Procesamiento

Tipos de operaciones

Las operaciones a realizar con los datos pueden ser de muy diversa naturaleza:

- **Aritméticas**: operaciones clásicas de suma, resta, multiplicación y división.
- **Lógicas**: comparaciones, negación, Y, O.
- **Concatenación:** unión de varios elementos (cadenas de caracteres o variables de diferentes tipos)
- **Bucles**: Implica la realización de acciones de forma repetida. En este caso convendrá distinguir dos tipos:
  - Número de veces a repetir la acción conocido previamente: usaremos las estructuras **Para** (en inglés **For**) o **Repetir**.
  - Número de veces a repetir dependiendo de valores obtenidos: usaremos las estructuras **Mientras** (While...do) o **Repetir hasta que** (Do ...while), según deseemos evaluar la condición antes o después de la primera iteración. Estas estructuras veremos que son condicionales además de repetitivas.
- **Condicionales**: implica la realización de unas acciones u otras tomando decisiones. Estructuras **Si-entonces** (If-else), **Según (**Switch**)**.

Los tres primeros tipos de operaciones ya los hemos ido viendo en los apartados anteriores. Nos centraremos ahora en las dos últimas , si bien introduciremos diversas operaciones en los ejemplos para profundizar en su uso.

Iteraciones y bucles

Para practicar con estas estructuras, realizaremos un pequeño programa que nos calcule el promedio de varios números. En primer lugar el programa solicita la cantidad de números a promediar, luego pide que se introduzcan los números tantas veces como le hayamos dicho (aquí está la repetición) Al final muestra el promedio. En este caso, como el número de iteraciones es conocido, usaremos la instrucción **Para.** Asimismo, utilizaremos **operadores aritméticos** y de **concatenación**.

Pasos 1 y 2: Análisis y diagrama de flujo del programa *Promedio de n números*

Los elementos implicados serán:

- **Inicio y fin** de algoritmo.
- **Salidas**: Solicitar número de calificaciones, mostrar promedio.
- **Entradas**: cantidad de números y números a promediar.
- **Almacenamiento**: número de elementos (número entero), números introducidos (número real), acumulado de la suma de números (número real) y promedio (número real)
- **Procesamiento**: iteraciones, sumas y división.

El diagrama de flujo asociado a la instrucción **PARA** es el siguiente:

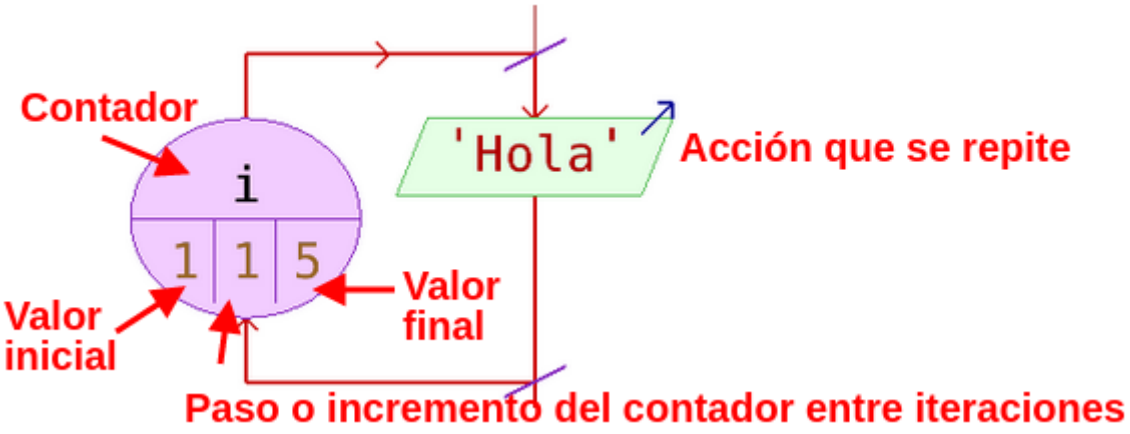
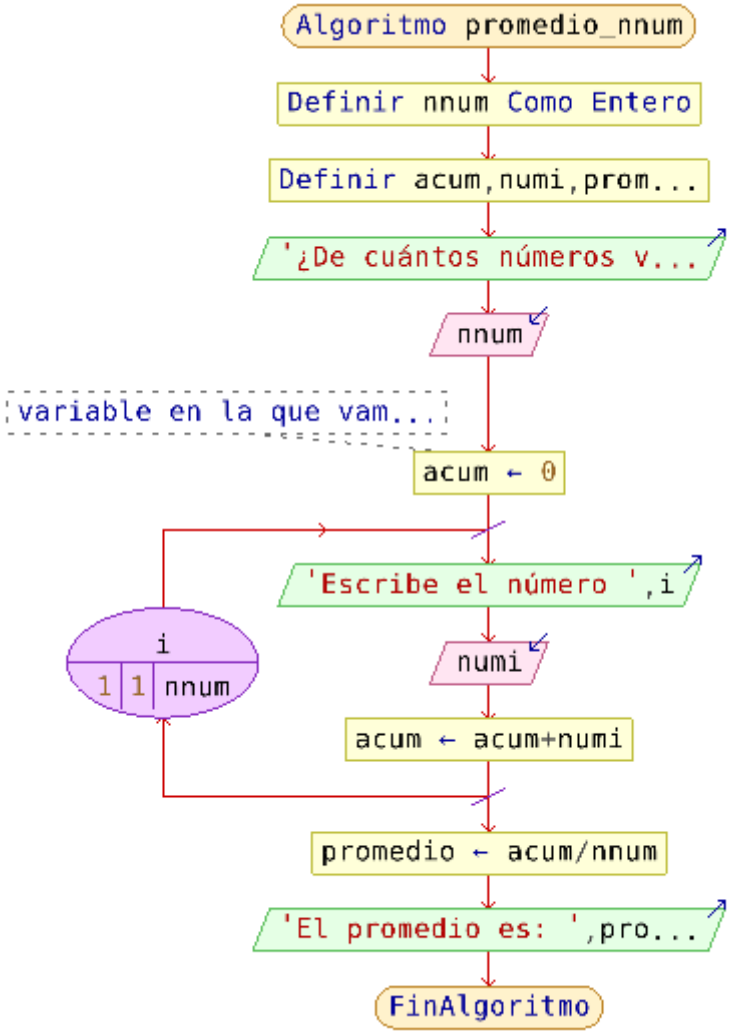


Diagrama de flujo:



Pasos 3, 4 y 5: Codificación, compilación y verificación del programa *Promedio de n números* con PSeInt

Como vimos en el apartado anterior, después de renombrar el algoritmo comenzamos por definir las variables implicadas escribiendo la expresión **Definir**

```
Algoritmo promedio_nnum
    Definir nnum Como Entero;
    Definir acum, numi, promedio Como Real;
```

PSeInt permite definir varias variables del mismo tipo en la misma línea, separadas con comas. Como veis, hemos definido una única variable *numi* donde guardaremos el número introducido por la persona usuaria cada vez.

A continuación solicitamos la cantidad de números a promediar (**Escribir**), leemos la respuesta y la almacenamos en la variable correspondiente (**leer**) e inicializamos el valor de acum en 0, aunque en realidad no sería necesario puesto que es el valor asignado por defecto.

```
Escribir "¿De cuántos números vas a calcular el promedio?";
leer nnum;
acum<-0; //variable en la que vamos a ir sumando todos los números
```

Con el valor de *nnum* definido por el usuario llega el momento de la repetición: solicitaremos al usuario que introduzca un número nnum veces. Para ello usaremos el comando **Para** disponible en la ventana de la derecha.

Al hacer clic sobre el comando, se nos escriben las siguientes instrucciones en nuestra hoja de algoritmo

```
Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
    secuencia_de_acciones
Fin Para
```

Como *variable numérica* habitualmente se define una variable **local** (i, j, k...) que hace de contador. El *valor inicial* es 1 y el *valor final* el número de veces que queremos repetir la instrucción. En *paso* se especifica el crecimiento del contador de una iteración a otra (en nuestro caso de uno en uno)

Por último en la *secuencia de acciones* hemos de poner que es lo que queremos que se repita en cada ocasión. En nuestro caso solicitar el número, leerlo y sumarlo a los anteriores guardando el resultado en la variable *acum*.

```
Para i<-1 Hasta nnum Con Paso 1 Hacer
    Escribir "Escribe el número ", i;
    Leer numi;
    acum<-acum+numi;
Fin Para
```

El valor de *numi* se refresca y varía en cada iteración, puesto que ya hemos guardado el valor del número anterior en el acumulado.

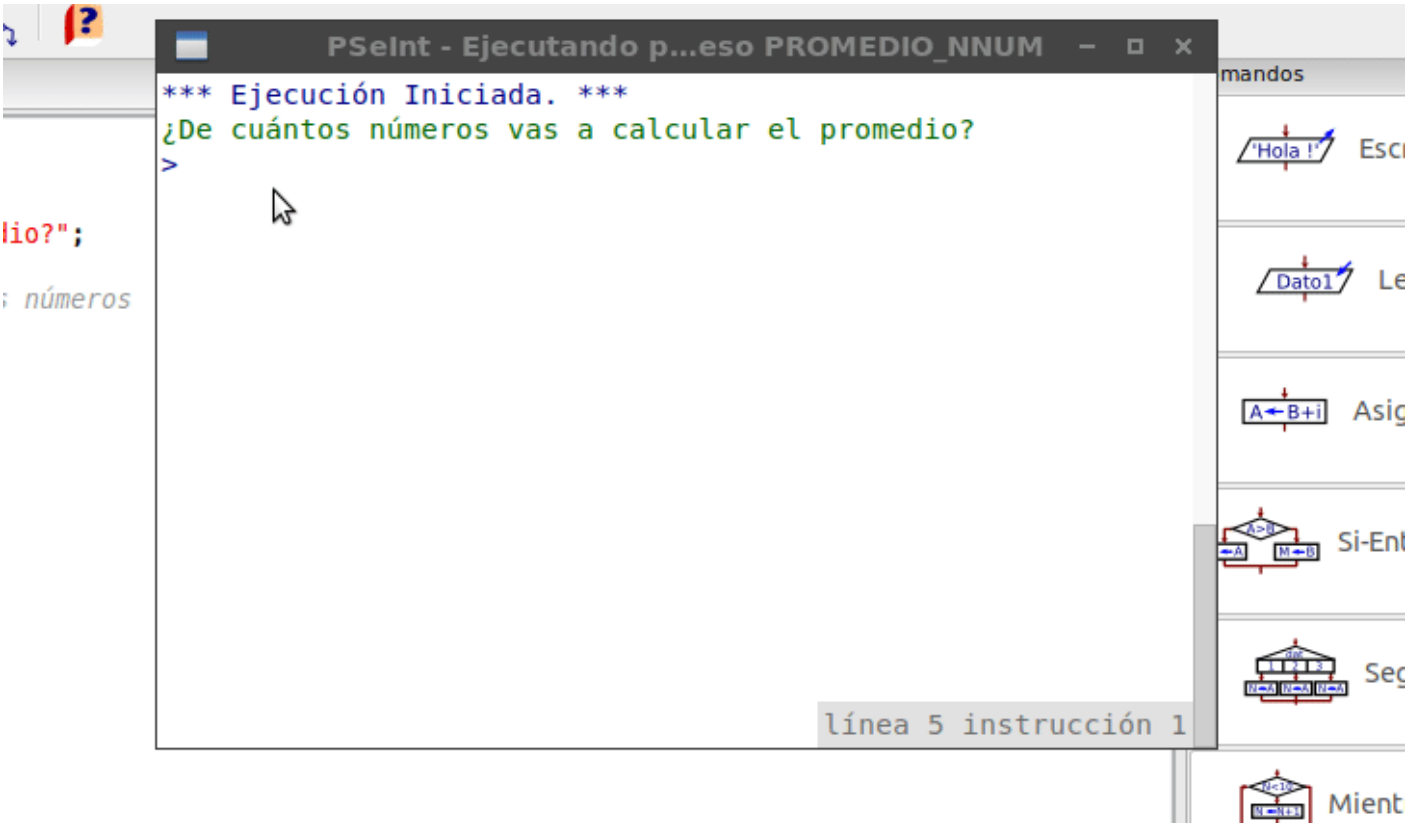
> En un programa es muy importante usar el **mínimo número de variables** precisas para no añadir complejidad innecesaria al mismo. (Principio KISS de diseño de software)

Hemos concatenado en el mensaje de solicitud la variable numérica de contador. Eso nos permite mostrarle al usuario en qué iteración se encuentra. La instrucción **Escribir** seguida de los elementos a concatenar separados por comas los coloca **en la misma línea y sin separación entre ellos**. Por eso es preciso considerar los espacios de separación en las cadenas de caracteres que incluyamos.

Por último, asignamos el valor del promedio al cálculo correspondiente y mostramos el resultado nuevamente concatenando dos mensajes.

```
    promedio<-acum/nnum;
    Escribir "El promedio es: ",promedio;
FinAlgoritmo
```

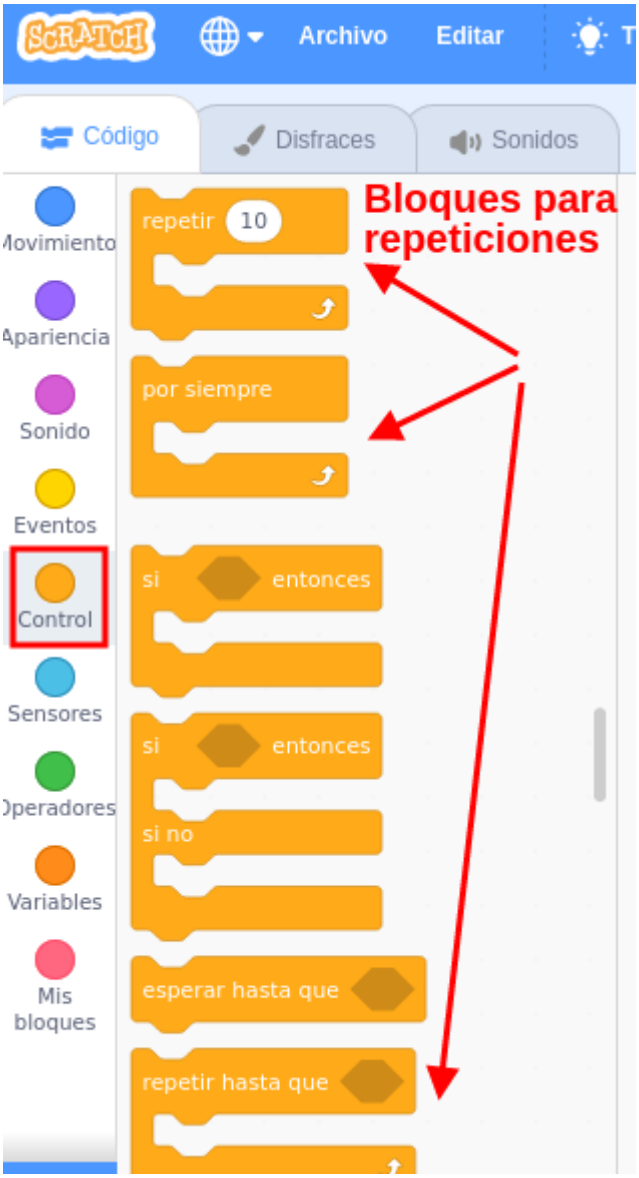
Finalmente le damos a **Ejecutar** y verificamos el proceso realizado.



Nuevamente en la verificación del programa podemos experimentar la robustez del programa frente a fallos, introduciendo números decimales con coma, negativos, etc...Invitamos a realizarlo e introducir las instrucciones necesarias para prevenir dichos fallos.

Pasos 3, 4 y 5: Codificación, compilación y verificación del programa *Promedio de n números* con Scratch

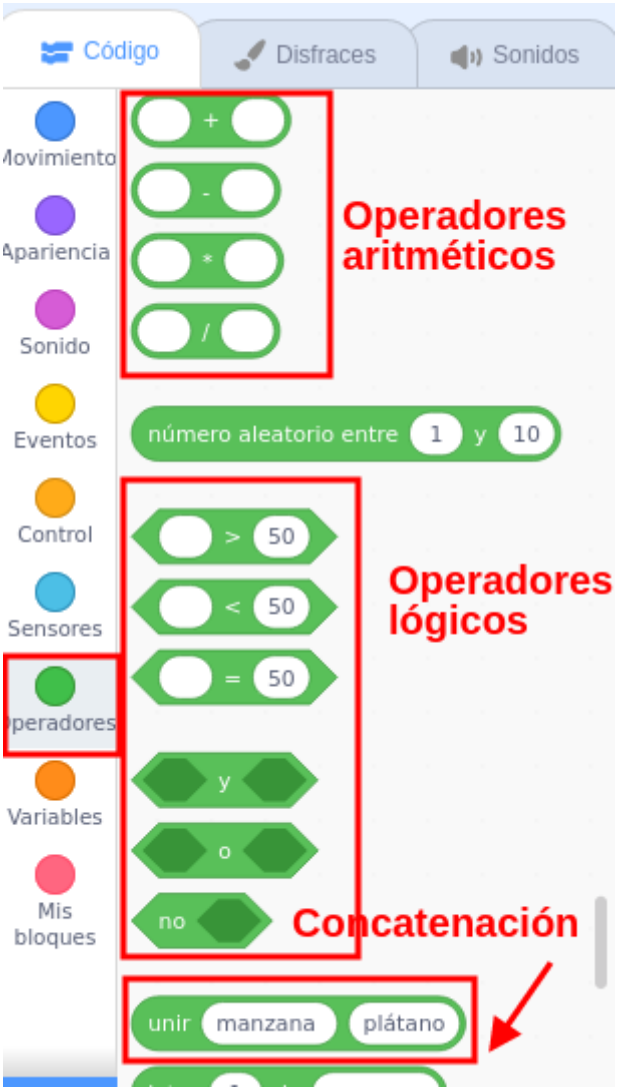
En Scratch los bloques relacionados con las estructuras repetitivas se encuentran en **Control** y son *Por siempre, Repetir y Repetir hasta que.*



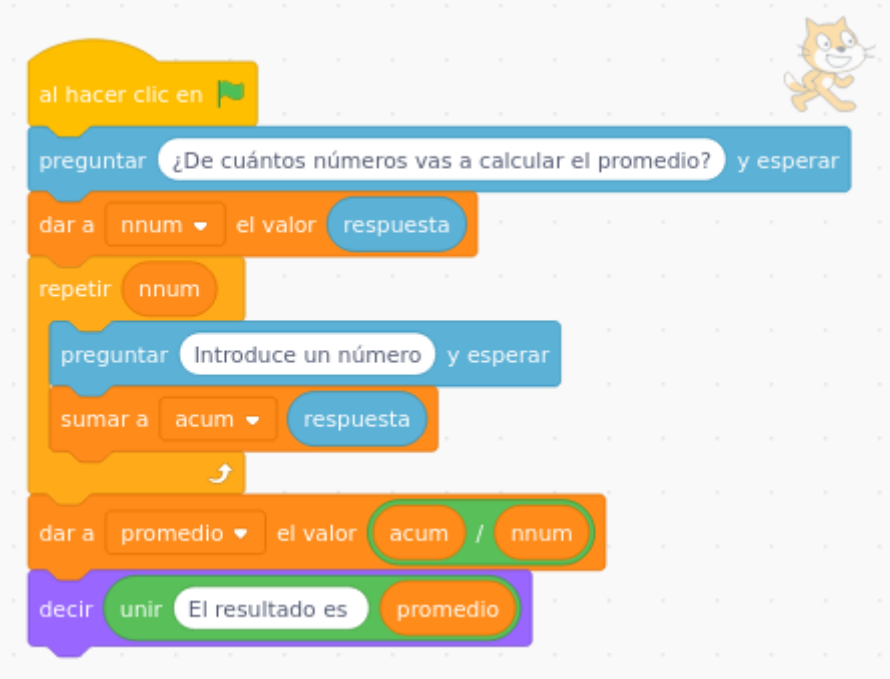
En nuestro caso, como el número estará definido, usaremos **Repetir.**

En primer lugar en el bloque **Variables** definiremos las variables necesarias.

De forma análoga, utilizando los bloques de **Sensores, Apariencia, Variables** explicados en los apartados de Entradas, Salidas y Datos, construimos el programa. En Operadores encontraremos los bloques necesarios para la realización de las operaciones aritmético, lógicas y de concatenación.



El programa finalmente con todos los elementos quedaría así:



Condicionales

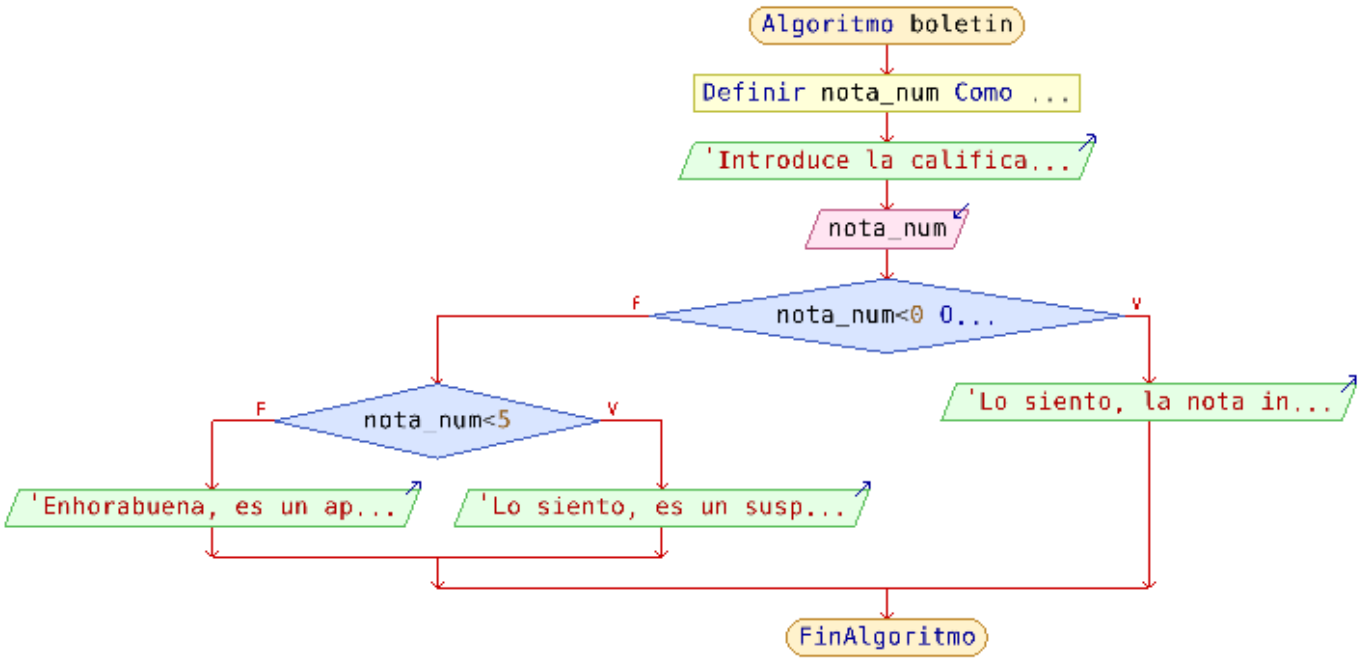
Para practicar con estas estructuras, realizaremos un pequeño programa en el que tras solicitar una nota numérica, y verificar en primer lugar si el valor recibido es correcto, nos indicará si la nota corresponde a un APROBADO o un SUSPENSO.

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

Los elementos implicados serán:

- **Inicio y fin** de algoritmo.
- **Salidas**: Solicitar calificación numérica, mostrar calificación textual o mensaje de error.
- **Entradas**: calificación numérica
- **Almacenamiento**: calificación numérica (número real, puede tener decimales)
- **Procesamiento**: lógicas (comparación,Y,O) y condicionales.

Diagrama de flujo:



En este caso hemos definido la condición de error mediante el operador lógico **O**, indicando que se considere errónea cualquier calificación menor que cero **o** mayor que 10. Este programa puede ser resuelto de forma análoga utilizando el operador **Y**, indicando que considere válida cualquier nota numérica mayor o igual que cero **y** menor o igual que 10.

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

En primer lugar renombramos el algoritmo (*boletin* en nuestro caso) y definimos las variables implicadas que solo es la calificación numérica (*nota_num*)

A continuación solicitamos la calificación numérica (Escribir) y la introducimos en la variable definida (Leer)

```
Algoritmo boletin
    Definir nota_num como Real;
    Escribir "Introduce la calificación numérica. Tiene que ser un número real entre 0 y 10. Si usas decimales recuerda poner un punto";
    Leer nota_num;
```

Ahora vendría la aplicación de la condición. Los comandos encargados en PSeInt de introducir las condiciones son **Si-entonces, Según, Mientras y Repetir**. Un ejemplo de la sintaxis de cada una se puede apreciar en la siguiente figura

```
Algoritmo Condicionales_PSeInt

    Si expresion_logica Entonces
        acciones_por_verdadero
    SiNo
        acciones_por_falso
    Fin Si

    Segun variable_numerica Hacer
        opcion_1:
            secuencia_de_acciones_1
        opcion_2:
            secuencia_de_acciones_2
        opcion_3:
            secuencia_de_acciones_3
        De Otro Modo:
            secuencia_de_acciones_dom
    Fin Segun

    Mientras expresion_logica Hacer
        secuencia_de_acciones
    Fin Mientras

    Repetir
        secuencia_de_acciones
    Hasta Que expresion_logica

FinAlgoritmo
```

- **Si-entonces**: verifica que se cumple una condición o no y en función de ello ejecuta unas acciones. Se pueden anidar unas condicionales dentro de otras.
- **Según**: se utiliza cuando una variable puede adoptar un número discreto de opciones (tipo menú) y se define el comportamiento ante ellas y ante el caso de que no coincida con ninguna.
- **Mientras**: evalúa una condición en bucle y mientras es verdadera ejecuta una acción. Mezcla condicional y bucle.
- **Repetir-hasta que** : ejecuta una acción en bucle hasta que una condición se cumple en cuyo momento se interrumpe. Mezcla condicional y bucle.

En nuestro ejemplo la condición **solo se evalúa una vez** por lo que descartamos las dos últimas, y el valor numérico introducido puede tener **infinitos valores** entre 0 y 10 por lo que la segunda tampoco es válida. Usaremos entonces el comando **Si-entonces.**

Vamos a aprovechar para introducir dos nuevos operadores, el operador **Y** representado por el carácter **&** y el operador **O** representado por la doble barra **||** que se utilizan cuando queremos que se evalúe más de una condición simultáneamente. En nuestro caso podemos considerar dos opciones:

- Si la nota introducida es menor que 0 **o** mayor que 10, no tiene sentido,y hay que dar un mensaje de error. En caso contrario habrá que distinguir (*condicional anidada*) si es menor que 5 y entonces el mensaje dicta suspenso, y si es mayor o igual que 5 que será aprobado.

```
Si nota_num<0 || nota_num>10 Entonces
    Escribir "Lo siento, la nota introducida no es válida";
SiNo
    Si nota_num<5  Entonces
        Escribir "Lo siento, es un suspenso.";
    SiNo
        Escribir "Enhorabuena, es un aprobado.";
    Fin Si
Fin Si
```

- Si la nota introducida es mayor o igual que 0 **y** menor o igual que 10, es correcta y hay que distinguir (*condicional anidada*) si es menor que 5 o mayor. En caso contrario sacar el mensaje de error.

```
Si nota_num>=0 & nota_num<=10 Entonces

    Si nota_num<5  Entonces
        Escribir "Lo siento, es un suspenso.";
    SiNo
        Escribir "Enhorabuena, es un aprobado.";
    Fin Si

SiNo
    Escribir "Lo siento, la nota introducida no es válida";
Fin Si
```

Las dos son correctas y elegir una u otra solo dependerá de la persona programadora y si prefiere usar operadores **Y** u **O**.

```
Algoritmo boletinOR
    Definir nota_num como Real;
    Escribir "Introduce la calificación numérica. Tiene que ser un número real entre 0 y 10. Si usas decimales recuerda poner un punto";
    Leer nota_num;


    Si nota_num<0 || nota_num>10 Entonces
        Escribir "Lo siento, la nota introducida no es válida";
    SiNo
        Si nota_num<5  Entonces
            Escribir "Lo siento, es un suspenso.";
        SiNo
            Escribir "Enhorabuena, es un aprobado.";
        Fin Si
    Fin Si
```

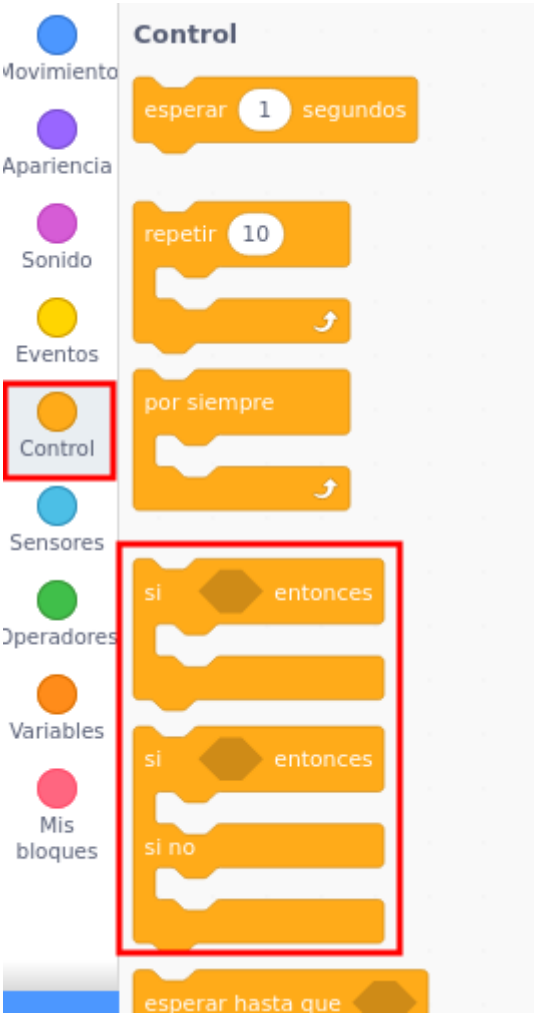
```
Algoritmo boletinAND
        Definir nota_num como Real;
        Escribir "Introduce la calificación numérica.Tiene que ser un número real entre 0 y 10. Si usas decimales recuerda poner un punto"
        Leer nota_num;


        Si nota_num>=0 & nota_num<=10 Entonces

            Si nota_num<5  Entonces
                Escribir "Lo siento, es un suspenso.";
            SiNo
                Escribir "Enhorabuena, es un aprobado.";
            Fin Si

        SiNo
            Escribir "Lo siento, la nota introducida no es válida";
        Fin Si

FinAlgoritmo
```

Finalmente sólo nos quedaría **Ejecutar** el programa y verificar su correcto funcionamiento.

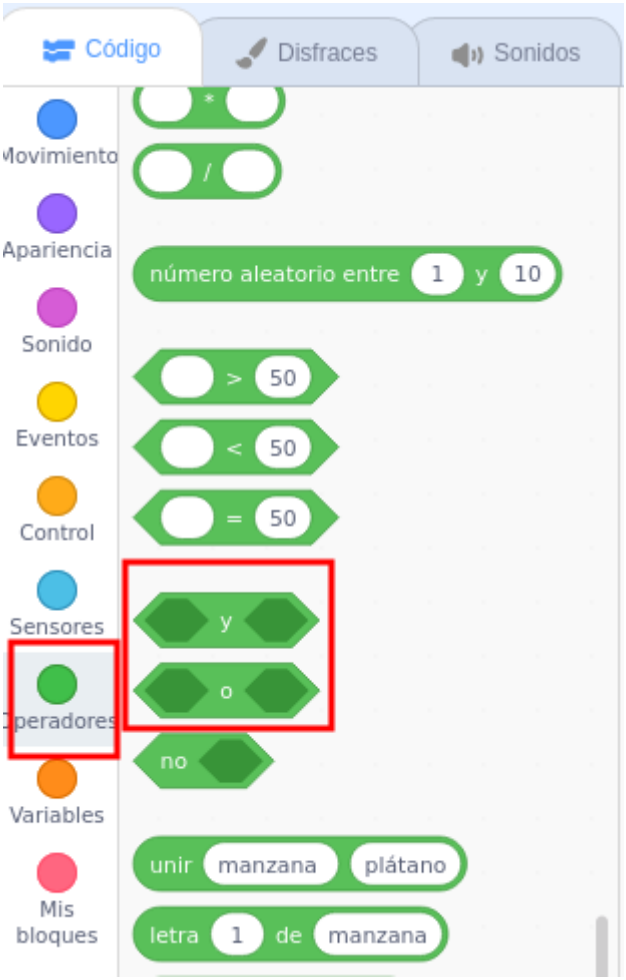
Dejamos como ejercicio de ampliación el reformular este programa para que siga preguntando hasta que reciba una calificación válida. Como pista sugerimos explorar las posibilidades de la instrucción **Repetir**.

Pasos 3, 4 y 5: Codificación, compilación y verificación del programa *Boletín* con Scratch

En Scratch los bloques relacionados con las estructuras condicionales se encuentran en **Control,** junto a los bucles.



Los operadores lógicos Y y O se encuentran en **Operadores**, junto con los de comparación y los ya vistos anteriormente: aritméticos, de concatenación...

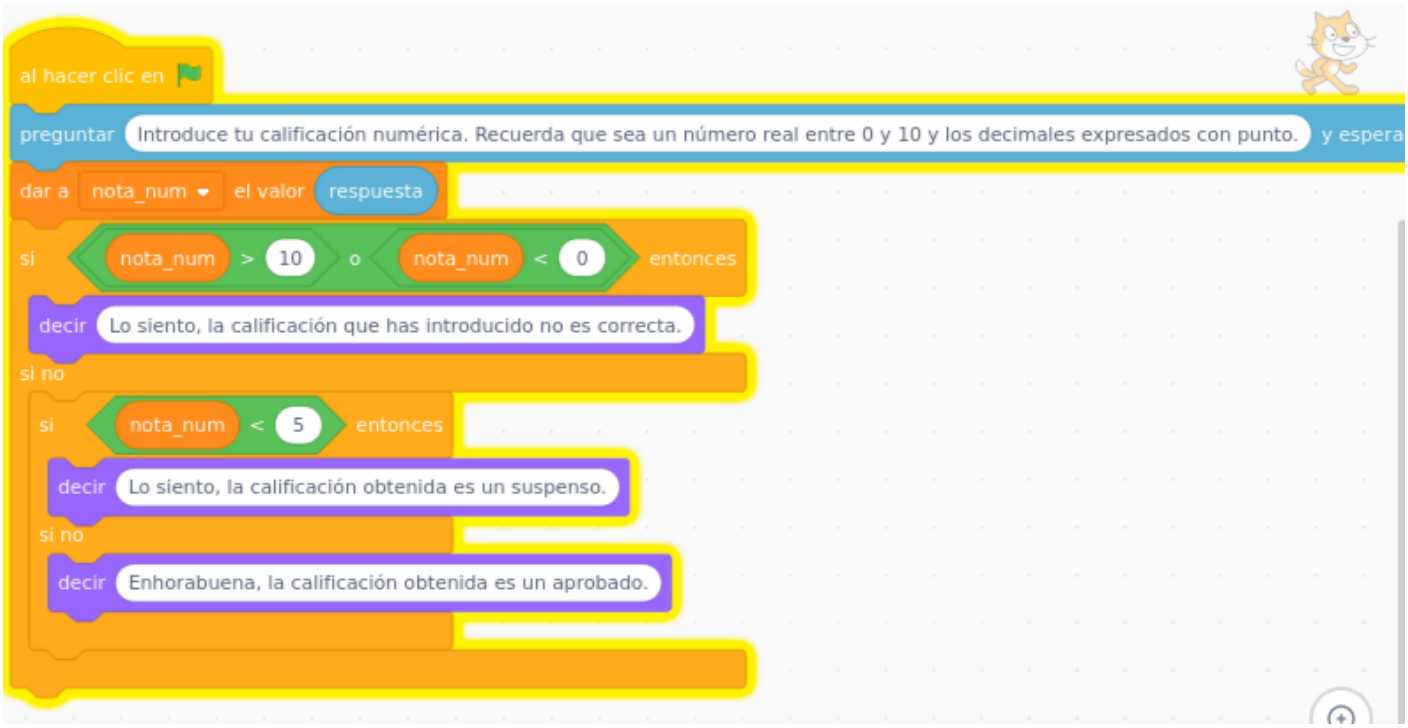


Como en todos los programas empezaremos definiendo las variables necesarias, en nuestro caso sólo una desde los bloques de **Variables**.

A continuación solicitaremos la calificación mediante el bloque disponible en **Sensores** y almacenaremos su respuesta en la variable recién creada.

Por último evaluaremos la respuesta obtenida mediante el bloque correspondiente de **Control** y utilizando los operadores lógicos de **Operadores**, según sea su valor ofreceremos el mensaje de error o escribiremos la calificación final con el bloque correspondiente de **Apariencia.**

El código resultante sería el siguiente:



En este caso hemos utilizado el operador **O** porque Scratch no dispone de los operadores combinados menor o igual y mayor o igual, y así nos ahorramos el tener que combinar ambos operadores con un operador lógico O. Recordad el principio KISS.

Por último, verificamos el correcto funcionamiento del programa y su robustez frente a los errores más comunes. En Scratch **ejecutamos** haciendo clic sobre la bandera verde.

Pruébalo aquí !!

https://scratch.mit.edu/projects/749701636/embed

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

Revision #36
Created 5 June 2022 14:46:56 by Ana López Floría
Updated 17 January 2023 15:48:32 by Equipo CATEDU