UTN FRGP
TUP

Técnico Universitario en Programación

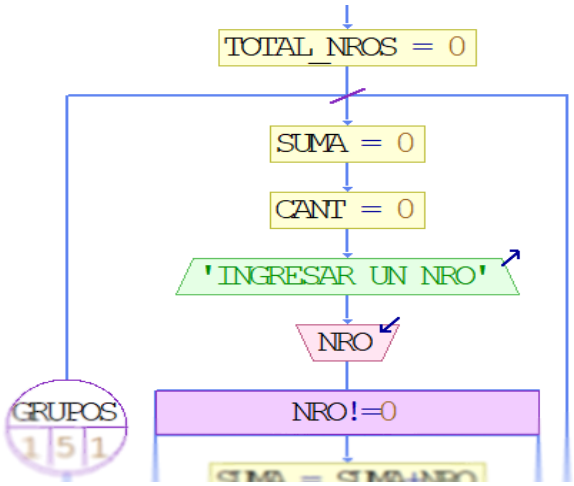
UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Gral. Pacheco

Apuntes de clase de la asignatura

Programación I

CICLOS COMBINADOS



2024

Abel Oscar Faure

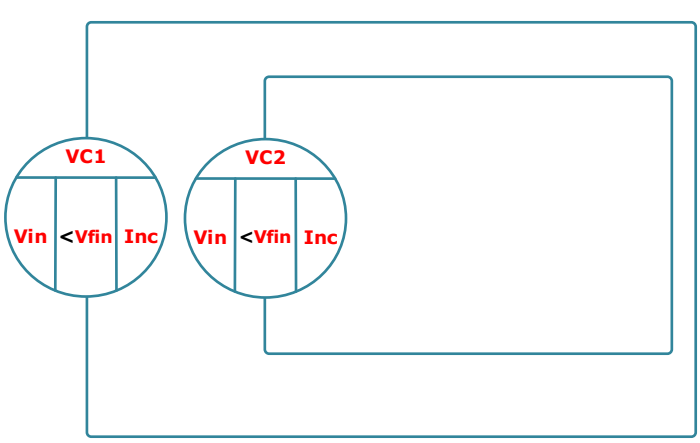
Lorena Raquel Palermo

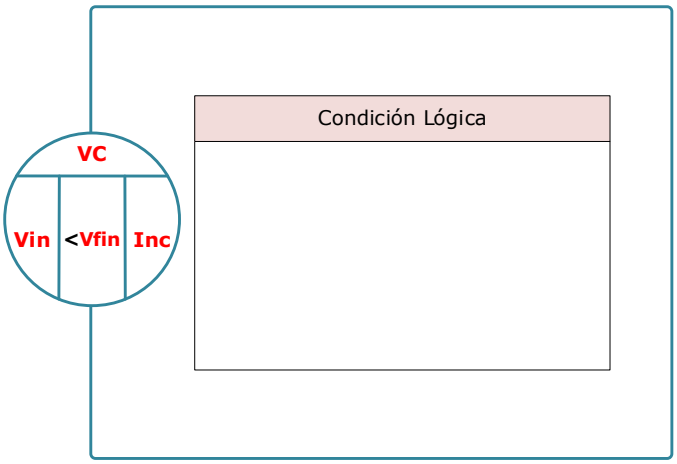
CICLOS COMBINADOS

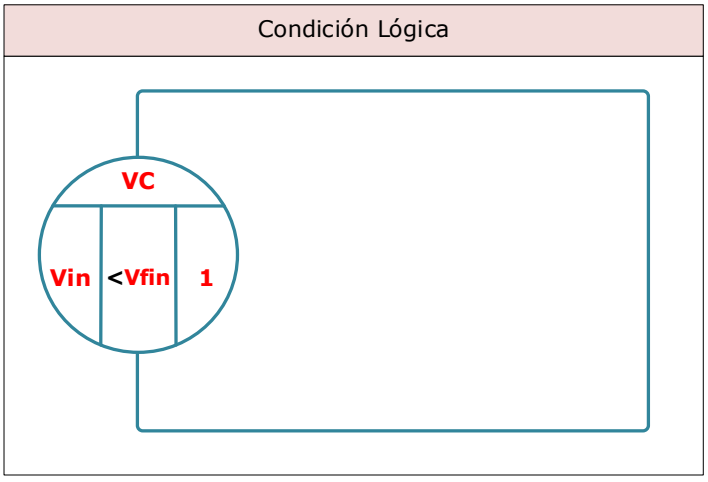
Los ciclos combinados en programación se presentan cuando se utilizan múltiples tipos de ciclos dentro de un mismo algoritmo. Generalmente, estos ciclos están anidados, lo que significa que un ciclo se encuentra contenido dentro de otro. El ciclo externo controla una iteración principal, mientras que el ciclo interno ejecuta tareas repetitivas relacionadas con la iteración del ciclo exterior.

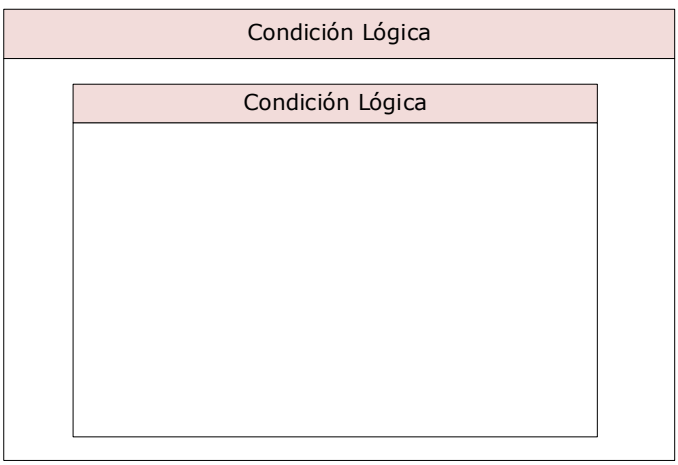
Este tipo de ciclos puede combinar diversas estructuras iterativas, como el ciclo exacto (for) y el ciclo inexacto (while o do-while), con el objetivo de permitir una ejecución más flexible de un conjunto de instrucciones. Los ciclos combinados son especialmente útiles cuando una única estructura iterativa no es suficiente para resolver problemas más complejos.

Estos son algunos ejemplos de ciclos combinados con anidación, aunque pueden incluir más ciclos anidados y diferentes combinaciones. La estructura exacta siempre dependerá del tipo de resolución que se aplique al problema.









La complejidad de estos problemas suele derivarse de los datos que se deben procesar, los cuales pueden estar organizados de acuerdo con una jerarquía. Esta jerarquía de datos puede estructurarse a través de agrupamientos, grupos, subgrupos y ordenación, conceptos que permiten clasificar y organizar los datos de distintas maneras.

Agrupamientos:

El agrupamiento es el proceso de organizar los elementos en grupos siguiendo ciertas reglas o criterios. Esto puede hacerse en varios niveles, como en una estructura de árbol donde los grupos se dividen en categorías y subcategorías.

Grupos y subgrupos:

Un grupo es un conjunto de elementos que tienen algo en común. Un subgrupo es un grupo más pequeño dentro del grupo principal, con alguna característica que la diferencia del resto. Los subgrupos pueden estar dentro de grupos más grandes.

Ordenación:

La ordenación es poner los elementos en un orden específico, ya sea ascendente, descendente, alfabético o numérico. Dentro de una jerarquía de datos, los elementos pueden estar organizados para que sea más fácil encontrarlos o analizarlos.

Ejemplo de jerarquía de datos:

- **Nivel superior:** Grupo de datos generales.
- **Niveles intermedios:** Subgrupos que dividen el grupo en partes más específicas.
- **Nivel de ordenación:** Los elementos de los grupos o subgrupos pueden estar ordenados de acuerdo con un criterio.
- **Agrupamiento:** El proceso que organizó los datos en grupos y subgrupos.

En los ciclos combinados, la jerarquía de datos desempeña un papel crucial. La ejecución sigue un orden jerárquico donde los ciclos internos dependen de los externos. El ciclo externo inicia la ejecución y, por cada iteración de este, el ciclo interno se ejecuta en su totalidad.

Esta estructura jerárquica define la prioridad de los ciclos en la ejecución y cómo se sincronizan entre ellos.

Cada ciclo puede tener diferentes condiciones de inicio y finalización, por lo que es fundamental gestionarlas adecuadamente para evitar bucles infinitos o errores lógicos.

Finalmente, las variables de control o aquellas que forman parte de la condición lógica de cada ciclo deben gestionarse cuidadosamente para evitar que los cambios en las variables del ciclo interno no afecten inesperadamente al ciclo externo.

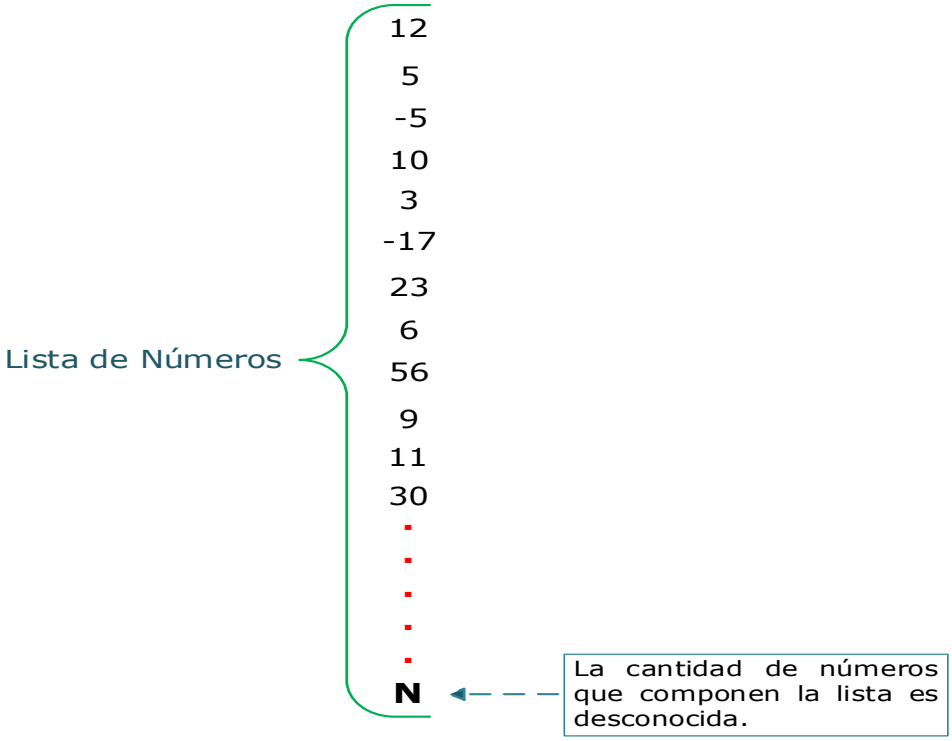
Ejemplo:

Dada una lista de números compuesta por 5 grupos y cada grupo separado del siguiente por un cero, se pide determinar e informar:

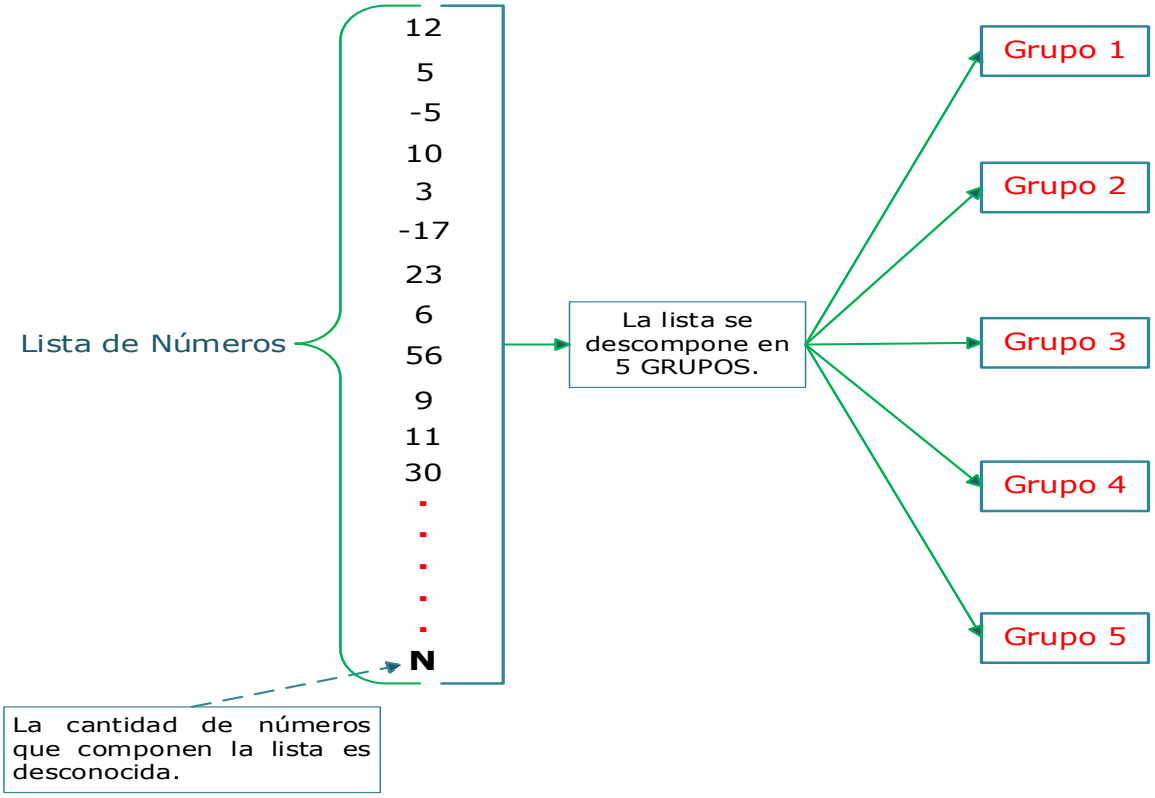
A. Calcular el promedio de valores de cada grupo.

B. Cuantos números había en total entre los 5 grupos.

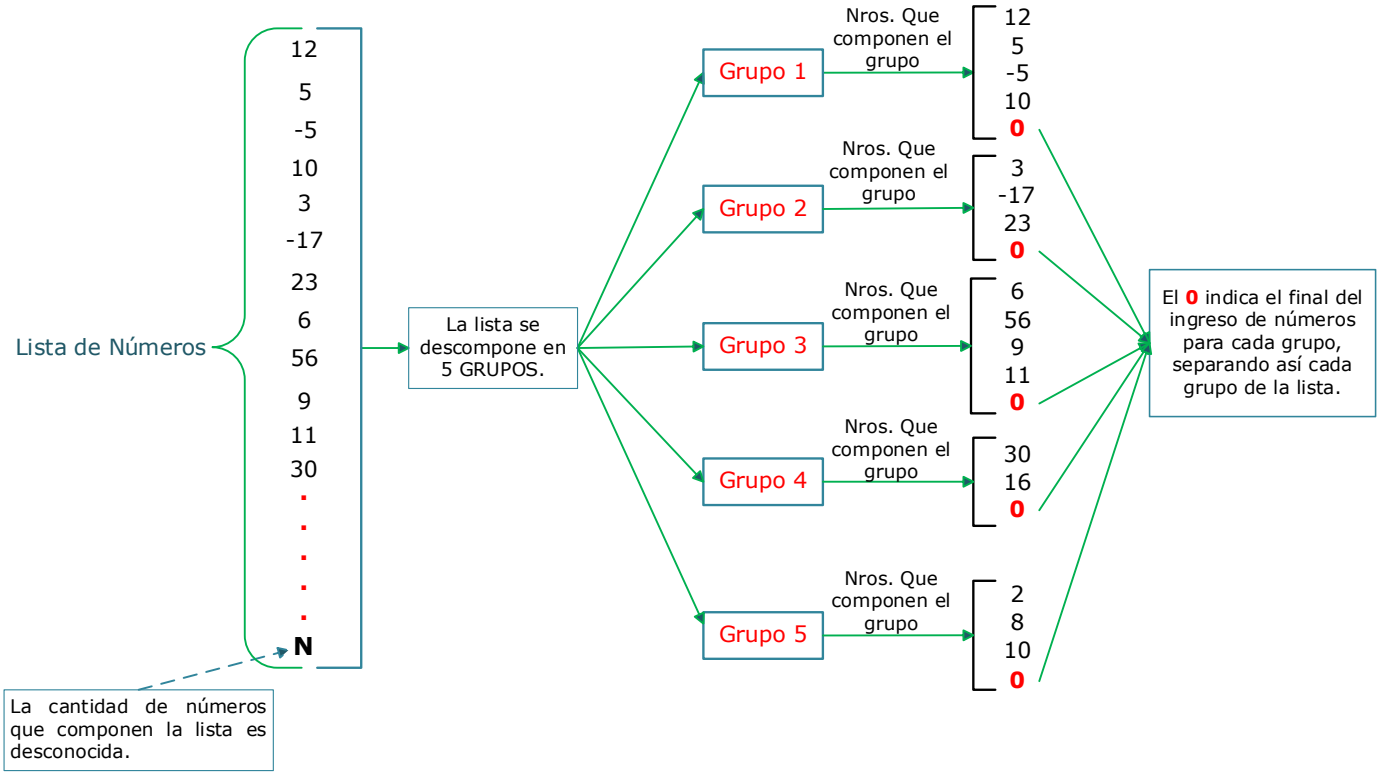
Antes de iniciar la resolución, es importante analizar y entender cómo está estructurada la lista de números. Esto es crucial para determinar qué tipo de ciclos repetitivos y cuántos necesitaremos para ingresar y procesar los datos correctamente, según lo que requiere el problema. Podemos empezar por ejemplificar la lista de números.



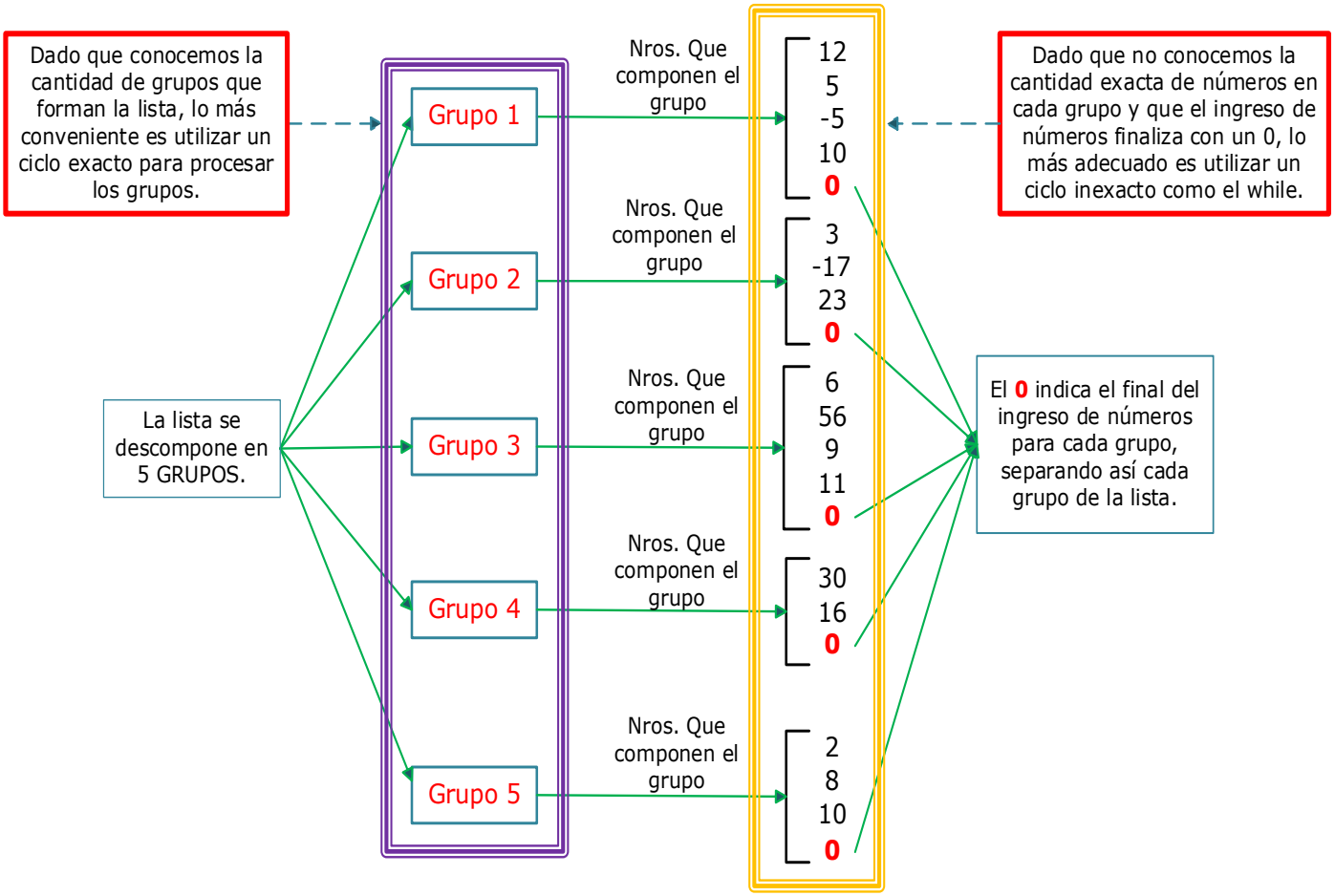
No se conoce la cantidad exacta de números en la lista, pero se indica que está formada por 5 grupos.



Cada grupo está compuesto por una cantidad indefinida de números. No se sabe cuántos números hay en cada grupo, pero el enunciado nos indica que cada grupo está separado del siguiente por un cero. Esto significa que se ingresan números en un grupo hasta que aparece un 0, lo que marca el fin del ingreso para ese grupo y el inicio del siguiente.



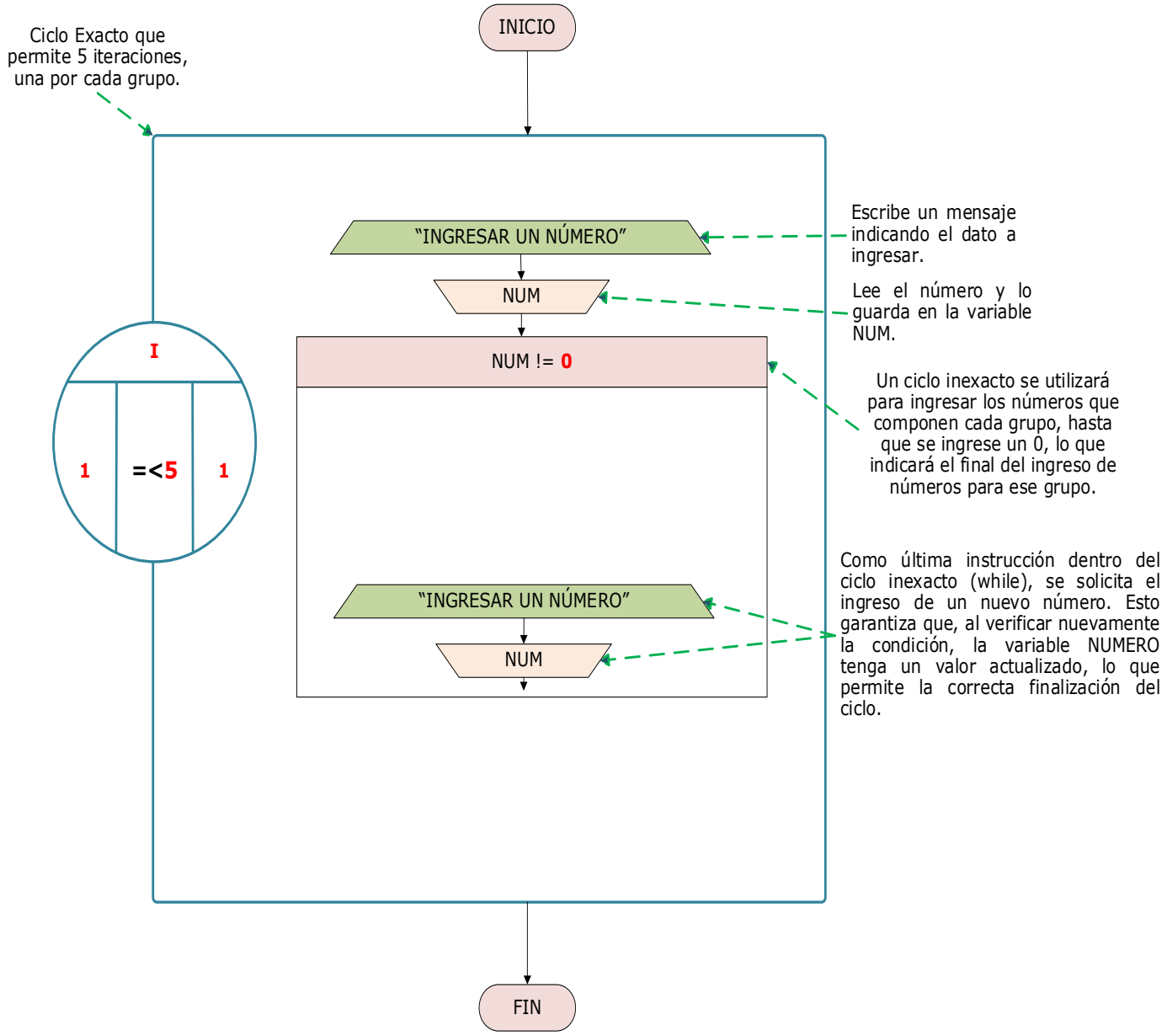
En la siguiente imagen podemos identificar y definir las estructuras cíclicas adecuadas para ingresar los números en cada grupo y procesarlos según lo que indica el enunciado del problema. Como conocemos que hay 5 grupos, lo más conveniente es utilizar un ciclo exacto que realice cinco iteraciones, una por cada grupo. Sin embargo, al no saber cuántos números componen cada grupo y sabiendo que el ingreso de números termina con un 0, utilizaremos un ciclo inexacto del tipo while para manejar el ingreso de números dentro de cada grupo.



Podemos comenzar diagramando cómo sería la estructura para ingresar los números que componen cada grupo. Como mencionamos, utilizaremos un ciclo exacto externo con 5 iteraciones, una por cada grupo. Dentro de este ciclo, emplearemos un ciclo inexacto del tipo while, que permitirá el ingreso de números hasta que se ingrese un 0, momento en el cual el ciclo finalizará.

Por cada iteración del ciclo externo (for), que gestiona los grupos, se ejecutará el ciclo interno (while), solicitando el ingreso de un número. La condición lógica de corte del ciclo interno será que la variable NUM tenga un valor distinto de cero; es decir, el ciclo continuará mientras no se ingrese un 0. Cuando esto ocurra, el ciclo interno finalizará, y el control pasará al ciclo externo, que continuará con las instruccio-

nes correspondientes y luego iniciará otra iteración, repitiendo el proceso para el siguiente grupo.



Ahora si pasamos a resolver los puntos A y B que nos solicita el problema:

Punto A:

Para este punto, en el que debemos calcular el promedio de los valores de cada grupo, necesitaremos un acumulador para sumar los números, al que llamaremos SUMA_NROS, y un contador de números, que llamaremos NROS_GRUPO. El contador se incrementa en 1 cada vez que se ingresa un número.

En cada iteración del ciclo externo (for), tanto el acumulador como el contador se inicializan en 0, ya que solo deben sumar y contar los números del grupo que se está procesando. Al comenzar la siguiente iteración del ciclo externo, ambos valores se vuelven a poner en 0 para procesar el siguiente grupo de números.

Como no sabemos cuántos números componen cada grupo, utilizamos un ciclo inexacto (while). La condición de este ciclo es que seguirá funcionando mientras el número ingresado sea distinto de 0. Por eso, pedimos el primer número fuera del ciclo while.

Dentro del ciclo while, sumamos cada número al acumulador SUMA_NROS y aumentamos en 1 el contador NROS_GRUPO. Este contador lleva el registro de la cantidad de números del grupo, lo cual será necesario para calcular el promedio.

Cuando se ingresa un 0, el ciclo while termina, ya que la condición de que el número sea distinto de 0 deja de cumplirse. A continuación, se calcula el promedio y se muestra en pantalla. Estas son las últimas instrucciones del ciclo externo (for).

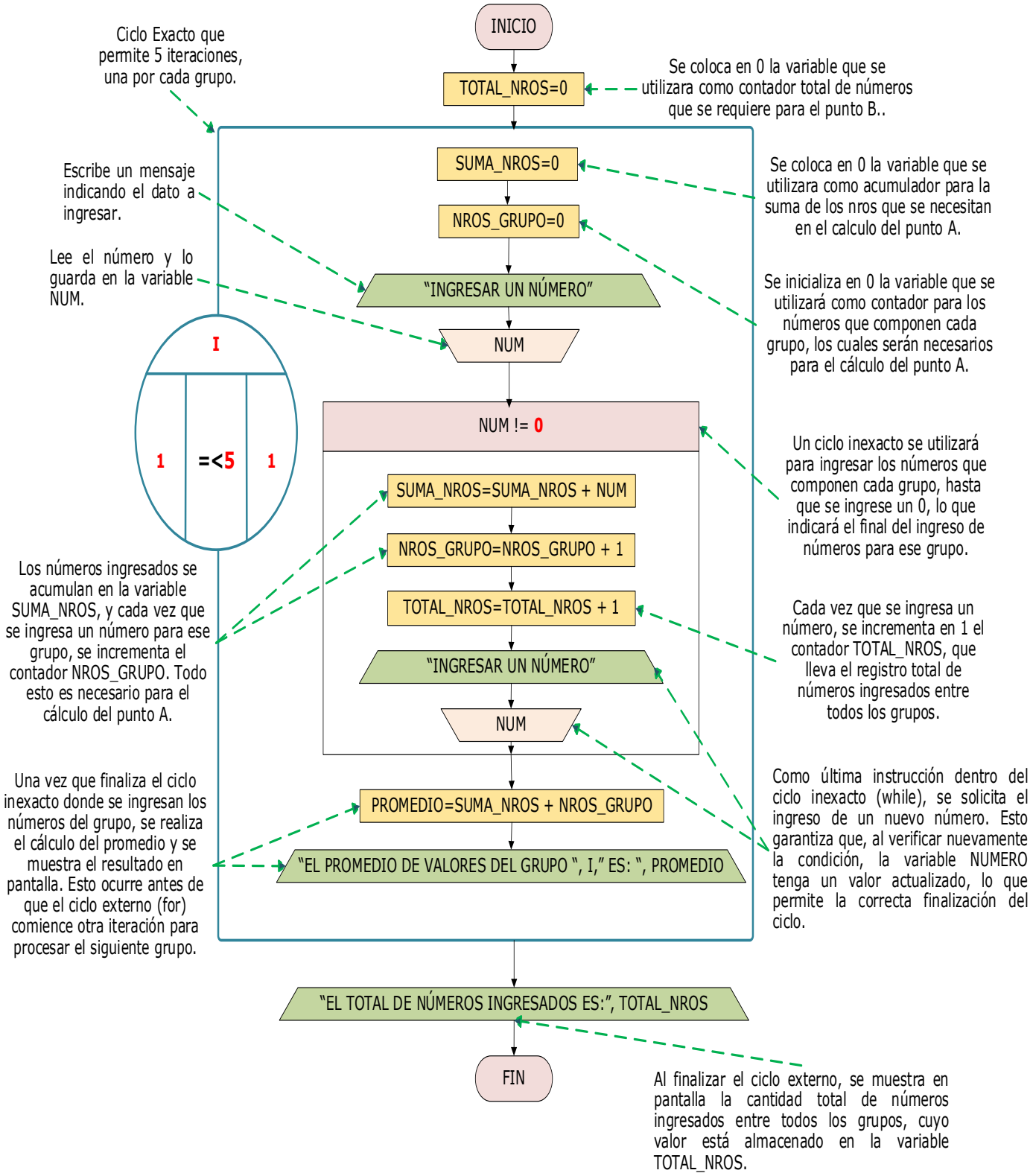
Si la variable de control del ciclo externo aún no ha alcanzado el valor final de 5, se inicia otra iteración y el proceso se repite para el siguiente grupo.

Punto B:

En este punto, debemos calcular el total de números ingresados entre todos los grupos. Para esto, utilizaremos un contador llamado TOTAL_NROS, que se incrementará en 1 cada vez que se ingrese un número, sin importar a qué grupo pertenezca.

Como TOTAL_NROS debe contar todos los números ingresados, lo inicializaremos una sola vez fuera del ciclo externo (for), a diferencia del acumulador y el contador del punto A, que se reinician en cada iteración del ciclo. Dado que los números se ingresan dentro del ciclo interno (while), es allí donde incrementaremos en 1 el contador TOTAL_NROS.

Una vez que se han procesado los 5 grupos de números, es decir, cuando el ciclo externo (for) finaliza, se muestra un mensaje con la cantidad total de números ingresados entre todos los grupos. Este valor estará almacenado en el contador TOTAL_NROS.


INICIO
TOTAL_NROS=0
SUMA_NROS=0
NROS_GRUPO=0
"INGRESAR UN NÚMERO"
NUM
NUM != 0
SUMA_NROS=SUMA_NROS + NUM
NROS_GRUPO=NROS_GRUPO + 1
TOTAL_NROS=TOTAL_NROS + 1
"INGRESAR UN NÚMERO"
NUM
PROMEDIO=SUMA_NROS + NROS_GRUPO
"EL PROMEDIO DE VALORES DEL GRUPO ", I," ES: ", PROMEDIO
"EL TOTAL DE NÚMEROS INGRESADOS ES:", TOTAL_NROS
FIN
I
1
=<5
1
Ciclo Exacto que permite 5 iteraciones, una por cada grupo.
Se coloca en 0 la variable que se utilizara como contador total de números que se requiere para el punto B..
Se coloca en 0 la variable que se utilizara como acumulador para la suma de los nros que se necesitan en el calculo del punto A.
Escribe un mensaje indicando el dato a ingresar.
Lee el número y lo guarda en la variable NUM.
Se inicializa en 0 la variable que se utilizará como contador para los números que componen cada grupo, los cuales serán necesarios para el cálculo del punto A.
Un ciclo inexacto se utilizará para ingresar los números que componen cada grupo, hasta que se ingrese un 0, lo que indicará el final del ingreso de números para ese grupo.
Los números ingresados se acumulan en la variable SUMA_NROS, y cada vez que se ingresa un número para ese grupo, se incrementa el contador NROS_GRUPO. Todo esto es necesario para el cálculo del punto A.
Cada vez que se ingresa un número, se incrementa en 1 el contador TOTAL_NROS, que lleva el registro total de números ingresados entre todos los grupos.
Una vez que finaliza el ciclo inexacto donde se ingresan los números del grupo, se realiza el cálculo del promedio y se muestra el resultado en pantalla. Esto ocurre antes de que el ciclo externo (for) comience otra iteración para procesar el siguiente grupo.
Como última instrucción dentro del ciclo inexacto (while), se solicita el ingreso de un nuevo número. Esto garantiza que, al verificar nuevamente la condición, la variable NUMERO tenga un valor actualizado, lo que permite la correcta finalización del ciclo.
Al finalizar el ciclo externo, se muestra en pantalla la cantidad total de números ingresados entre todos los grupos, cuyo valor está almacenado en la variable TOTAL_NROS.