Sesión 3

Aprendizajes esperados

Al final de esta sesión verifica que puedas:

- Usar contadores en la programación de temporizadores.
- Usar acumuladores que permitan, a partir de conteos, filtrar datos para la ejecución de procesos en un programa.
- Automatizar máquinas de estados a partir de contadores y acumuladores dentro de condicionales y bucles anidados.

Duración sugerida

15% 70% 15%

Material para la clase

- ○ *Tinkercad* - cuenta de estudiante o personal.

Lo que sabemos, lo que debemos saber

Esta sección corresponde al 15% de avance de la sesión

En la ciudad del futuro también habrá eventos emocionantes como carreras de velocidad. Usando el mismo semáforo que programamos anteriormente, aprenderemos a adaptarlo para que funcione como un semáforo de carreras, contando los segundos hasta dar la señal de salida. Con la ayuda de contadores y acumuladores, veremos cómo modificar y personalizar los sistemas para adaptarse a diferentes necesidades, demostrando así la versatilidad y adaptabilidad de la tecnología en nuestra ciudad ideal.

Haremos una variación del semáforo creado en la sesión pasada, para lo cual se recomienda realizar un nuevo montaje en *Tinkercad*.

> *¿Has notado que en las carreras de coches hay semáforos que indican a los corredores cuándo es el momento de partir?*

Estos semáforos no funcionan con los cuatro estados que encontramos en las calles de la ciudad; en cambio, operan con tres estados: comienzan con la luz roja encendida, luego se enciende la luz amarilla, y finalmente, en un tercer estado, se enciende la luz verde, señalando el inicio de la carrera. Vamos a programar este tipo de semáforo usando condicionales dentro de bucles y un par de variables (contador y acumulador) que nos ayudarán a llevar un conteo en el arreglo de LED de la *micro:bit*.

Los **contadores** y **acumuladores** son conceptos de programación utilizados para realizar un seguimiento de operaciones o condiciones dentro de un programa.

Los **contadores** son variables que se usan para contar el número de veces que ocurre un evento específico; se incrementan o aumentan (o, en algunos casos, se decrementan o disminuyen) cada vez que se produce dicho evento. Los **acumuladores**, por otro lado, son variables que se emplean para sumar (o acumular) valores durante la ejecución de un programa.

A diferencia de los contadores, que solo cuentan, los acumuladores mantienen un total al que se agregan valores continuamente.

Piensa en el caso de una caja registradora en un supermercado: cada vez que un artículo pasa por el escáner, se incrementa un contador que lleva el número total de artículos comprados. Al mismo tiempo, el precio de cada artículo se suma a un acumulador que mantiene el total a pagar.

Glosario

- **Contador:** variable que va incrementando a medida que sucede un evento.
- **Acumulador:** variable que almacena y suma valores sucesivos a medida que el programa avanza.
- **Display o matriz de LED:** conjunto de 25 LEDs en forma de matriz 5x5 en la *micro:bit*, utilizado para mostrar imágenes, texto o gráficos simples.
- **Bucles anidados:** estructuras de programación donde un bucle se encuentra dentro de otro, permitiendo iteraciones múltiples en diferentes niveles.

Figura 1. Montaje de semaforo

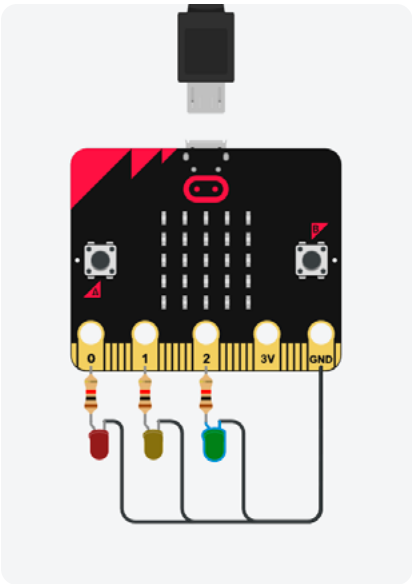


Figura 2. Programación de variables en bloque "On Start"

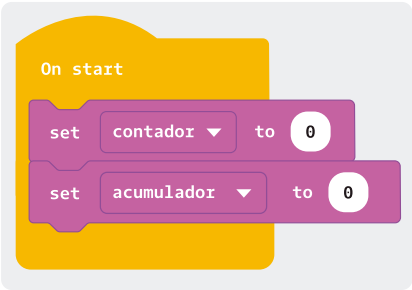
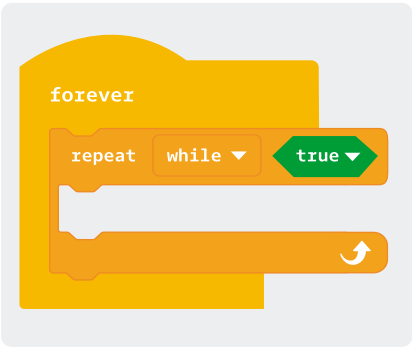


Figura 3. Bloque de repetición en bloque "Forever"



Manos a la obra

Conectadas

Esta sección corresponde al 85% de avance de la sesión

Sigue los pasos teniendo en cuenta que vamos a usar un montaje electrónico idéntico al del primer semáforo, pero esta vez nos enfocaremos en automatizar los estados del semáforo de carreras usando un contador y un acumulador dentro de bucles y condicionales anidadas.

Paso 1: haz exactamente el mismo montaje del semáforo que creaste en la práctica anterior, ver *Figura 1.*

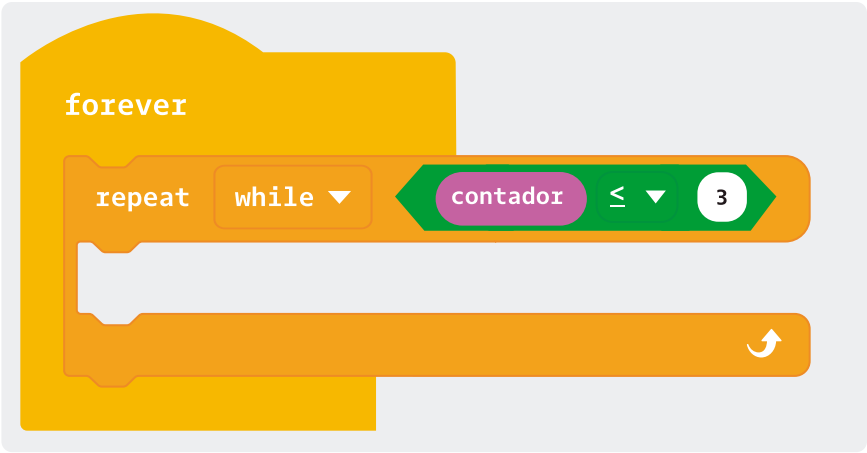
Paso 2: en la zona de código busca la paleta "Variables" y crea las variables "acumulador" y "contador". Conéctalas a la función "On start" o "Al empezar" y déjalas en cero como puede ver en la *Figura 2.*

Con la variable contador llevaremos un conteo de 0 a 3 en el **display o matriz de LED** de la *micro:bit* que indicará el cambio de estado del semáforo, mientras que con el **acumulador** vamos a ir sumando los conteos, por esta razón ambas variables deben iniciar en cero.

Paso 3: en la paleta control busca la opción "repeat while (true)" o "repetir mientras (verdadero)", anida este bucle con el bucle "forever" o "para siempre", ver *Figura 3.*

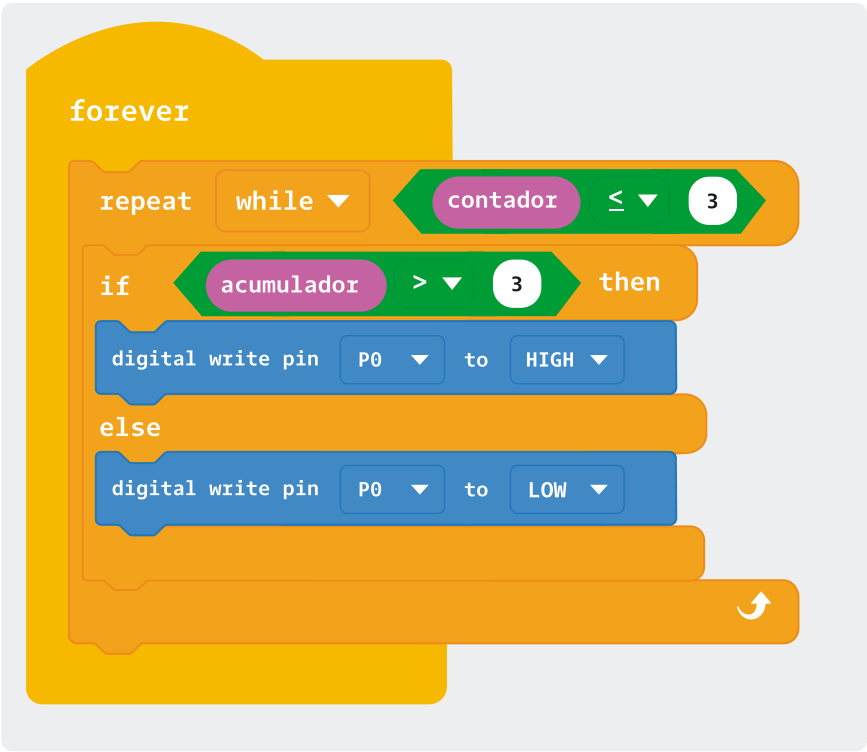
Paso 4: ahora busca en las paletas "Math" y "Variables" y completa la sentencia condicional "repetir mientras que contador sea menor o igual que 3", conéctala en "true" o verdadero, ver *Figura 4.*

Figura 4. Programación condicional con operadores



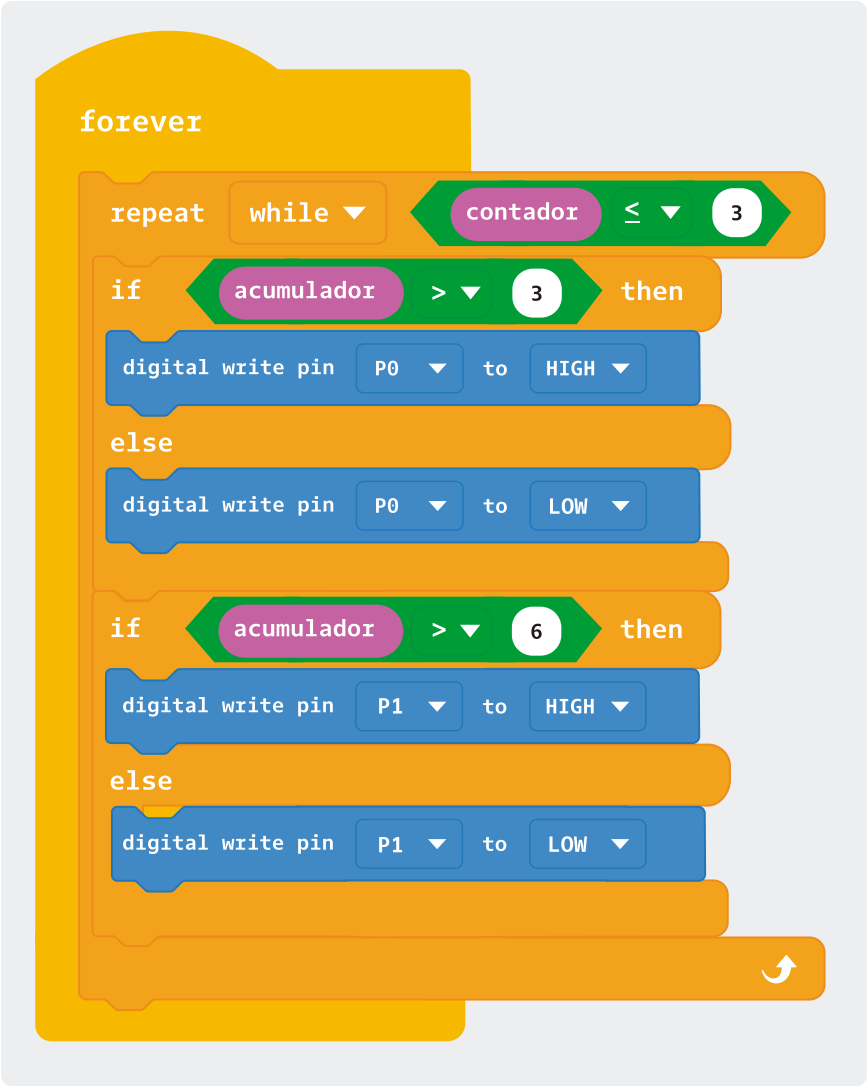
Paso 5: anida una condicional doble al bucle "while" y como sentencia condicional pregunta si el acumulador es mayor que 3. En caso de ser cierto pin 0 donde está conectado el LED rojo se encenderá, en caso contrario permanecerá apagado o en "LOW", ver *Figura 5.*

Figura 5. Programación completa de condicional



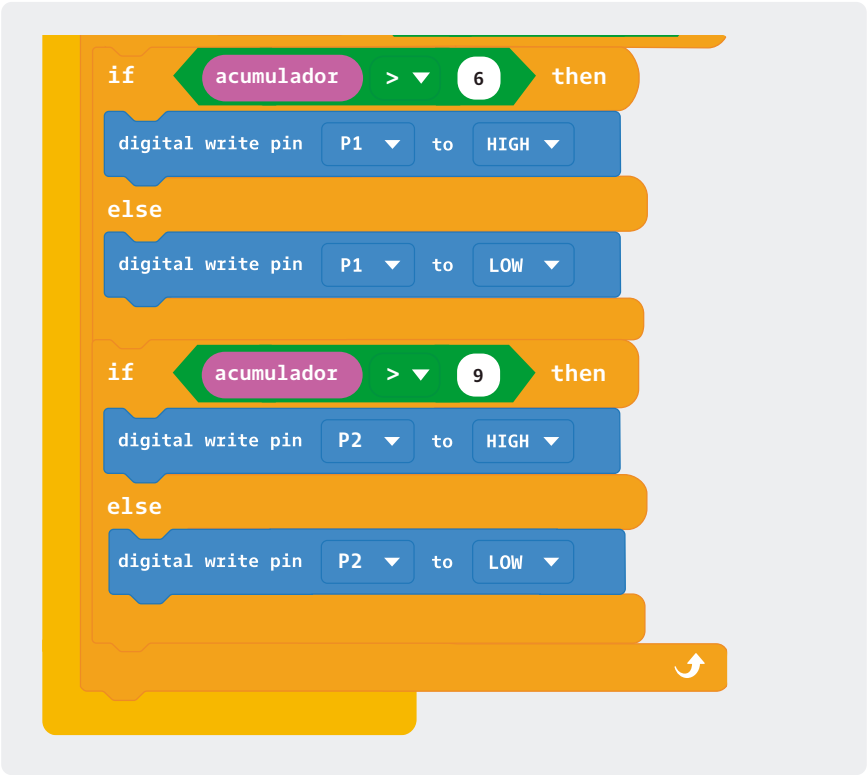
Paso 6: crea otra condicional donde se encienda el LED amarillo conectado a pin 1 cuando el acumulador sea mayor a 6. Conéctala dentro del ciclo "while" o mientras, debajo de la primera condicional , como se observa en la *Figura 6*.

Figura 6. Programación de otro condicional



Paso 7: vuelve a crear otra condicional donde preguntes si el acumulador es mayor a 9, en caso de ser cierto activará en HIGH a pin 2 donde se encuentra conectado el LED verde. Conéctala debajo de las otras dos condicionales.

Figura 7. Programación de otro condicional



Paso 8: crea una nueva condicional doble donde preguntes si el acumulador es menor que 9. En caso de que sea cierto, mostrará el valor de la variable contador de 0 a 3 en el display o matriz de LED de la *micro:bit* y sumará un 1 al contador. En caso contrario, se mostrará un ícono de chulito en el display o matriz de LED de la *micro:bit.*

Figura 8. Creación de condicional doble

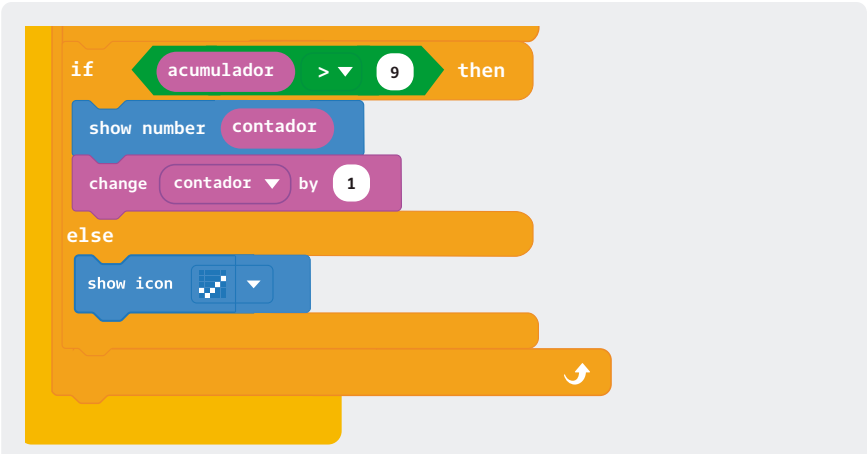
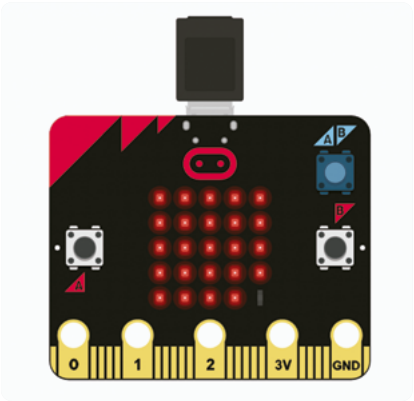
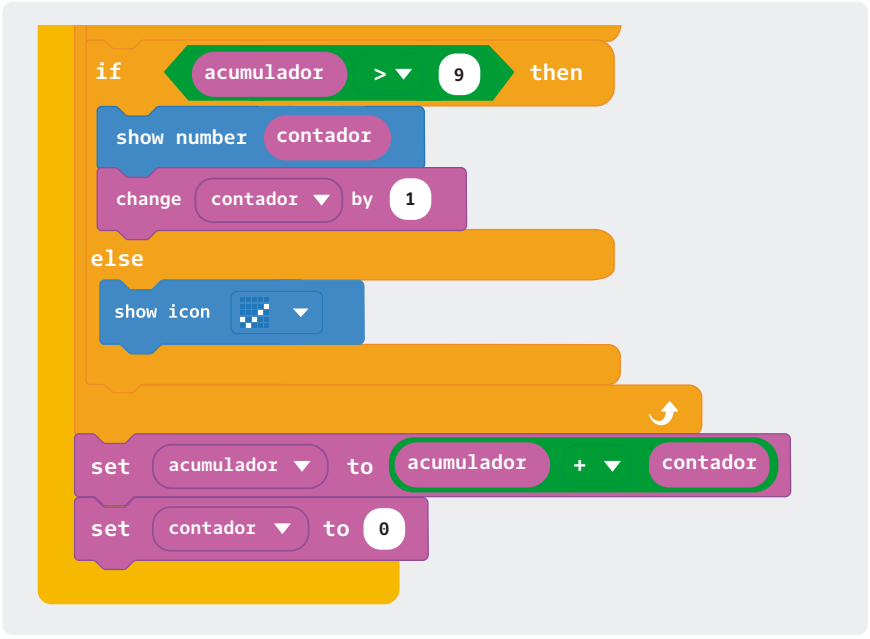


Figura 10. Filas y Columnas en el display o matriz de LED de la *micro:bit*



Paso 9: fuera del ciclo "while" o "mientras que" asigna al acumulador la suma de sí mismo más el contador y posteriormente reinicia el contador a cero.

Figura 9. Suma de variable y reinicio de contador a a cero



Simula el semáforo y notarás que el contador no supera el número 3 iniciando con el cero, debido a que fue declarado con cero desde la función "on start" o "al iniciar" y posteriormente cada vez que alcanza el valor de 3 se sale del ciclo "while" y vuelve a retomar el valor de cero.

¿?

¿Qué relación encuentras entre el contador y el acumulador?

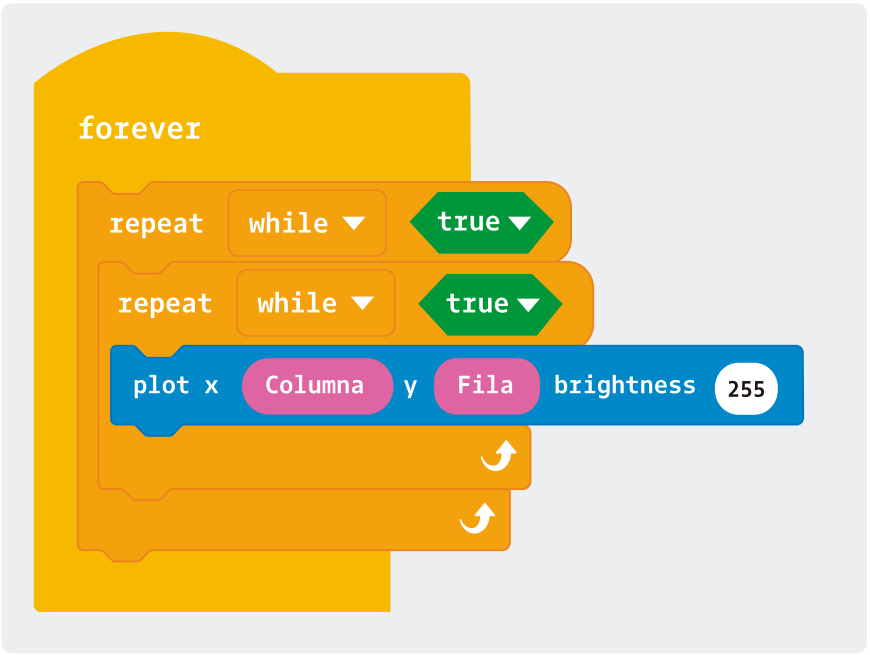
¿Qué ocurriría con el programa si el contador o el acumulador se inician con un número diferente a cero?

Para ir más lejos

Los **bucles anidados** son usados en nuestra cotidianidad pues se trata de repeticiones que dependen de otras. Un ejemplo de estos bucles anidados es nuestra percepción y medición del tiempo. Piensa en un reloj. Podemos decir que la unidad básica de tiempo son los segundos, que al irse contando desde cero y llegar hasta 59 hacen el incremento de los minutos en un minuto, tras lo cual los segundos se reinician. Igualmente ocurre cuando se completan 60 minutos, pues se reinician los minutos a 0 y se incrementan en 1 las horas. También podríamos seguir anidando los ciclos del tiempo y decir que al completarse 24 horas se incrementan en 1 los días, al llegar a 30 o 31 días se incrementan los meses y al completar 12 meses se incrementan los años.

Tu reto consiste en utilizar los bucles anidados y los contadores para rellenar gradualmente el arreglo de LED de la *micro:bit*. Para solucionarlo, crea dos variables "Fila" y "Columna" para ir llenando el display o matriz de LED de la *micro:bit Figura 10*, usa la *Figura 11* como pista para completar el código.

Figura 11. Bucles anidados y contadores



Antes de irnos

Esta sección corresponde al 100% de avance de la sesión

Revisa los aprendizajes de la sesión de forma individual respondiendo las preguntas de forma que mejor reflejen tu progreso:

1. ¿Puedes aplicar contadores para el registro de veces que se ejecuta un evento en un programa?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

2. ¿Puedes acumuladores para totalizar la cantidad de conteos y discriminar con condicionales que nuevos eventos ejecutar de acuerdo con el total acumulado?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

3. ¿Puedes usar condicionales, bucles anidados, contadores y acumuladores para temporizar acciones que se ejecutan desde la *micro:bit*?

   - ◯ Sí
   - ◯ Parcialmente
   - ◯ Aún no

Si tus respuestas fueron "Parcialmente" o "Aún no", discute con tu compañera o compañero de grupo el trabajo realizado relacionado con los bucles anidados, verifica la solución que dieron a uno de los ejercicios planteados y consulta con tu docente las preguntas que aún tengas.

Ahora te proponemos la siguiente pregunta

> *Si regresamos al reto, ¿para qué serviría lo que hemos aprendido en esta sesión?*

Imaginemos un caso de la vida cotidiana.

Imagina que estás diseñando un semáforo para una ciudad. Considerando lo aprendido sobre contadores, acumuladores y bucles anidados, ¿cómo lo utilizarías para optimizar el tiempo de cambio de luces en función del flujo de tráfico?

Discute brevemente cómo adaptarías el código para ajustar automáticamente los tiempos de cada luz en función de la cantidad de vehículos detectados.

Aprovecha este espacio final para hacerte un esquema en que resumas algo de lo que aprendiste, por ejemplo, colocando las definiciones de palabras y algún ejemplo de uso de los tipos de variables **contador** y **acumulador** en **bucles anidados**.