UTN FRGP
TUP

Técnico Universitario en Programación

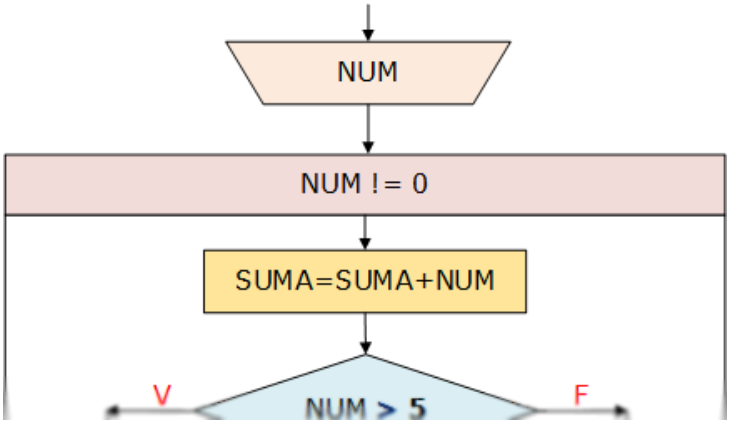
UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Gral. Pacheco

Apuntes de clase de la asignatura

Programación I

CICLO INEXACTO



2024

Abel Oscar Faure

Lorena Raquel Palermo

CICLO INEXACTO

Un ciclo inexacto es aquel que continúa ejecutándose mientras se cumpla una determinada condición lógica.

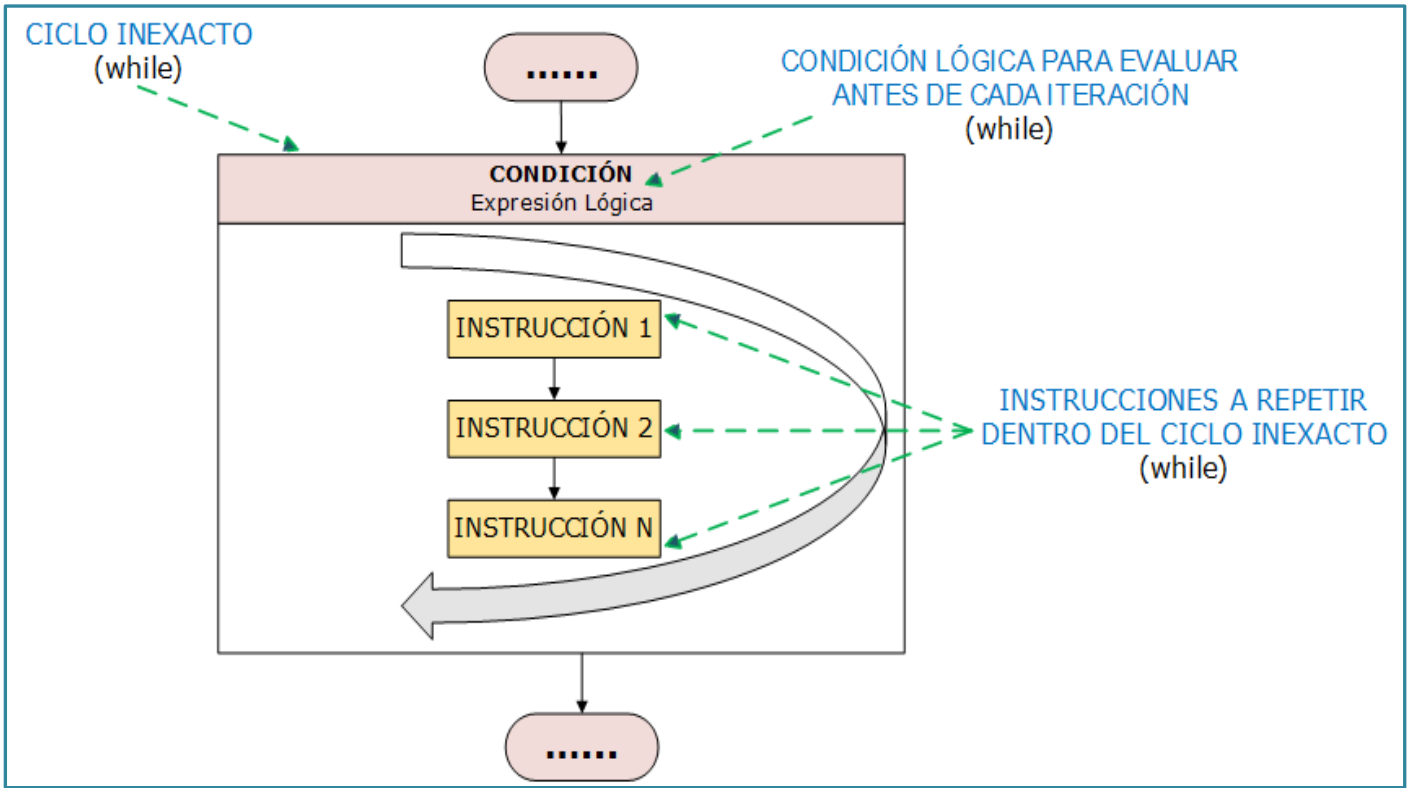
Esta condición puede depender de factores que cambian durante la ejecución del ciclo, lo que hace que la cantidad de repeticiones no sea predecible de antemano.

Tipos de Ciclos Inexactos:

- **Ciclo Precondicional (while).**
- **Ciclo Poscondicional (do-while).**

CICLO PRECONDICIONAL (while)

Este tipo de ciclo evalúa una expresión lógica antes de cada iteración. Si es verdadera, el ciclo se ejecuta; si es falsa, se detiene. En términos lógicos, puede verse como una operación que se repite mientras la condición (Expresión Lógica) sea verdadera. Al evaluarla antes de cada vuelta, se asegura que todo esté en orden antes de ejecutar cualquier acción.



Funcionamiento básico de un ciclo inexacto (while)

1- Antes de ejecutar el bloque de instrucciones, se evalúa una condición lógica que puede ser verdadera o falsa.

2- Si la condición es verdadera, se ejecuta el conjunto de instrucciones dentro del ciclo.

3- Tras cada iteración, se vuelve a evaluar la condición. Si sigue siendo verdadera, el ciclo se repite; si es falsa, el ciclo termina.

Importante:

Para que el ciclo termine en algún momento, debe haber una acción dentro de él que altere la condición evaluada, de modo que eventualmente esta se vuelva falsa y el ciclo se detenga, evitando que se convierta en un bucle infinito.

Ejemplo:

Se requiere implementar un algoritmo que permita ingresar una lista de números enteros, finalizando cuando se ingrese un 0. Luego, el algoritmo debe informar la suma total de los números ingresados. Representar este algoritmo mediante un diagrama de flujo.

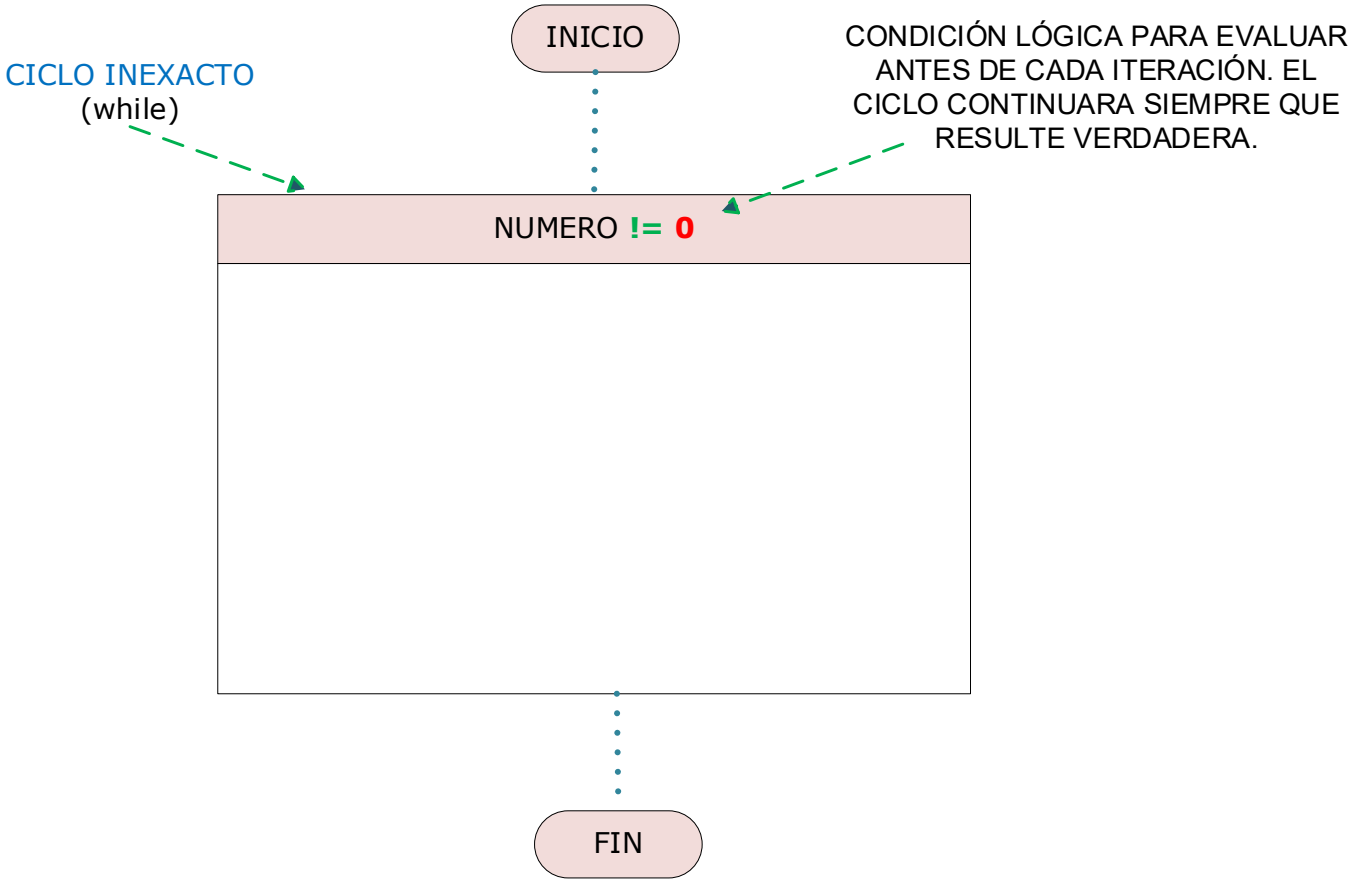
El problema consiste en ingresar una serie de números enteros y calcular la suma de todos los valores ingresados. El proceso de ingreso de datos se detiene cuando se ingresa un 0, que debe ser excluido de la suma, aunque no afecte al resultado final.

Podemos empezar analizando como podrían ser esta lista de números enteros, veamos algunos ejemplos:

Lista 1	Lista 2	Lista 3	Lista 4
5	2	6	11
-9	-47	77	7
15	-5	**0**	16
25	35		23
0	40		-7
	125		-2
	8		18
	10		25
	0		13
			99
			20
			0

Como se puede observar en los ejemplos, la lista 1 tiene 4 números, la lista 2 tiene 8 números, la lista 3 tiene solo 2 números y, finalmente, la lista 4 cuenta con 11 números. Dado que el ejercicio no especifica cuántos números tiene cada lista, no sabemos cuántas veces debemos repetir el ciclo para solicitar el ingreso de los números. Por lo tanto, es necesario utilizar un ciclo inexacto como el while.

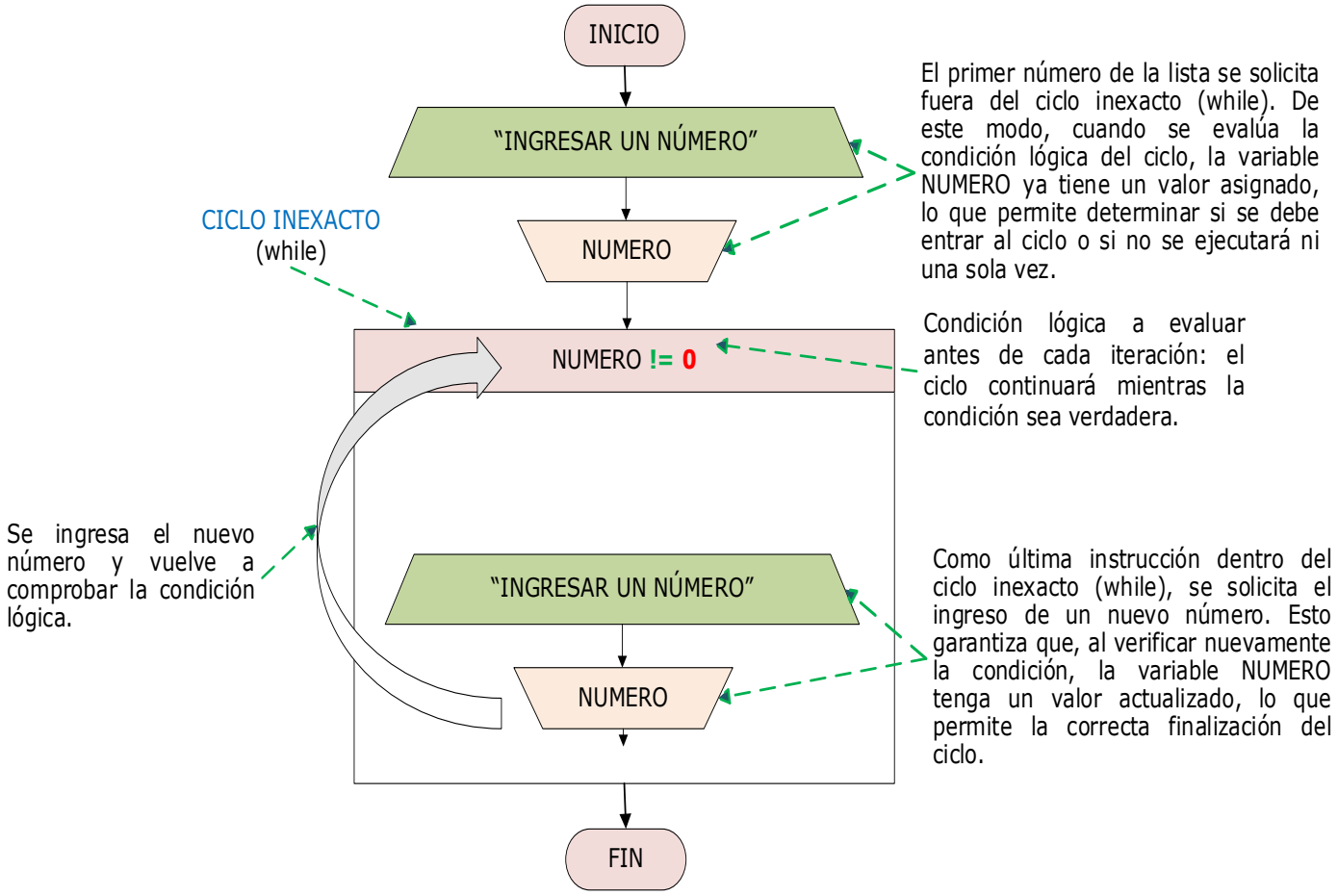
Para determinar cuándo debe detenerse el ciclo, el enunciado nos indica que el ingreso finaliza al ingresar un 0. Esto significa que el ciclo continuará siempre que el número ingresado sea distinto de 0. Podemos comenzar representando el ciclo y su condición.



Para continuar con el paso a paso, vamos a representar cómo se ingresan los números de la lista para poder procesarlos y cómo se produce el corte del ciclo inexacto (while).

Es importante mencionar que las variables que forman parte de la condición en un ciclo se conocen como centinelas, ya que su función es similar a la de un guardia que "vigila" cuándo debe detenerse el ciclo. El centinela controla la condición que permite al ciclo continuar o finalizar. Es el valor clave que se verifica en cada iteración para decidir si el ciclo debe seguir ejecutándose o detenerse.

Por lo general, la variable centinela está involucrada en la expresión lógica del ciclo. En este problema, el ciclo se repite mientras la variable NUMERO sea distinta de 0, esa variable NUMERO actúa como centinela. Cuando su valor es 0, el ciclo se detiene automáticamente.



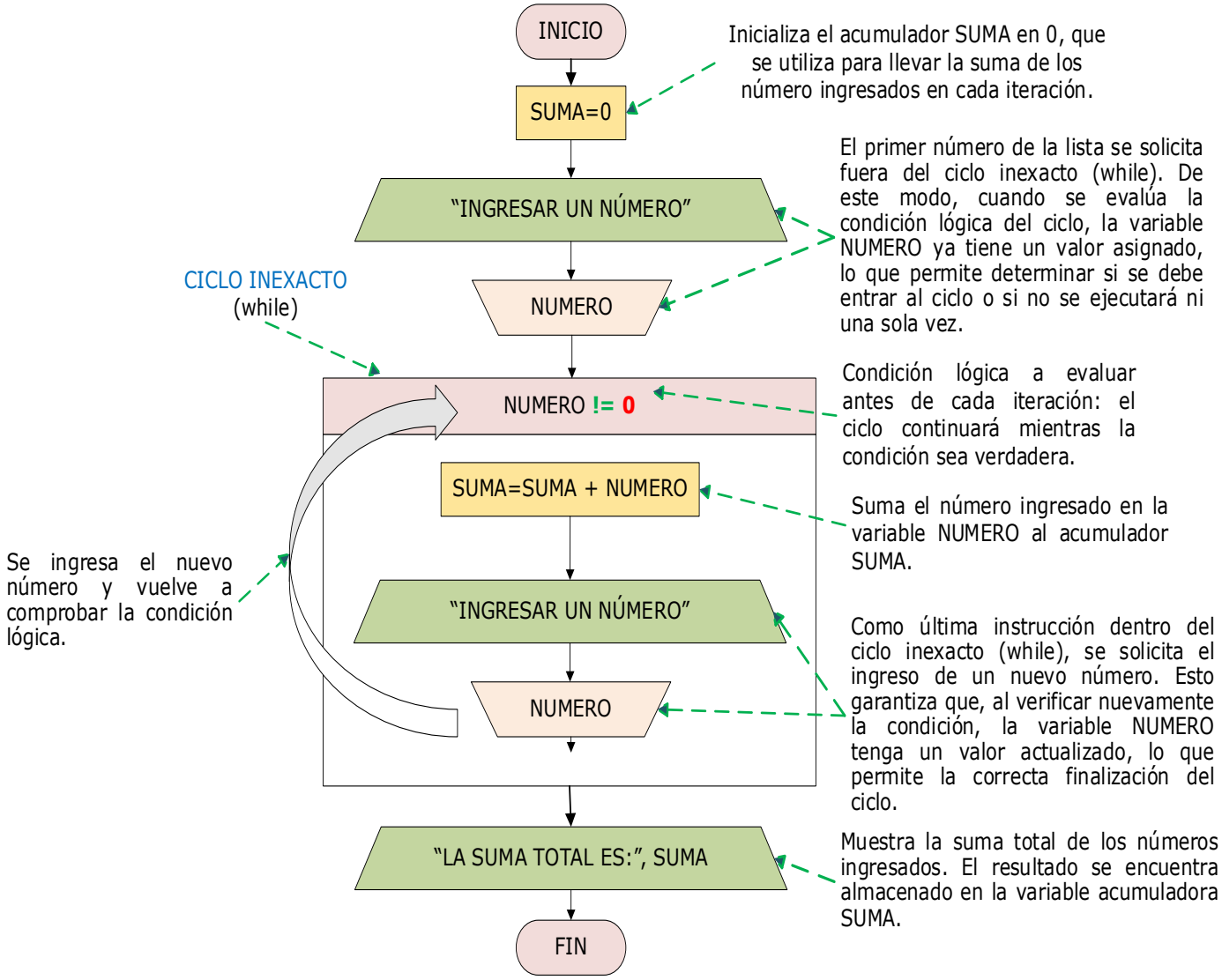
El último paso que vamos a representar es la suma total de todos los números ingresados. Para esto, necesitamos una variable que funcione como acumulador, similar a como usamos las variables contadoras. Esta variable, que llamaremos SUMA, debe inicializarse en 0 antes de comenzar a utilizarla.

Dentro del ciclo inexacto (while), iremos sumando en la variable SUMA todos los números que se vayan ingresando. Una vez que el ciclo termine, se mostrará el resultado de la suma total a través de un mensaje.

Tanto la inicialización de la variable SUMA como el mensaje con el resultado final deben estar fuera del ciclo. La inicialización de SUMA se realiza una sola vez, antes de empezar a acumular los números. Si se colocara dentro del ciclo, la variable se reiniciaría a 0 en cada iteración, perdiendo el valor acumulado.

En cuanto al mensaje con el resultado de la suma, este también debe estar fuera del ciclo, ya que, si lo colocáramos dentro, se mostraría una suma parcial en cada

iteración. Queremos que el mensaje aparezca una sola vez, al final del proceso, mostrando el resultado total.

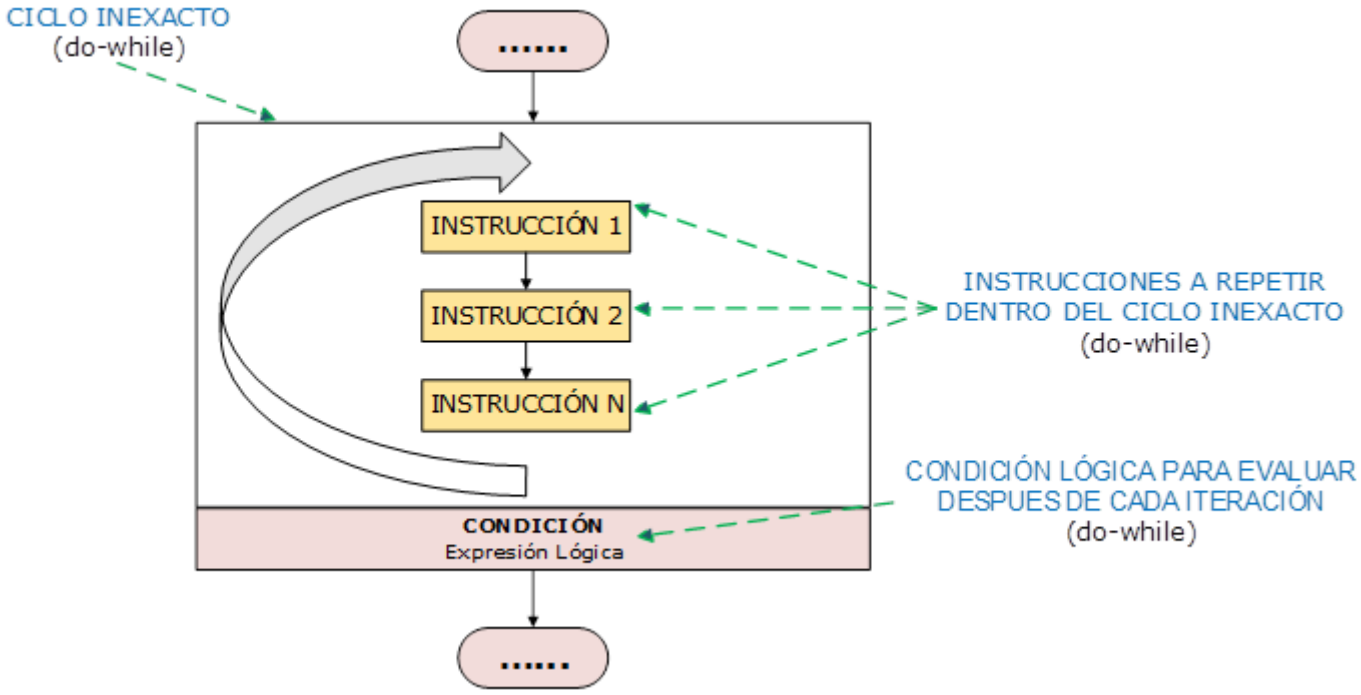


CICLO PRECONDICIONAL (do-while)

Un ciclo inexacto postcondicional (do-while) es un tipo de estructura repetitiva que se ejecuta al menos una vez antes de evaluar la condición de finalización.

A diferencia de los ciclos precondicionales (como el while), en los cuales se verifica la condición antes de la primera ejecución, en un ciclo postcondicional la condición se evalúa después de que el bloque de instrucciones se haya ejecutado.

Esto asegura que el ciclo siempre se ejecute por lo menos una vez, independientemente de la condición.



Funcionamiento básico de un ciclo inexacto (do-while)

1- El bloque de instrucciones dentro del Ciclo Inexacto (do-while) se ejecuta sin comprobar la condición lógica.

2- Al finalizar la ejecución del bloque de instrucciones, se evalúa la condición lógica.

3- Si la condición lógica es verdadera, el ciclo vuelve a ejecutar el bloque de instrucciones. Si la condición lógica es falsa, el ciclo termina.

Importante:

Para que el ciclo termine en algún momento, debe haber una acción dentro de él que altere la condición evaluada, de modo que eventualmente esta se vuelva falsa y el ciclo se detenga, evitando que se convierta en un bucle infinito.

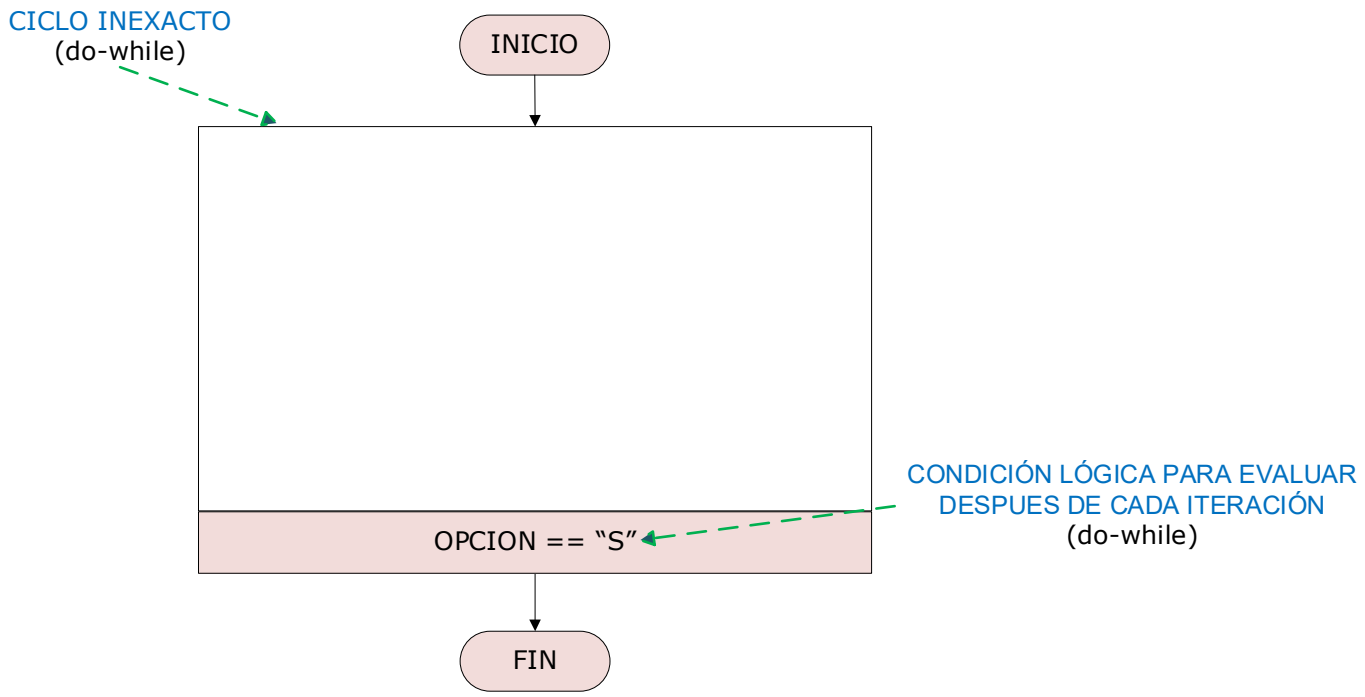
Ejemplo:

Desarrollar un algoritmo para un negocio que permita ingresar la cantidad de cierto articulo y el precio unitario de dicho artículo. Tras ingresar los datos de cada artículo, el programa debe preguntar: **"¿Deseas seguir ingresando más artículos? [S/N]"**. La carga de datos concluirá cuando el usuario ingrese "N". Al finalizar, el algoritmo debe mostrar el total de la venta realizada. Al menos siempre se procesa un artículo en la venta.

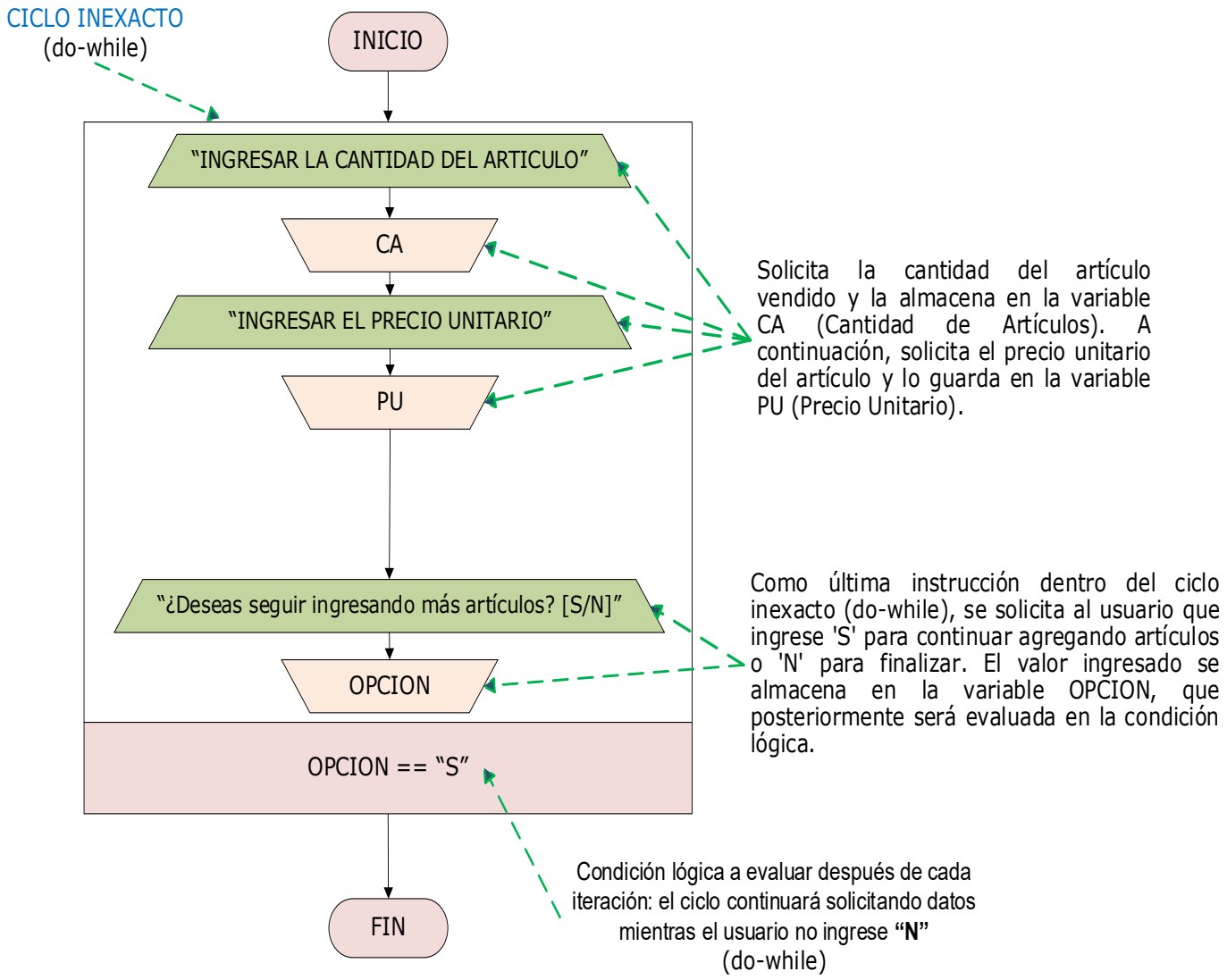
El algoritmo debe permitir ingresar la cantidad y el precio unitario de cada artículo. Después de ingresar estos datos, preguntará si se desean seguir ingresando más artículos. Si el usuario ingresa **"S"**, el proceso continúa. Si ingresa **"N"**, la carga de datos se detiene y se muestra el total de la venta.

Como se aclara al final del enunciado que se ingresará al menos un artículo, lo ideal es utilizar un ciclo inexacto do-while. De este modo, siempre ejecutaremos el bloque de instrucciones al menos una vez antes de evaluar la condición lógica.

Para determinar cuándo debe detenerse el ciclo, el enunciado indica que la entrada de artículos finaliza cuando el usuario ingresa una "N". Esto significa que el ciclo continuará siempre que el usuario ingrese una "S" para indicar que desea seguir añadiendo artículos. Es importante notar que no estamos realizando ninguna validación sobre los datos ingresados, por lo que confiamos en que el usuario ingresará correctamente los valores solicitados: "S" o "N". Podemos comenzar representando el ciclo y su condición.



Para seguir con el procedimiento, representaremos cómo se ingresan los datos de la cantidad de artículos y su precio unitario, además de solicitar la opción de continuar cargando más artículos o finalizar.



El último paso que vamos a representar es el cálculo de la suma total de todos los artículos vendidos. Cada venta está compuesta por la cantidad de artículos vendidos y su precio unitario. Para determinar el total de cada venta, se debe multiplicar la cantidad por el precio unitario (CA × PU). Luego, ese total se acumula en una variable, es decir, se suma progresivamente. Para esto, necesitamos una variable que actúe como acumulador, similar a las variables contadoras. Esta variable, que llamaremos VENTA_FINAL, debe inicializarse en 0 antes de comenzar a utilizarla.

Dentro del ciclo inexacto (do-while), iremos sumando en la variable VENTA_FINAL los importes calculados para cada artículo individual que se ingrese. Una vez que el ciclo termine, se mostrará el resultado de la suma total de la venta de todos los artículos, almacenado en VENTA_FINAL, a través de un mensaje.

Tanto la inicialización de VENTA_FINAL como el mensaje con el resultado final deben ubicarse fuera del ciclo. La inicialización de VENTA_FINAL se realiza solo una vez, antes de comenzar a acumular los importes de cada venta. Si la colocáramos

dentro del ciclo, la variable se reiniciaría a 0 en cada iteración, perdiendo el valor acumulado.

Por otro lado, el mensaje con el resultado de VENTA_FINAL también debe estar fuera del ciclo, ya que, si lo colocáramos dentro, se mostraría una suma parcial en cada iteración. Queremos que el mensaje aparezca solo una vez, al final del proceso, mostrando el resultado total de todos los artículos vendidos.