

# **INTRODUCCIÓN A LA PROGRAMACIÓN I**

## **Algoritmos y Programación en Pseudocódigo**

### **Anexo - Tema 1**

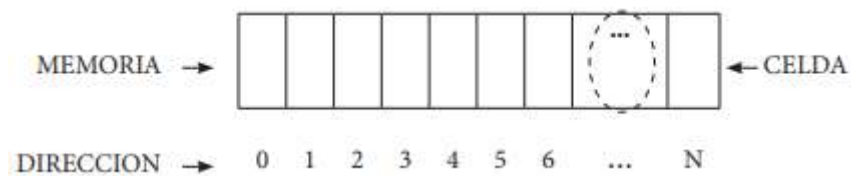
## Contenido

1.	Tipos de Datos y Expresiones .....	1
1.1	Tipos de Datos .....	1
1.2	Identificadores.....	2
1.3	Expresiones aritméticas.....	2
2.	Algoritmo y Pseudocódigo.....	4
2.1	Características de un algoritmo.....	4
2.2	Tipos de Instrucciones .....	5
2.3	Instrucciones Secuenciales .....	5
2.4	Formato General del Pseudocódigo .....	10
3.	Instrucciones de Selección .....	11
3.1	Estructuras de Selección Simple.....	11
3.2	Estructuras de Selección Compuesta .....	13
3.3	Estructuras de Selección Anidada.....	14
3.4	Estructuras de Selección Múltiple.....	15
4.	Instrucciones de Repetición .....	16
4.1	Estructuras de Repetición “mientras” .....	16
4.2	Estructuras de Repetición “Haga mientras” .....	17
4.3	Estructuras de Repetición “Repita hasta” .....	18
4.4	Estructuras de Repetición “Para” .....	19
5.	Enunciados de Ejercicios .....	21
5.1	Ejercicio 1 .....	21
5.2	Ejercicio 2 .....	21
5.3	Ejercicio 3 .....	21
6.	Respuestas a los Ejercicios.....	23
6.1	Respuesta Ejercicio 1.....	23
6.2	Respuesta Ejercicio 2.....	24
6.3	Respuesta Ejercicio 3.....	26

# 1. Tipos de Datos y Expresiones

## 1.1 Tipos de Datos

Todo dato que se utilice en un programa de ordenador debe ser almacenado en memoria. La memoria del ordenador está dividida en “trozos” del mismo tamaño dentro de los cuales se puede guardar información. Cada “trozo” es una celda y cada celda tiene asociada una dirección única en memoria que permite conocer su ubicación y acceder la información contenida en ella para consultarla, modificarla o borrarla.



Los datos se pueden clasificar en: Simples o Estructurados, según la cantidad de celdas que se utilicen para almacenarlos. Los tipos Simples utilizan una sola celda, los Estructurados, más de una dependiendo de la cantidad de datos a almacenar.

Los tipos de datos simples que existen en el pseudocódigo son:

### Numéricos

Que a su vez se clasifican en:

- Enteros Ejemplo: 3, -3, 1234, 0, 6 -45
  - No Enteros o Número con Punto Flotante Ejemplo: 3.5, -2.02, 4.3
- Por tratarse de datos simples requieren de una sola celda para ser almacenados.  
Por ejemplo:



### Lógicos o Booleanos

Representan sólo dos valores: falso o verdadero, o en inglés false o true, que se abrevian con F y V.

### Caracteres

Pueden ser una letra del alfabeto, un dígito o un símbolo especial (incluido el espacio en blanco), a todos estos se les conoce como símbolos alfanuméricos.

- Ejemplo de letras: a, b, c; A, B, C
- Ejemplo de dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Ejemplo de símbolos: +, /, \*, ?, %, \$, #, !, , , ä, .

Este tipo de datos se escribe entre comillas simples, por ejemplo: 'a' '0' ',' 'W' En este caso los dígitos entre comillas son totalmente diferentes a los datos tipo Numérico, por tratarse de caracteres no pueden ser empleados en operaciones aritméticas.

Entre comillas puede haber más de un carácter, a esto se le conoce como cadena de caracteres o simplemente cadena. Las cadenas se escriben entre comillas dobles en lugar de comillas simples, y son un tipo de dato estructurado porque requieren más de una celda. Por ejemplo:

## 1.2 Identificadores

En un programa, las celdas no se reconocen por su dirección sino por un nombre o identificador.

El identificador de la celda debe iniciar con una letra, a continuación pueden existir más letras o números; el único símbolo permitido en un identificador es el guion bajo '\_'.



En pseudocódigo los identificadores son sensibles a mayúsculas y minúsculas, lo que significa que Suma, suma y SUMA no son el mismo identificador, por lo tanto corresponden a tres celdas diferentes.

Según el comportamiento de la información almacenada en las celdas, éstas se pueden clasificar en dos tipos:

**Constantes:** celdas cuyo valor nunca cambia durante la ejecución del programa.

**Variables:** celdas cuyo valor cambia durante la ejecución del programa. Cuando esto ocurre, el valor antiguo es reemplazado por el nuevo sin modificar el tamaño de la celda.

## 1.3 Expresiones aritméticas

Toda expresión que se construya debe cumplir con:

Operando1 Operador Operando2

Donde Operando representa un dato y el Operador, el símbolo que representa la operación que se va a realizar.

### Operadores Aritméticos:

Son utilizados para construir expresiones aritméticas, los operandos son datos de tipo numéricos y el resultado obtenido también es numérico. En pseudocódigo existen los siguientes operadores aritméticos, ver Tabla:

*Tabla 1. Operadores Aritméticos*

Operador	Operación	Ejemplo	Resultado
**	Potencia	3**3	27
*	Multiplicación	3*3	9
/	División	3/2	1.5
MOD	Módulo	5 MOD 2	1
DIV	División entera	5 DIV 2	2
+	Suma	5 + 2	7
-	Resta	5 - 2	3

Fuente: Autores

### Operadores Relacionales:

Son utilizados para construir expresiones relacionales o de comparación, estas expresiones incluyen el manejo de todos los tipos de datos y el resultado obtenido es Verdadero o Falso.

En el Pseudocódigo existen los siguientes operadores relacionales, ver Tabla:

**Tabla 3. Operadores Relacionales**

Operador	Operación	Tipo de Datos
=	Igual que	Numéricos, Lógicos y Caracteres, Cadena
<>	Diferente	
>=	Mayor o igual que	Numéricos, Caracteres
<=	Menor o igual que	
>	Mayor que	
<	Menor que	

Fuente: Autores

Ejemplos:

Expresión	Resultado	Explicación
"hola" = "Hola"	F	Es F porque los caracteres mayúsculas y minúsculas son diferente
F <> V	V	Es V porque falso y verdadero son diferentes
3 <= 2	F	Es F porque 3 es mayor que 2
5 < 12	V	Es V porque 5 es menor que 12
'a' >= 'a'	V	Es V porque aunque 'a' no es mayor que 'a' son iguales

### Operadores Lógicos

Son utilizados para construir expresiones lógicas, estas expresiones incluyen el manejo de sólo los tipos de datos lógicos o booleanos y el resultado obtenido es Verdadero o Falso. Dentro de estos operadores existe uno solo que no necesita de dos operandos, el NOT, lo que lo convierte en un operador unario.

En el Pseudocódigo existen los siguientes operadores lógicos, ver Tabla:

**Tabla 4. Operadores Lógicos**

Operador	Operación	Ejemplo	Resultado
AND	Conjunción	V AND F	F
OR	Disyunción	F OR F	F
NOT	Negación	NOT F	V

Fuente: Autores

A continuación, se presenta el resultado de la aplicación de los tres operadores lógicos, por medio de sus tablas de verdad:

**Tabla 5. Operadores Lógicos**

A	NOT A	A	B	A AND B	A OR B
F	V	F	F	F	F
V	F	F	V	F	V
		V	F	F	V
		V	V	V	V

Fuente: Autores.

Los operadores lógicos, como los aritméticos, tienen precedencia de operadores y son asociativos por la izquierda.

OPERADOR	PRECEDENCIA	ASOCIACIÓN
( )	<p>Mayor</p>  <p>Menor</p>	
NOT		Por la derecha
AND		Por la izquierda
OR		Por la izquierda

## 2. Algoritmo y Pseudocódigo

### 2.1 Características de un algoritmo

Un Algoritmo es una secuencia de pasos para resolver un problema y debe contar con las siguientes características:

- **Preciso:** cada paso debe ser claro y exacto en su construcción para que así determine puntualmente lo que se desea hacer.
- **Definido:** toda vez que se ejecute el algoritmo con los mismos datos de entrada, éste debe generar el mismo resultado.
- **Finito:** todo algoritmo debe tener un fin.

El algoritmo se construye usando palabras del idioma y debe poder ser entendido por cualquier persona. Cuando se utilizan líneas de código se crea un programa y éstos se escriben usando lenguajes de programación.

Si se emplea cuasi código en la construcción del programa, se tiene un Pseudocódigo.

Un algoritmo se construye en forma general usando los siguientes pasos:

1. Inicio.
2. Capturar, conocer o ingresar todos los datos que permitan resolver el problema.
3. Resolver, calcular o llevar a cabo los procesos con los datos capturados. Se sugiere en este punto colocar las ecuaciones necesarias.
4. Mostrar, visualizar o imprimir todos los resultados que el problema exija.
5. Fin.

El siguiente ejemplo sencillo muestra un error clásico en la elaboración de un algoritmo: Construya un algoritmo que encuentre y muestre el valor del área de un rectángulo.

PASOS (incorrecto)

- 1 Conocer los valores de altura y base
- 2 Calcular el área
- 3 Mostrar el valor del área hallado

El algoritmo no es preciso en sus pasos porque no indica a que figura geométrica hay que calcularle el área, por lo tanto no se puede identificar la fórmula a aplicar. Además, para mostrar que un algoritmo termina es conveniente incluir como último paso la palabra fin.

El algoritmo para resolver el ejemplo anterior sería entonces:

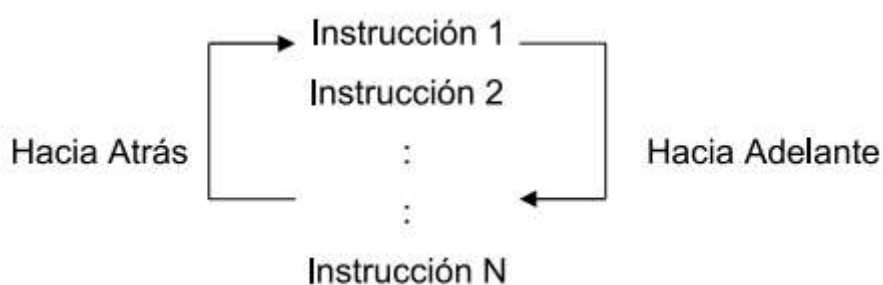
PASOS (correcto)

- 1 Inicio
- 2 Conocer los valores de altura y base del rectángulo
- 3 Calcular el área del rectángulo
- 4 Mostrar el valor del área del rectángulo
- 5 Fin

## 2.2 Tipos de Instrucciones

A partir de los **algoritmos** se pueden elaborar programas y para ello simplemente basta con “convertir” los pasos del algoritmo a sus instrucciones respectivas según el lenguaje de programación que se esté utilizando, pero, como existen diferentes lenguajes, inicialmente se construirán los programas usando una de las herramientas de programación: el **Pseudocódigo**.

El Pseudocódigo se construye con instrucciones que no son específicas de un lenguaje de programación pero que reflejan un comportamiento similar; al tener un programa en pseudocódigo se puede convertir éste con mayor facilidad a un lenguaje de programación. En pseudocódigo existen instrucciones que son de ejecución única y que se ejecutan de modo secuencial, denominadas **Instrucciones o Estructuras Secuenciales** e instrucciones de bifurcación que pueden ser a su vez de dos tipos: **Estructuras de Selección y Estructuras de Repetición**. En el caso de las estructuras de selección, las instrucciones se ejecutan hacia adelante, siguiendo la secuencia del cuerpo de instrucciones del programa mientras que en las estructuras de repetición las instrucciones se pueden ejecutar hacia adelante o hacia atrás dentro del bloque de ejecución del programa. En ambos casos se hace evaluando el resultado de una condición.



## 2.3 Instrucciones Secuenciales

### Instrucciones de Inicio y Fin

Se utilizan para dar comienzo y terminación al pseudocódigo, por cada programa escrito debe existir un solo Inicio y Fin. A continuación, se muestran como se escriben las instrucciones en el pseudocódigo y a su vez, su representación en la segunda herramienta de programación: el diagrama de flujo, ver Tabla:

**Tabla 7. Representación de las instrucciones Inicio y Fin en el diagrama de flujo.**

PSEUDOCÓDIGO Instrucción	DIAGRAMA DE FLUJO
Inicio	INICIO
Fin	FIN

Fuente: Autores

Inicio y Fin son Palabras Reservadas, es decir, palabras que pueden ser usadas únicamente para cumplir una función específica, por tanto, éstas no pueden ser usadas como identificadores.

### Instrucción de Asignación

Esta instrucción permite asignar el valor de una operación a un identificador de tipo variable. Siempre que se desee realizar un procedimiento, operación o cálculo – con operadores aritméticos, relacionales y/o lógicos –, se debe utilizar una instrucción de Asignación. En el mundo de las matemáticas se puede encontrar una expresión como la siguiente:

$$Y = X + 2$$

En donde para saber el valor de Y primero se debe conocer a X y luego llevar a cabo la operación de suma.

En el mundo del pseudocódigo la anterior expresión se convierte a:

$Y \leftarrow X + 2$

LADO DERECHO

El símbolo **Flecha** representa la instrucción de Asignación.

Para conocer el valor de Y primero se debe resolver el lado derecho de la instrucción. En forma general se interpreta toda instrucción de Asignación de la siguiente manera:

**“EL valor o el resultado del lado derecho de la flecha se almacena, guarda o asigna en el lado izquierdo”.**

Las letras al lado izquierdo de la asignación, como en el ejemplo anterior la Y, representan un **identificador**. Las letras al lado derecho de la asignación pueden representar un valor (de un tipo de dato determinado) o el resultado de una operación.

Veamos un ejemplo en el que la instrucción de la asignación se realiza con datos de tipo entero:



Ejemplos:

REPRESENTACIÓN DE LA EJECUCIÓN		
INSTRUCCIÓN	Ejecución de las instrucciones	Id
	X y Y son de tipo entero	
$X \leftarrow 3$	Al ejecutarse se almacena en X el valor de 3, así:	<div style="text-align: center;">X</div> <div style="border: 1px solid black; width: 50px; text-align: center; margin: 0 auto;">3</div>
$Y \leftarrow X + 2$	El contenido de la celda X se suma con el número 2 y se almacena en Y, entonces en Y se guarda el resultado, el número 5, así:	<div style="text-align: center;">Y</div> <div style="border: 1px solid black; width: 50px; text-align: center; margin: 0 auto;">5</div>

Para poder emplear un identificador dentro de una expresión, éste debe:

1. Estar declarado: lo que significa que debemos haber indicado que el tipo de dato de dicho identificador. (Más adelante veremos que al utilizar un lenguaje de programación esta declaración permite que el ordenador le asigne un espacio a dicho identificador en la memoria).
2. Estar inicializado: es necesario que los identificadores del lado derecho tengan un valor inicial para emplearlo en la expresión. (Ver ejemplo de tabla anterior: primero se asigna un valor a X y luego se utiliza en la operación siguiente para asignar el resultado al identificador Y).
3. Corresponder en tipo de dato: tanto los identificadores de la derecha como los de la izquierda deben ser del mismo tipo.

### Instrucción de Lectura

La gran mayoría de problemas necesitan para ser resueltos, datos ingresados por el usuario y éstos deben ser capturados por el programa, o datos que son leídos de un fichero. En el mundo del pseudocódigo existe una instrucción que representa la lectura de estos datos. Es la instrucción Leer(). Por tanto, Leer es una palabra reservada.

La instrucción Leer tiene el siguiente formato general en el pseudocódigo:

Leer ( a )

Leer (a, b, c)

Dentro del paréntesis se debe colocar el identificador o identificadores en los que se guardará la información leída.

Toda instrucción Leer debe cumplir con lo siguiente:

1. Lo(s) identificador(es) que se encuentre(n) dentro de los paréntesis, debe(n) estar declarados.
2. Debe existir coincidencia en el tipo de dato que se va leer con el tipo de dato que se ha colocado dentro de los paréntesis.
3. Así mismo, debe existir coincidencia entre el número de datos leído y el número de identificadores utilizados en la instrucción "Leer".

Por ejemplo: si se quiere capturar los datos de edad, peso y altura en cm de una persona se construye la instrucción Leer de la siguiente manera:

Entero edad  
 Flotante peso  
 Flotante altura → Con estas instrucciones declaramos los identificadores

**Leer** (edad, peso, altura)

Los datos a leer (introducir por el usuario o capturarse de un fichero) deben ser por ejemplo:

25 63,2 175

De este modo y al ejecutarse la instrucción “Leer” se guardará automáticamente 25 en el identificador “edad”, 63,2 en el identificador “peso” y 175 en el identificador “altura”. Una alternativa podría ser:


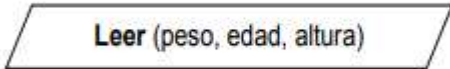
**Leer** (edad)

**Leer** (peso)

**Leer** (altura)

Utilizando diagramas de flujo la instrucción “Leer” se representa mediante un romboide:

**Tabla 8. Representación de la instrucción Leer en el diagrama de flujo.**

PSEUDOCÓDIGO Instrucción	DIAGRAMA DE FLUJO
<b>Leer</b> (A)	
<b>Leer</b> (peso, edad, altura)	

Fuente: Autores

## Instrucción de Escritura

Durante la solución de un problema a través de pseudocódigo, generalmente se necesita mostrar resultados parciales o totales, esto se logra con la instrucción Escribir.

De forma general, siempre que se desee mostrar, visualizar o imprimir los resultados, en el mundo del pseudocódigo se usa la instrucción Escribir(), siendo así una palabra reservada.

La instrucción Escribir tiene el siguiente formato:



**Escribir** ( )

← Aquí se puede colocar identificadores, texto o ambos

La instrucción Escribir puede tener los siguientes formatos:


1. Escribir ( A ) Cuando se ejecuta esta instrucción se muestra el contenido del identificador siempre y cuando esté declarado y contenga un dato.
2. Escribir ( “ texto “ ) Utilizaríamos este formato para mostrar algún comentario, información o texto en general
3. Escribir ( “ texto ”, A) Si se desea acompañar el contenido de una celda con un texto, se emplea este formato. El identificador debe estar declarado y contener un dato.

Algunos ejemplos son:

Tabla 9. Ejemplos de Instrucciones y ejecución									
INSTRUCCIÓN	REPRESENTACIÓN DE LA EJECUCIÓN								
	Ejecución de las instrucciones	Celda							
$T \leftarrow \text{"hola"}$ Escribir ( T )	Al ejecutarse se almacena en T el dato cadena "hola", Cuando se ejecuta la instrucción <b>Escribir</b> aparece en pantalla lo siguiente:	<div style="text-align: center;">T</div> <table><tr><td>h</td><td>o</td><td>l</td><td>a</td></tr></table> <div style="text-align: center;"><div style="border: 1px solid black; padding: 5px; display: inline-block;">hola</div><div style="color: red; font-weight: bold; margin-top: 10px;">Pantalla</div> </div>	h	o	l	a			
h	o	l	a						
$D \leftarrow 3.5$	Cuando se ejecuta se almacena en D 3.5	<div style="text-align: center;">D</div> <table><tr><td>3.5</td></tr></table>	3.5						
3.5									
$A \leftarrow 13$	Cuando se ejecuta se almacena en A 13	<div style="text-align: center;">A</div> <table><tr><td>13</td></tr></table>	13						
13									
$B \leftarrow \text{"sacaron"}$	Cuando se ejecuta se almacena en B sacaron	<div style="text-align: center;">B</div> <table><tr><td>s</td><td>a</td><td>c</td><td>a</td><td>r</td><td>o</td><td>n</td></tr></table>	s	a	c	a	r	o	n
s	a	c	a	r	o	n			
Escribir ( A , "estudiantes" , B, D )									
Cuando se ejecuta la instrucción <b>Escribir</b> se muestra en									
<div style="text-align: center;">Pantalla</div> <div style="text-align: center;"><div style="border: 1px solid black; padding: 10px; display: inline-block;">13 estudiantes sacaron 3,5</div><div style="color: red; font-weight: bold; margin-top: 10px;">Pantalla</div> </div>									

La representación de la instrucción en un diagrama de flujo es:


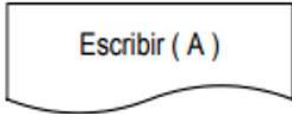
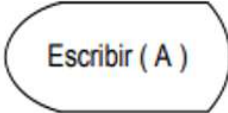
**Representación de la instrucción Escribir en el diagrama de flujo.**

PSEUDOCODIGO Instrucción	DIAGRAMA DE FLUJO
<b>Escribir (A)</b>	1. 

Este símbolo es idéntico al utilizado con la instrucción Leer(), es el símbolo que se utiliza por tanto para representar instrucciones de E/S.

Si además queremos especificar si los resultados se desean imprimir en una impresora o pantalla se utilizan los siguientes símbolos:

**Representación de la instrucción Escribir en el diagrama de flujo.**

PSEUDOCODIGO Instrucción	DIAGRAMA DE FLUJO
<b>Escribir (A)</b>	1. 
<b>Escribir (A)</b>	2.  3. 

## 2.4 Formato General del Pseudocódigo

Todo programa escrito en pseudocódigo está conformado por los siguientes tres bloques:

1. **La Cabecera** donde se coloca el nombre del programa escrito en pseudocódigo:  
Algoritmo\_nombre\_del\_programa
2. **El Bloque de Declaración** es donde se indica cuantos identificadores y de qué tipo se van a necesitar. El Bloque de Declaración puede contener dos secciones: Declaración de Variables y Declaración de Constantes.

**var**

Tipo\_de\_dato : Lista de identificadores

**const** Identificador = valor

Ejemplos:

**var**

Entero : A, B

**const**

Pi = 3.1416

**var**

Entero : A

Decimal : B

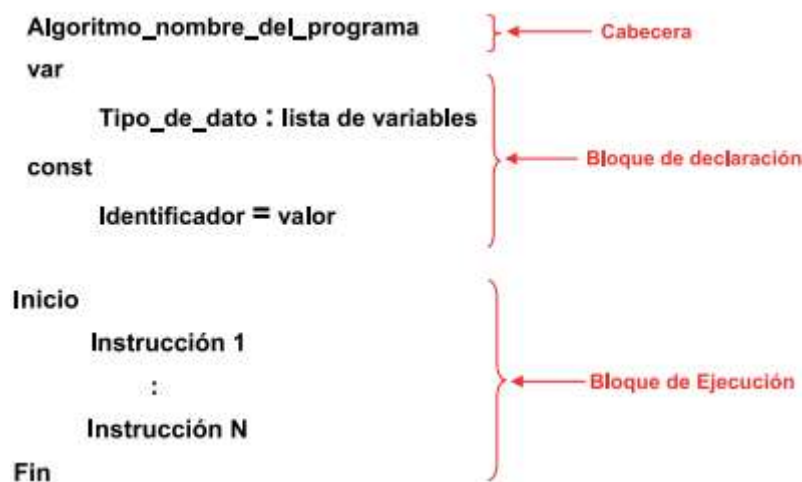
Caracter : C

Booleano: G

Cadena: S

3. **El Bloque de Ejecución:** es el cuerpo de instrucciones que ejecutará el algoritmo. Comienza y termina con las palabras reservadas Inicio y Fin, y en su interior podrá contener la repetición de las otras instrucciones las veces que sea necesario.

El formato general del pseudocódigo es:



### 3. Instrucciones de Selección

#### 3.1 Estructuras de Selección Simple

Tienen el siguiente formato:

```
Si ( condición ) entonces  
    Instrucción 1  
    Instrucción 2  
    :  
    :  
    Instrucción N  
Fin_si
```

La estructura de selección simple contiene tres palabras reservadas: **Si**, **entonces** y **Fin\_si**. Siempre comienza con la palabra Si y termina con Fin\_si. Además, contiene un cuerpo

de instrucciones entre ambas palabras, y una condición dentro de paréntesis. La condición al ser evaluada debe siempre generar un valor de tipo booleano: verdadero o falso.

La condición se puede construir de dos formas:

- Usando los operadores relacionales y/o operadores lógicos en una expresión que al ser resuelta genera un valor booleano
- O usando una variable de tipo booleano.

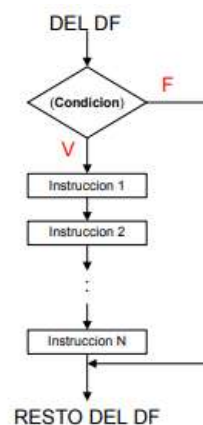
Cuando la **condición** genera un resultado **verdadero** automáticamente se ejecuta de una manera secuencial el cuerpo de instrucciones, y cuando se alcanza el “Fin\_si”, se da por terminada la ejecución del bloque que empezó con “Si”. A continuación ejecutará las instrucciones que se encuentren después del Fin\_si.



Cuando la **condición** genera un resultado **Falso** automáticamente se salta a la palabra reservada Fin\_si, se termina la ejecución del Si y se sigue con la ejecución del resto de instrucciones del programa.



La siguiente ilustración muestra el formato general de un diagrama de flujo para la estructura de selección simple:



### 3.2 Estructuras de Selección Compuesta

La estructura de selección compuesta tiene el siguiente formato general en el pseudocódigo:

**Si (condición) entonces**

Instrucción 1  
Instrucción 2  
:  
:  
Instrucción N

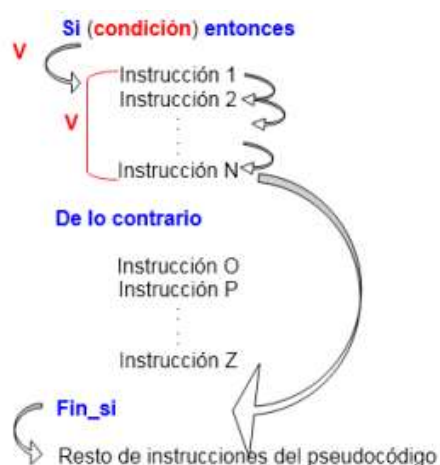
**De lo contrario**

Instrucción O  
Instrucción P  
:  
:  
Instrucción Z

**Fin\_si**

La Estructura de Selección Compuesta contiene cuatro palabras reservadas: **Si**, **entonces**, **De lo contrario** y **Fin\_si**. Contiene dos cuerpos de instrucciones: el primero es el que se ejecuta cuando se cumple la condición y el segundo el que se ejecuta en caso contrario. La condición al ser evaluada debe generar un valor booleano: verdadero o falso.

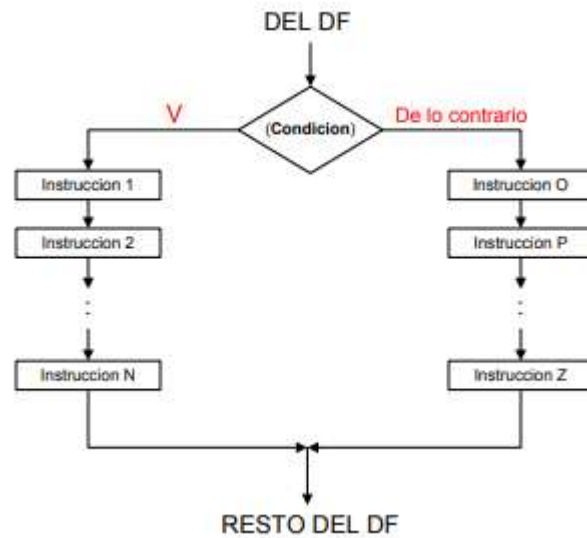
Condición = Verdadero:



Condición = Falso:



La siguiente ilustración muestra el formato general de un diagrama de flujo para la estructura de selección compuesta:



### 3.3 Estructuras de Selección Anidada

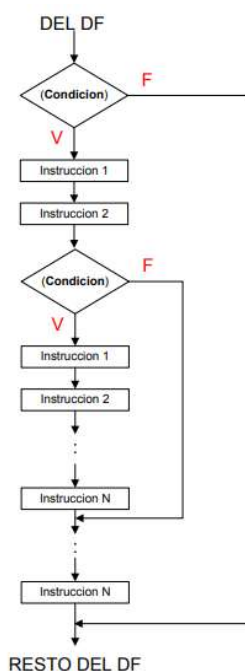
La estructura de selección anidada no tiene un formato general en el pseudocódigo, ni en el diagrama de flujo. Consiste en utilizar las estructuras anteriormente mencionadas dentro de alguna o varias de ellas mismas, por ejemplo, utilizar varias estructuras de selección simple dentro de otra estructura de selección simple, o dentro de una estructura de selección compuesta, o dentro de una estructura de selección múltiple, o viceversa.

Aspectos a tener en cuenta para construir una expresión de selección anidada:

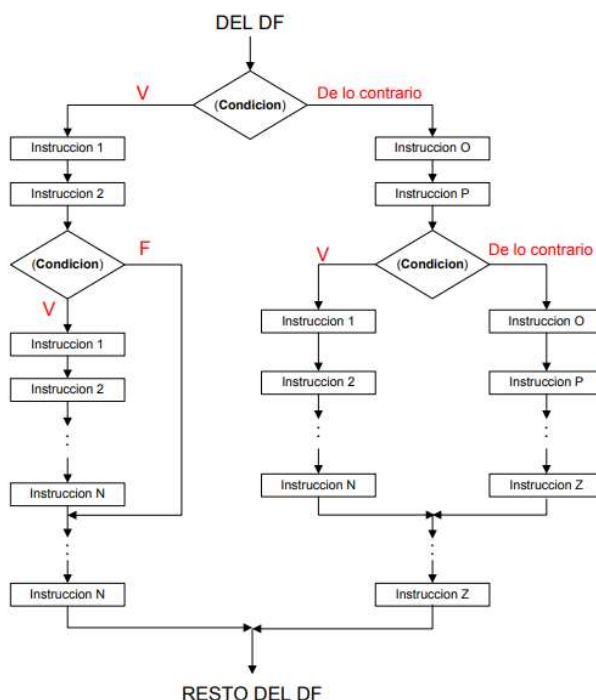
- Deben existir tantos Fin\_si como Si existan. Debe tenerse presente que un Fin\_si nunca cierra más de un Si.
- A el Fin\_si más interno le corresponderá única y exclusivamente al último Si más interno.

Ejemplos:

Simple anidado



Compuesto anidado:





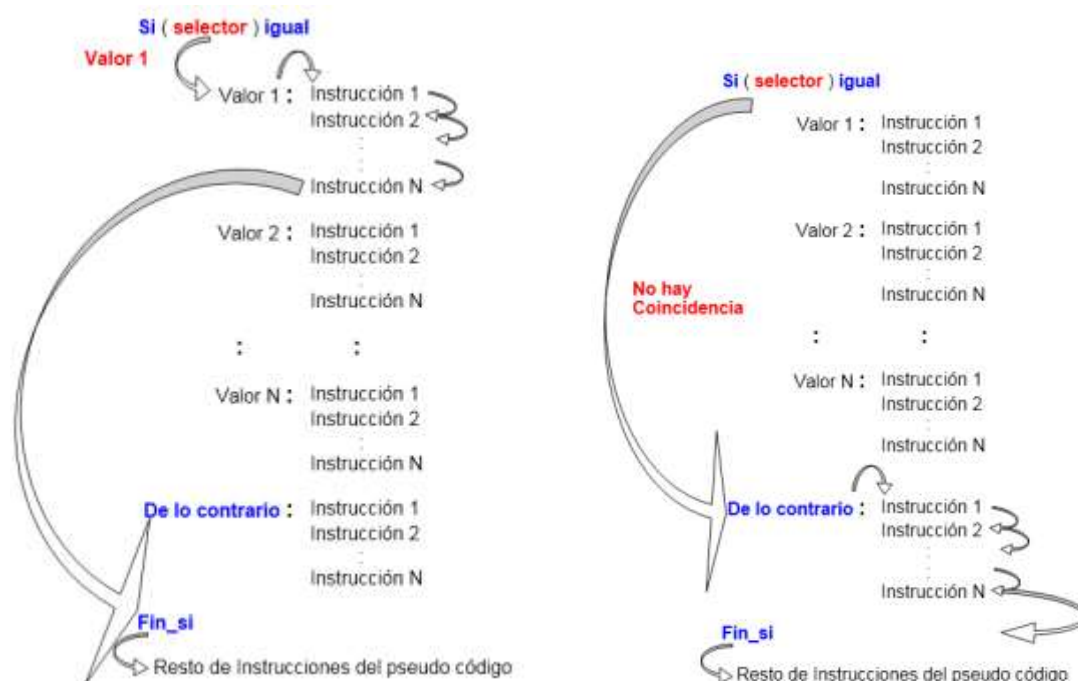
### 3.4 Estructuras de Selección Múltiple

La estructura de selección múltiple tiene el siguiente formato general en el pseudocódigo:

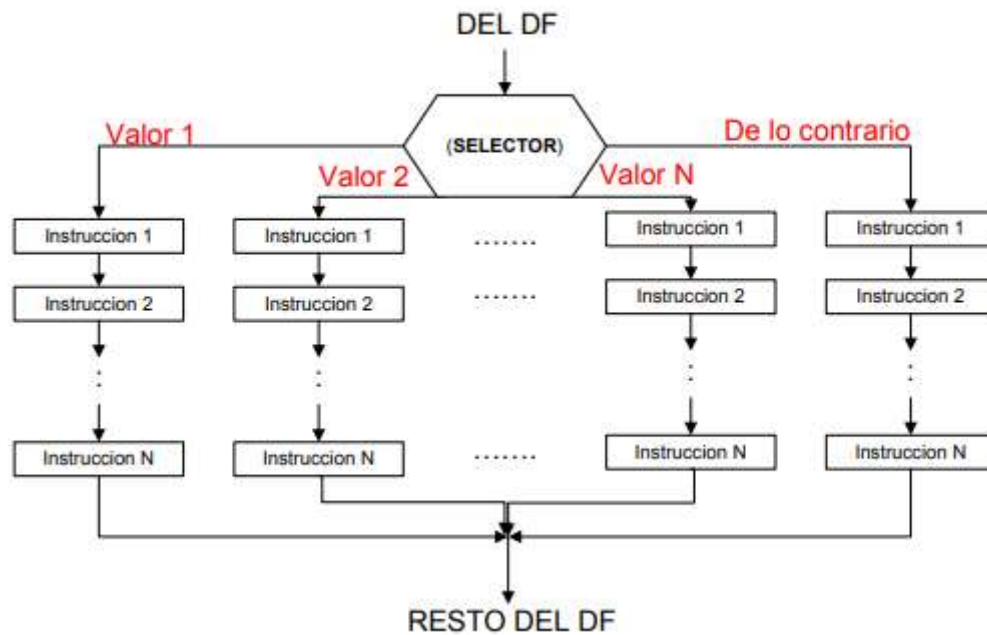
```
Si ( selector ) igual
    Valor 1 : Instrucción 1
              Instrucción 2
              :
    Valor 2 : Instrucción 1
              Instrucción 2
              :
              Instrucción N
    :
    Valor N : Instrucción 1
              Instrucción 2
              :
              Instrucción N
De lo contrario : Instrucción 1
                  Instrucción 2
                  :
                  Instrucción N
```

La estructura de selección múltiple utiliza un **selector** en vez de una condición. Está conformada por las palabras reservadas: **Si, igual, De lo contrario y Fin\_si**. Entre las palabras Si y Fin\_si, al lado izquierdo de dos puntos, se coloca toda la gama de valores posibles que puede generar el selector.

El selector es una variable o una expresión conformada por operadores que al ser resuelta genera un valor. Si el valor generado por el selector no se encuentra dentro de la gama de valores posibles, automáticamente se ejecutaría el bloque de instrucciones asociado a “De lo contrario”. El De lo contrario es opcional, esto significa que no es obligatorio colocarlo.



La siguiente ilustración muestra el formato general mediante un diagrama de flujo para la estructura de selección múltiple:



## 4. Instrucciones de Repetición

Las estructuras de repetición y las estructuras de selección pertenecen a las instrucciones de bifurcación y aunque ambas usan una condición para determinar la acción a ejecutar, las estructuras de repetición se diferencian de las de selección porque **permiten que ciertas instrucciones se ejecuten más de una vez hasta que se satisfaga la condición. Las estructuras de repetición también son conocidas como ciclos o bucles.**

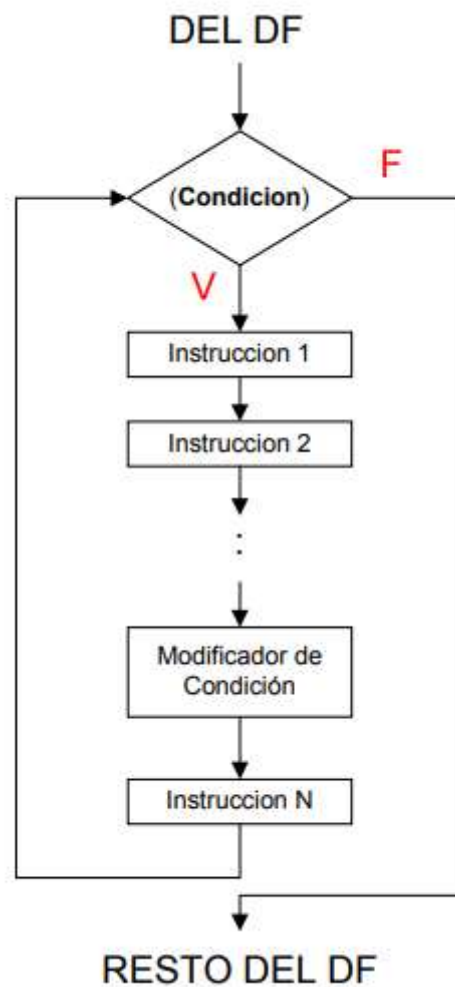
### 4.1 Estructuras de Repetición “mientras”

La estructura de repetición **Mientras** contiene tres palabras reservadas: **Mientras, haga y Fin\_mientras**. Comienza con la palabra **Mientras** y termina con **Fin\_mientras**. Además, contiene un cuerpo de instrucciones entre ambas palabras, y una condición dentro de paréntesis. La condición al ser evaluada debe siempre generar un valor de tipo booleano: verdadero o falso. Dentro del cuerpo de instrucciones debe contener una en particular, el Modificador de Condición, esta instrucción se encarga de cambiar el contenido de la variable que se encuentra en la condición, esto permite que en algún momento se termine la ejecución del ciclo. La ausencia o mala construcción del Modificador de Condición puede ocasionar ciclos infinitos (ciclos que nunca terminan) y por tanto, se pierde el control de la ejecución del programa.

```

Mientras ( condición ) haga
    Instrucción 1
    Instrucción 2
    :
    Modificador de Condición
    Instrucción N
Fin_mientras
  
```

La siguiente figura muestra el formato general de un diagrama de flujo para de la estructura de repetición Mientras Haga:



## 4.2 Estructuras de Repetición “Haga mientras”

La estructura de repetición Haga Mientras tiene el siguiente formato general en el pseudocódigo:

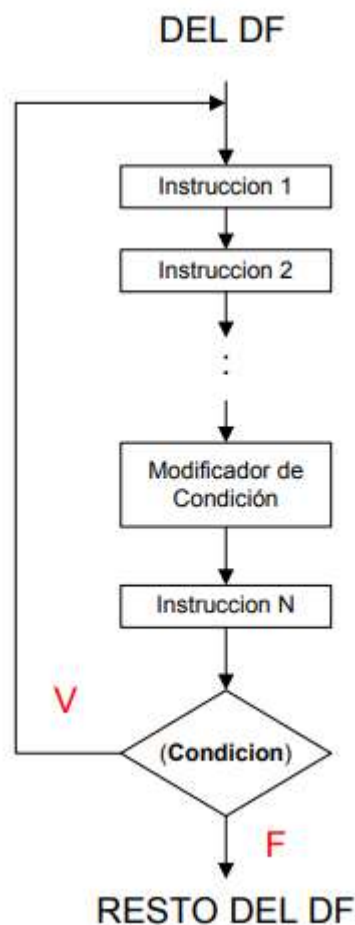
**Haga**

Instrucción 1  
Instrucción 2  
:  
Modificador de Condición  
Instrucción N

**Mientras ( condición )**

En la estructura de repetición Haga Mientras se evalúa la condición al final, por lo que todas las instrucciones que hacen parte de la estructura se ejecutan al menos una vez ya que se ejecutan antes de evaluar la condición, es decir, primero se ejecuta y luego se pregunta. Esa es la principal diferencia con la estructura de repetición Mientras.

La siguiente figura muestra el formato general del diagrama de flujo para la estructura de repetición Haga Mientras:



### 4.3 Estructuras de Repetición “Repita hasta”

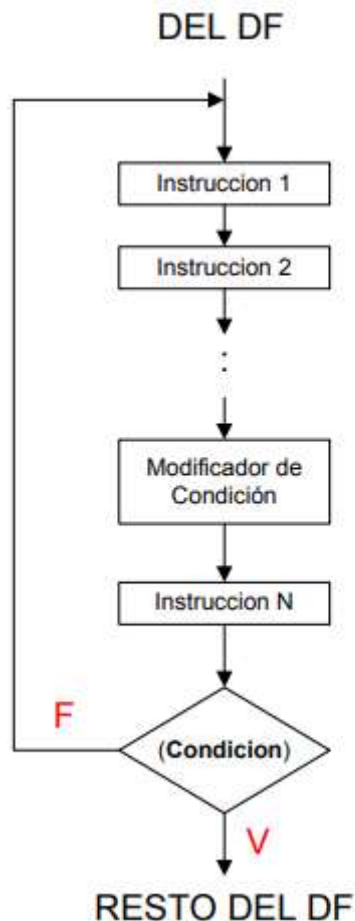
La estructura de repetición Repita Hasta tiene el siguiente formato general en el pseudocódigo:

```

Repita
    Instrucción 1
    Instrucción 2
    :
    Modificador de Condición
    Instrucción N
Hasta ( condición )
  
```

Igual que en la estructura de repetición “Haga Mientras”, la estructura de repetición Repita Hasta ejecuta las instrucciones asociadas a la estructura por lo menos una vez y luego pregunta. Sin embargo, en una estructura de repetición Repita Hasta se repite la ejecución del cuerpo de instrucciones, hasta que se cumple la condición es decir se repite la ejecución cuando la condición genera falso como resultado.

En siguiente figura se puede apreciar el formato general del diagrama de flujo para la estructura de repetición Repita Hasta:



#### 4.4 Estructuras de Repetición “Para”

La Estructura de Repetición Para tiene el siguiente formato general en el pseudocódigo:

**Para**  $V \leftarrow V_i$  **hasta**  $V_f$  (INC o DEC) **haga**

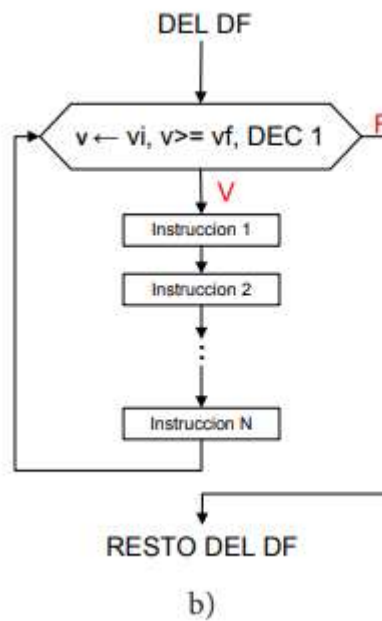
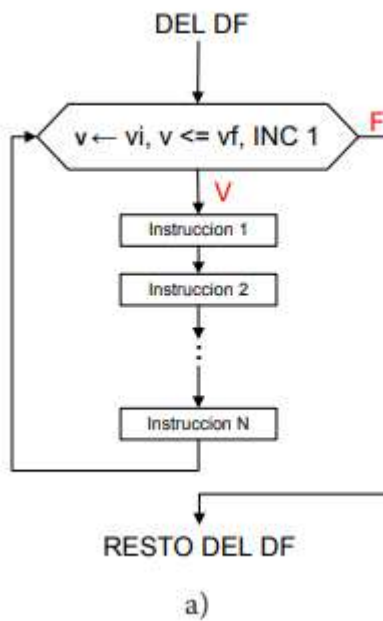
Instrucción 1  
 Instrucción 2  
 :  
 Instrucción N

**Fin\_para**

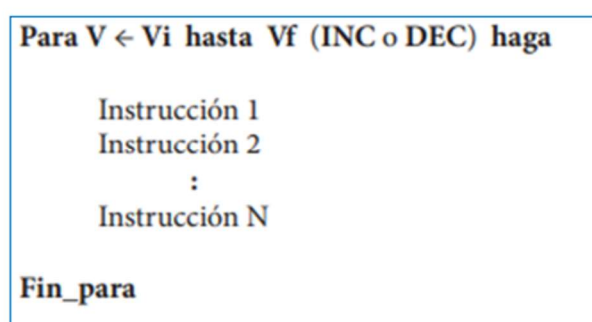
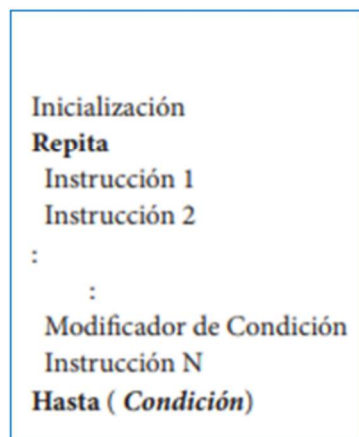
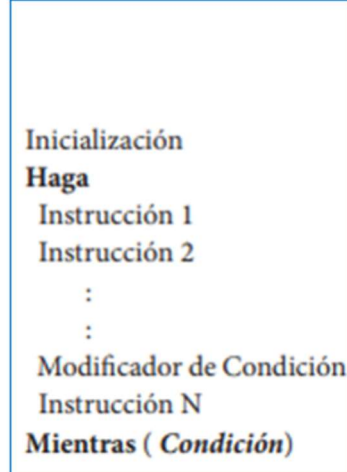
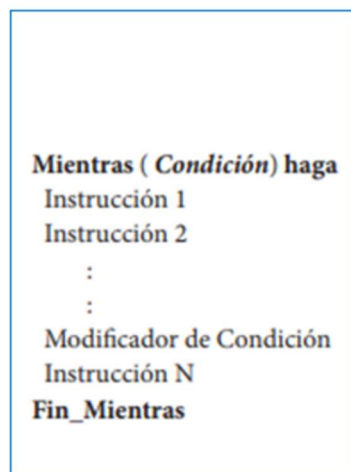
La estructura de repetición Para contiene las siguientes palabras reservadas: **Para**, **hasta**, **haga** y **Fin\_para**. Comienza con la palabra Para y termina con Fin\_para y dentro de éstas se encuentra el cuerpo de instrucciones.

Está conformada por la Inicialización de una variable ( $V \leftarrow V_i$ ), una variable final  $V_f$  con la cual comparamos el valor de V, lo que correspondería a la condición, y dos palabras reservadas - INC, DEC - que permiten el incremento o decremento de la variable V.

La siguiente figura muestra el formato general del diagrama de flujo para la Estructura de Repetición “Para”:



Resumiendo, podemos ver en la siguiente ilustración todas las estructuras de repetición vistas hasta ahora:



## 5. Enunciados de Ejercicios

### 5.1 Ejercicio 1

**Construye un algoritmo, pseudocódigo y diagrama de flujo tal que, conocida la base y altura de un rectángulo, calcule y muestre su área y perímetro.**

### 5.2 Ejercicio 2

**Dados tres números enteros positivos diferentes construya un pseudocódigo que permita determinar cuál es el mayor de ellos.**

Como se especifica “números enteros positivos” no son válidos los números negativos, por lo que será necesario utilizar la condición de una estructura de selección del tipo:

Si (  $A \geq 0$  Y  $B \geq 0$  Y  $C \geq 0$ ) entonces

Además, el ejercicio exige hallar el mayor de tres números diferentes por lo que los números no son válidos si son iguales, se requiere de la siguiente estructura de selección:

Si (  $A < B$  Y  $A < C$  Y  $B < C$ ) entonces

### 5.3 Ejercicio 3

**Construya un pseudocódigo tal que, lea un número entero N cualquiera y muestre la cantidad de términos y el resultado de la siguiente serie:**

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots \pm \frac{1}{N}$$

Nota que en la serie anterior N se encuentra en el denominador y que se incrementa de uno en uno. Además, los números pares de la serie están precedidos por el signo menos y los impares con el más. N representa la cantidad de números de dicha serie y su valor es introducido por el usuario mediante la instrucción “Leer”. Nota además que si el usuario introduce el número cero (0), indicando cero términos de la serie, el pseudocódigo en su ejecución no debe mostrar ni valor alguno ni término de la serie. El primer término (término 1) es el número 1.

El incremento de uno en uno ofrece la pista para la construcción de la serie y, para llevar a cabo esto, se utiliza la instrucción conocida como el *contador*, es decir:

$$X \leftarrow X + 1$$

El *contador* cumple tres funciones en este ejemplo, primero ofrece el número que se ubicará en el denominador, segundo, informa sobre los pares para así incluir el signo respectivo (- o +) en la expresión y, por último, lleva la cuenta de la cantidad de veces que se debe realizar o ejecutar la Estructura de Repetición.

Para saber si un número es par, se hace uso de la siguiente expresión:

$$Y \text{ MOD } 2 = 0$$

Es la operación de cálculo del resto de la división de la variable Y entre el número 2. (MOD). Si esta expresión es cierta el número será par (resto de la división = 0) y en caso contrario será impar.



## 6. Respuestas a los Ejercicios

### 6.1 Respuesta Ejercicio 1

Lo primero es comprender el problema, para ello se conceptualiza y determina el objetivo:

Conceptualización y Objetivo: el problema consiste en obtener de una figura geométrica conocida - el rectángulo - su área y perímetro. Para ello se deben utilizar las ecuaciones:  $A = B \times H$   $P = 2(B + H)$ . Por tanto, el objetivo es calcular el valor del área y del perímetro de un rectángulo, utilizando la base y altura ingresadas por el usuario.

Una vez comprendido el problema, se procede a construir el algoritmo:

1. Inicio
2. Capturar (Leer) la base y altura del rectángulo
3. Resolver las ecuaciones del área y perímetro del rectángulo  $A = B \times H$   $P = 2(B + H)$
4. Mostrar (Escribir) los resultados del área y perímetro del rectángulo
5. Fin

Una vez construido el algoritmo el siguiente paso es elaborar el pseudocódigo:

Se determina el nombre que se va a dar al programa y se coloca en la cabecera, por ejemplo: `Calculo_Area_y_Perimetro_Rectangulo`.

Para determinar los identificadores de la zona declaración de variables tendremos en cuenta los datos que necesitamos conocer para resolver el problema. En este caso: la base y la altura. Por tanto, necesitamos crear dos variables el nombre de cada una de ellas será: base y altura. Como los valores de estas variables son distancias y pueden tener decimales indicaremos que dichas variables son números reales (decimales).

Luego hay que pensar en cuantas variables necesitamos para almacenar los resultados. En nuestro caso, hay que hacer dos operaciones:  $A = B \times H$   $P = 2(B + H)$ , por tanto, usaremos dos variables cuyos nombres serán: área y perímetro y serán como en el caso anterior de tipo Real o Decimal.

A continuación, habría que pensar en las operaciones de captura, ejecución y muestra de los datos para concluir el algoritmo. En este caso y por ser muy sencillo abordaremos directamente su transcripción en pseudocódigo:

```
Algoritmo_calculo_area_y_perimetro_rectangulo ← Cabecera
var
    Decimal : base, altura, perímetro, area
const
    C1 = 2
Inicio
    Escribir (" ingrese base y altura del rectángulo ")
    Leer (base, altura)
    area ← base * altura
    perímetro ← C1 * ( base + altura)
    Escribir ("El área y perímetro del rectángulo son:",
    area, perímetro)
Fin
```

**Bloque de Declaración**

**Bloque de Ejecución**

## 6.2 Respuesta Ejercicio 2

### **ALGORITMO usando ESS**

```
Inicio
Capturar tres números enteros positivos.
Si (A>=0 Y B >= 0 Y C>=0) entonces
  Si (A<> B Y A<>C Y B<>C) entonces
    Si (A > B Y A > C) entonces
      Mostrar el mayor es A
    Fin_si
  Si (B > A Y B > C) entonces
    Mostrar el mayor es B
  Fin_si
  Si (C > A Y C > B) entonces
    Mostrar el mayor es C
  Fin_si
Fin_si
Si (A = B O A = C O B = C) entonces
  Mostrar existen números iguales
Fin_si
Fin_si
Si (A < 0 O B < 0 O C < 0) entonces
  Mostrar numero no valido
Fin_si
Fin
```

### **Pseudocódigo usando ESS**

```
Algoritmo_encuentra_mayor
var
  Entero : A,B,C
Inicio
  Escribir("Ingrese tres números enteros positivos
    diferentes")
  Leer (A,B,C)
  Si (A >= 0 Y B >= 0 Y C >= 0) entonces
    Si (A<>B Y A<>C Y B<>C) entonces
      Si (A > B Y A > C) entonces
        Escribir ("El numero mayor es:",A)
      Fin_si
    Si (B > A Y B > C) entonces
      Escribir ("El numero mayor es:",B)
    Fin_si
    Si (C > A Y C > B) entonces
      Escribir ("El numero mayor es:",C)
    Fin_si
  Fin_si
  Si ( A = B O A = C O B = C) entonces
    Escribir ("Existen números iguales")
  Fin_si
Fin_si
Si ( A < 0 O B < 0 O C < 0) entonces
  Escribir ("numero no valido")
Fin_si
Fin
```

### **ALGORITMO usando ESC**

Inicio

Capturar tres números enteros positivos.

Si (  $A \geq 0$  Y  $B \geq 0$  Y  $C \geq 0$  ) entonces

Si (  $A < B$  Y  $A < C$  Y  $B < C$  ) entonces

Si (  $A > B$  Y  $A > C$  ) entonces

Mostrar el mayor es A

De lo contrario

Si (  $B > A$  Y  $B > C$  ) entonces

Mostrar el mayor es B

De lo contrario

Mostrar el mayor es C

Fin\_si

Fin\_si

De lo contrario

Mostrar existen números iguales

Fin\_si

De lo contrario

Mostrar numero no valido

Fin\_si

Fin

### **Pseudocódigo usando ESC**

Algoritmo\_encuentra\_mayor

var

Entero : A,B,C

Inicio

Escribir("Ingrese tres números enteros positivos  
diferentes")

Leer (A,B,C)

Si (  $A \geq 0$  Y  $B \geq 0$  Y  $C \geq 0$  ) entonces

Si (  $A < B$  Y  $A < C$  Y  $B < C$  ) entonces

Si (  $A > B$  Y  $A > C$  ) entonces

Escribir ("El numero mayor es:",A)

De lo contrario

Si (  $B > A$  Y  $B > C$  ) entonces

Escribir ("El numero mayor es:",B)

De lo contrario

Escribir ("El numero mayor es:",C)

Fin\_si

Fin\_si

De lo contrario

Escribir ("Existen números iguales")

Fin\_si

De lo contrario

Escribir ("numero no valido")

Fin\_si

Fin

## 6.3 Respuesta Ejercicio 3

### Herr. ALGORITMO usando ESS

1. Inicio
2. Capturar la cantidad de términos de la serie a visualizar (TOPE).
3. Inicializar acumulador  
 Si (TOPE>0) entonces  
   Para X = 1 hasta TOPE haga  
     Si (X MOD 2 = 0) entonces  
       Crear serie con negativos  
       sum = sum - 1/X  
       visualizar termino  
     Fin\_si  
     Si (X MOD 2 <> 0) entonces  
       Crear serie con positivos  
       sum = sum + 1/X  
       visualizar termino  
     Fin\_si  
   Fin\_para  
   Mostrar el resultado de la serie  
   Fin\_si  
   Si(TOPE <= 0) entonces  
     Mostrar dato no valido  
   Fin\_si  
 4. Fin

### Pseudocódigo usando ESS y ERP

Algoritmo\_muestra\_terminos\_y\_resultado\_serie

```

var
    Decimal : sum
    Entero: X,N
Inicio
    Escribir(" Ingrese la cantidad de términos de la
    serie a
                visualizar y obtener su resultado ")
    Leer (N)
    sum ← 0
    Si (N >0) entonces
        Escribir ("La serie es:")
        Para X← 1 hasta N haga
            Si (X MOD 2 = 0) entonces
                sum ← sum - 1/X
                Escribir ( "- 1/", X)
            Fin_si
            Si (X MOD 2 <> 0) entonces
                sum ← sum + 1/X
                Si (X =1) entonces
                    Escribir ( "1")
                Fin_si
                Si (X <>1) entonces
                    Escribir ( "+ 1/", X)
                Fin_si
            Fin_si
        Fin_para
        Escribir ("El resultado de la serie es", sum)
    Fin_si
    Si (N<=0) entonces
        Escribir ("Dato no valido")
    Fin_si
Fin
  
```