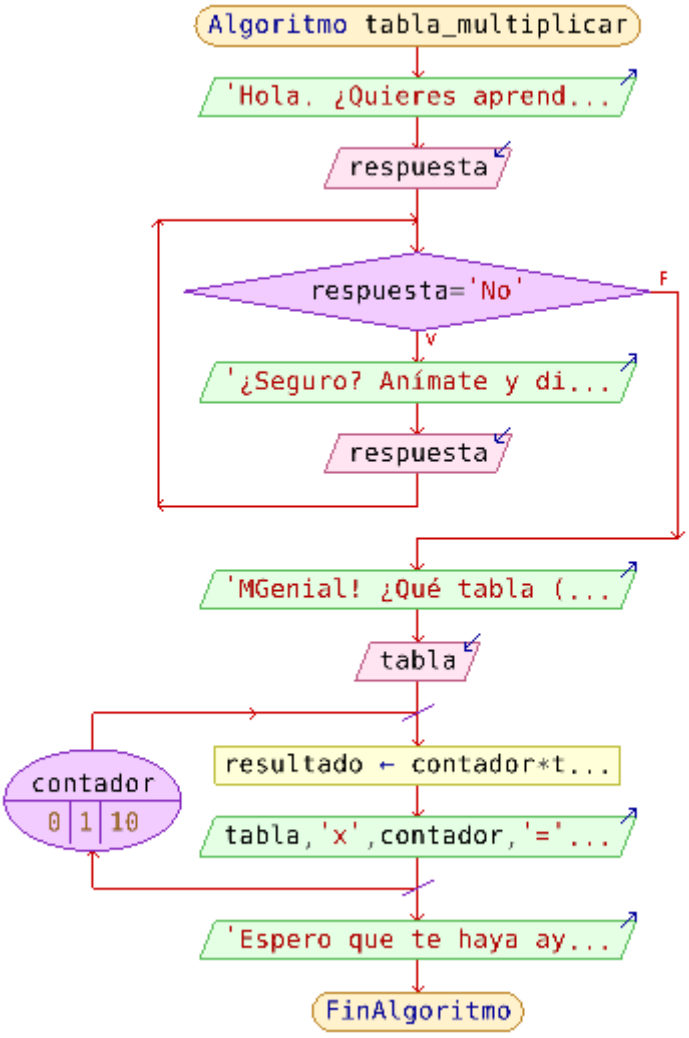
Seguimos los mismos pasos vistos en el capítulo anterior en la creación de un programa.

Pasos 1 y 2: Análisis y diagrama de flujo del programa *Tabla de multiplicar*

Los elementos implicados son

- **Salidas**: Saludar, preguntar si se participa, preguntar la tabla a recitar, recitar la tabla, saludar al final.
- **Entradas**: Respuesta Sí/No a participar, número de la tabla.
- **Almacenamiento de datos**: almacenamos las respuestas obtenidas, y una variable contador para recorrer la tabla del 0 al 10
- **Operaciones**: bucle condicional (repetir mientras la respuesta sea no), bucle con número prefijado (multiplicar del 0 al 10), multiplicaciones, comparaciones, concatenaciones.

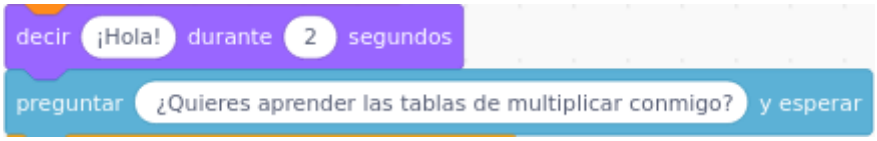
Diagrama de flujo:



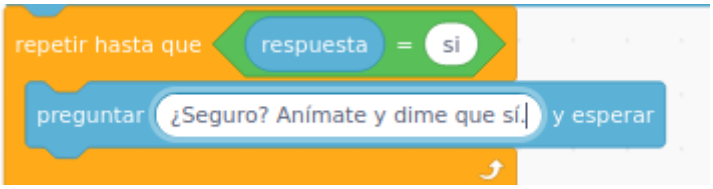
Pasos 3, 4 y 5: Codificación, compilación y verificación del programa *Tabla de multiplicar* con Scratch

Como siempre empezamos por definir las variables. Definimos una variable contador y le asignamos el valor 0 al inicio del programa. La otra variable no será necesario crearla puesto que al ser respuesta de una pregunta, la crea el programa por defecto.

Saludamos y realizamos la pregunta con los bloques correspondientes de **Apariencia** y **Sensores** ya que son **Salidas**.

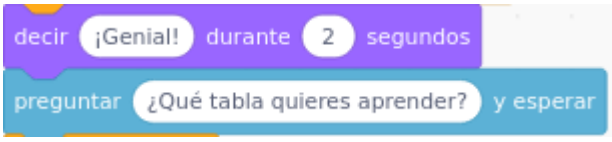


Ahí comienza el primer bucle, en este caso condicional puesto que depende de la respuesta obtenida. En scratch el bloque que realiza esta operación es **Repetir hasta que**

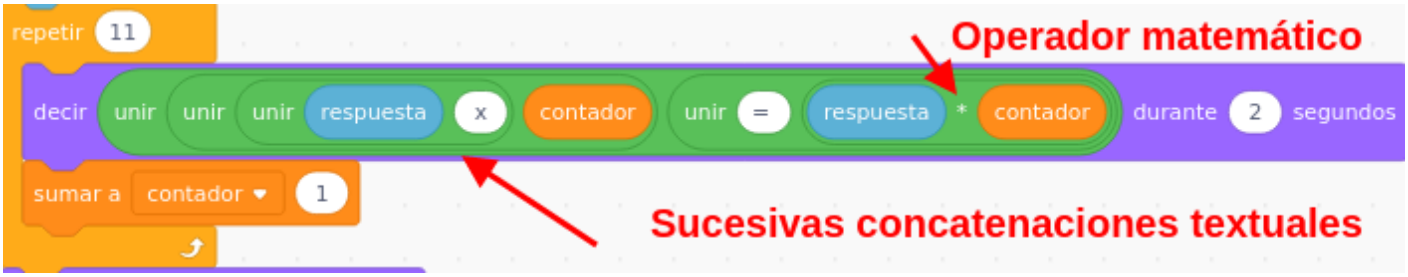


> Cuando realizamos comparaciones con textos, hay que tener en cuenta las variaciones de mayúsculas, minúsculas, tildes, etc...Lo mejor en ese caso sería usar el operador "O" que nos admite como válidas cualquiera de las posibilidades (Sí, Si, SI, si)

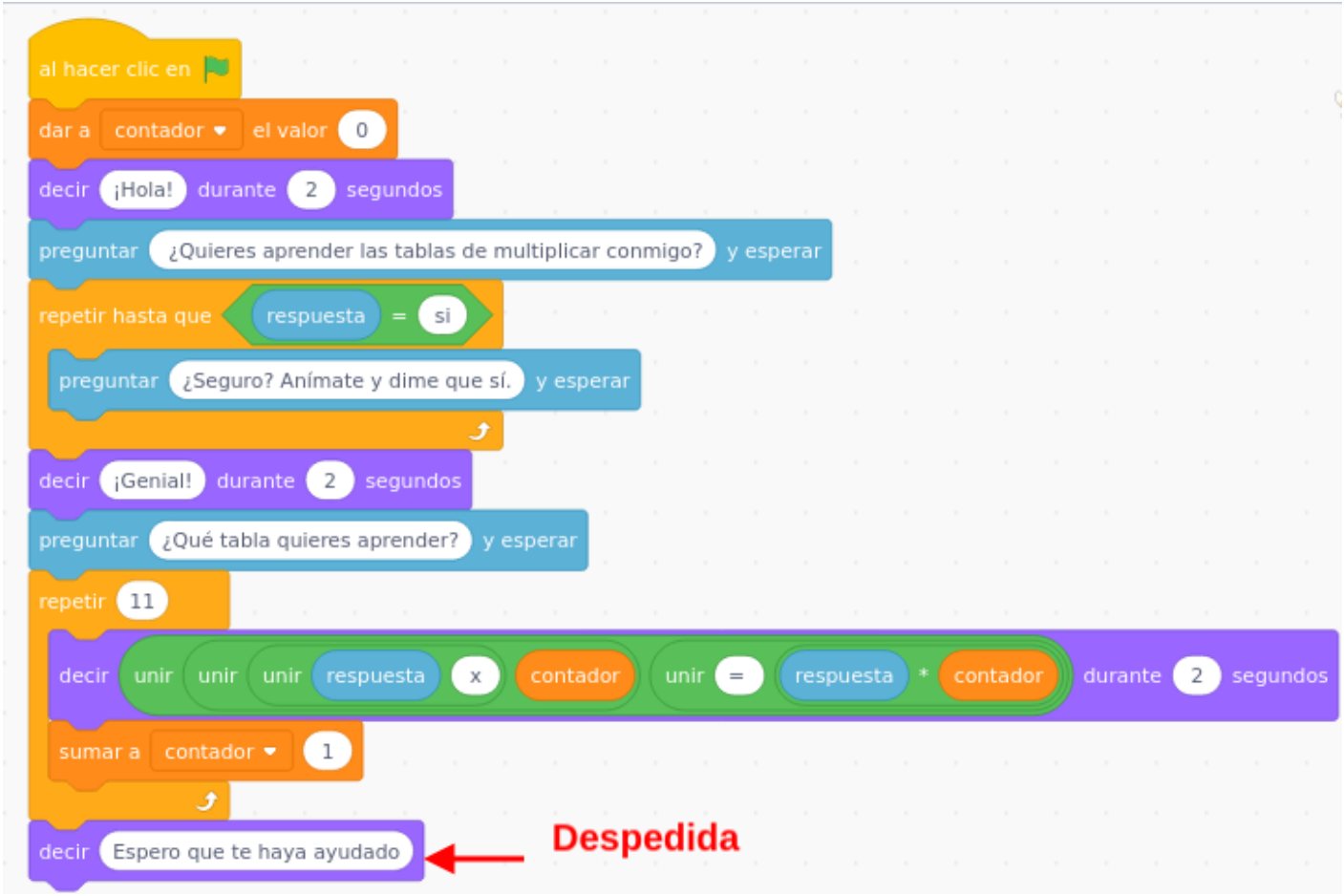
Una vez obtenida la respuesta afirmativa (ENTRADA), pasamos a preguntar de nuevo por el número del 1 al 10 del que se quiere usar la tabla (SALIDA).



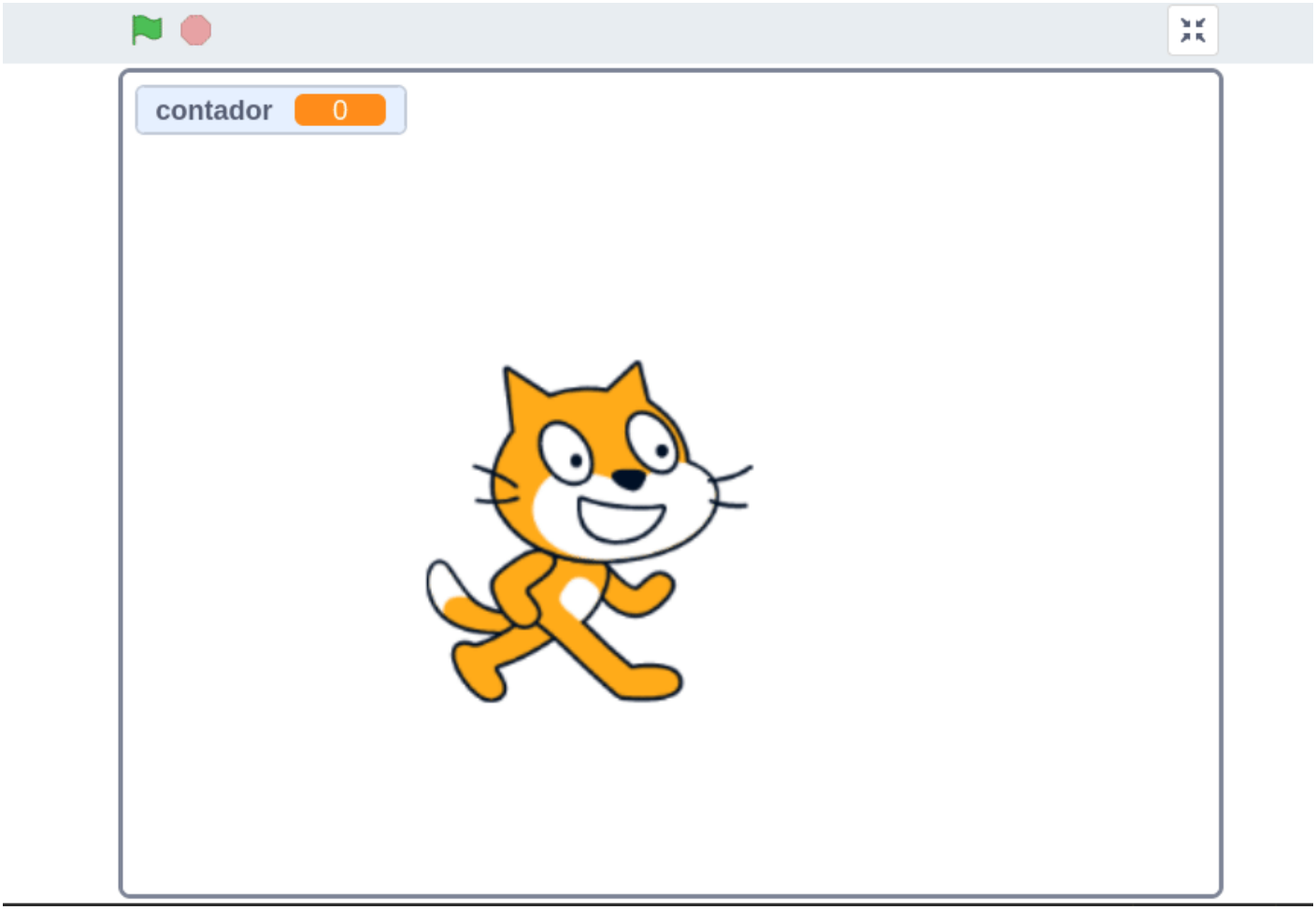
Con la respuesta obtenida (ENTRADA) realizamos otro bucle en este caso del 0 al 10 por lo que son 11 veces, donde se va a mostrar en pantalla al personaje diciendo el numero de la tabla x contador = resultado y el valor del contador se va incrementando de uno en uno en cada iteración.



Tras una última salida con la despedida, el programa quedaría así.



Por último vendría el momento de verificación del funcionamiento.



Consideraciones finales:

- Este programita se podía haber hecho de muchas otras formas, como casi todos los programas. Se pueden utilizar estructuras condicionales anidadas, por ejemplo. Te invitamos a que explores algunas de ellas.
- Se ha escogido esta forma porque es una de las que menos bloques de código requiere. Es una buena práctica para nuestro alumnado exponerle a crear un programa de distintas formas y hacer un análisis crítico de cada una de ellas.
- Es bueno tener presente siempre en programación las siglas DRY (Don't Repeat Yourself) Si se repiten muchas líneas de código o muchos bloques iguales, probablemente hay una forma más eficiente de hacerlo.

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

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
GOBIERNO DE ARAGON

Y la LOMLOE "pa cuándo"?

En la nueva normativa educativa publicada el año 2022 se impulsa de forma decisiva la enseñanza de contenidos relacionados con la programación tanto desde las primeras etapas como por supuesto en Secundaria y Bachillerato.

En concreto los contenidos abordados en este curso corresponden a la Competencia Específica nº 5 de la materia de Tecnología y Digitalización desarrollada en la **Orden ECD/1172/2022, de 2 de agosto, por la que se aprueban el currículo y las características de la evaluación de la Educación Secundaria Obligatoria .** En concreto esta competencia consiste en "*Desarrollar algoritmos y aplicaciones informáticas en distintos entornos, aplicando los principios del pensamiento computacional e incorporando las tecnologías emergentes, para crear soluciones a problemas concretos, automatizar procesos y aplicarlos en sistemas de control o en robótica.*"

Específicamente en los criterios de evaluación de dicha competencia, para el curso de 2º de ESO, se habla de:

- *5.1. Describir, interpretar y diseñar soluciones a problemas informáticos a través de algoritmos básicos y diagramas de flujo sencillos, aplicando los elementos y técnicas de programación de manera creativa.*
- *5.2. Programar aplicaciones sencillas, de forma guiada con una finalidad concreta y definida, para distintos dispositivos (ordenadores, dispositivos móviles y otros) aplicando herramientas de edición y empleando los elementos de programación de manera apropiada.*

Asimismo el currículo de dicha materia establece como uno de los bloques de saberes básicos de esta materia aquellos concernientes a **Programación, pensamiento computacional y robótica,** estableciendo como conocimientos, destrezas y actitudes a desarrollar con el alumnado en 2º de ESO las siguientes:

– Algorítmica y diagramas de flujo.
– Aplicaciones informáticas sencillas para ordenadores: Programación por bloques.
– Autoconfianza e iniciativa: el error, la reevaluación y la depuración de errores como parte del proceso de aprendizaje

Por lo tanto tanto los ejercicios planteados en el curso así como la metodología encajan perfectamente en la programación de esta materia en 2º de ESO, siendo los contenidos de robótica tratados en otros cursos de Aularagón más propios de la misma materia pero en 3º.

A pesar de que en el currículo habla de programación por bloques, nos ha parecido interesante en este curso introducir en paralelo un **programa de pseudocódigo** como PSeInt para acompañar al alumnado en esa transición desde lo intuitivo de una programación por bloques, visual y con la que muchos ya vienen familiarizados desde Educación Primaria, con la sintaxis de los lenguajes de programación, con los que tendrán que empezar a manejarse en cursos posteriores.

Una vez que el alumnado se ha familiarizado con los conceptos básicos de la programación estructurada contenidos en este curso, es tiempo de plantearle **situaciones de aprendizaje** en los que aplicarlos, preferentemente en la resolución de problemas reales y aplicando metodologías de trabajo en equipo. Esto correspondería a la última fase de Ejercicios de Creación. descrita en el apartado de Ejercicios resueltos.

La Competencia arriba descrita también se encuentra en el Currículo de la materia **optativa de 3º de ESO de Programación y Robótica** como Competencia Específica nº 4 de esa materia, pudiendo aplicarse todo lo dicho anteriormente también en el desarrollo de dicha materia.

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

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
GOBIERNO DE ARAGON