
	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	1/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Guía práctica de estudio 05: Pseudocódigo

---



Elaborado por	Actualizado por:	Revisado por:
M.C. Edgar E. García Cano Ing. Jorge A. Solano Gálvez	M.C. Cintia Quezada Reyes M.C. Laura Sandoval Montaño	M.C. Laura Sandoval Montaño

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	2/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

# Guía práctica de estudio 05: Pseudocódigo

## Objetivo:

El alumno elaborará pseudocódigos que representen soluciones algorítmicas empleando la sintaxis y semántica adecuadas.

## Actividades:


- Elaborar un pseudocódigo que represente la solución algorítmica de un problema en el cual requiera el uso de la estructura de control de flujo condicional.
- A través de un pseudocódigo, representar la solución algorítmica de un problema en el cual requiera el uso de la estructura de control iterativa.

## Introducción

Una vez que un problema dado ha sido analizado (se obtiene el conjunto de datos de entrada y el conjunto de datos de salida esperado) y se ha diseñado un algoritmo que lo resuelva de manera eficiente (procesamiento de datos), se debe proceder a la etapa de codificación del algoritmo.

Para que la solución de un problema (algoritmo) pueda ser codificada, se debe generar una representación de éste. Una representación algorítmica elemental es el pseudocódigo.

Un pseudocódigo es la representación escrita de un algoritmo, es decir, muestra en forma de texto los pasos a seguir para solucionar un problema. El pseudocódigo posee una sintaxis propia para poder realizar la representación del algoritmo (solución de un problema).

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	3/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Sintaxis de pseudocódigo

El lenguaje pseudocódigo tiene diversas reglas semánticas y sintácticas. A continuación, se describen las más importantes:

1. Alcance del programa: Todo pseudocódigo está limitado por las etiquetas de INICIO y FIN. Dentro de estas etiquetas se deben escribir todas las instrucciones del algoritmo.
2. Palabras reservadas con mayúsculas: Todas las palabras propias del pseudocódigo deben de ser escritas en mayúsculas.
3. Sangría o tabulación: El pseudocódigo debe tener diversas alineaciones para que el código sea más fácil de entender y depurar.
4. Lectura / escritura: Para indicar lectura de datos se utiliza la etiqueta LEER. Para indicar escritura de datos se utiliza la etiqueta ESCRIBIR.

### Ejemplo

```
ESCRIBIR "Ingresar la altura del polígono"
LEER altura
```

5. Declaración de variables: la declaración de variables la definen un identificador (nombre), seguido de dos puntos, seguido del tipo de dato, es decir:


```
<nombreVariable>:<tipoDeDato>
```

Los tipos de datos que se pueden utilizar son:

```
ENTERO -> valor entero positivo y/o negativo
REAL -> valor con punto flotante y signo
BOOLEANO -> valor de dos estados: verdadero o falso
CARACTER -> valor tipo carácter
CADENA -> cadena de caracteres
```

### Ejemplo

```
contador: ENTERO
producto: REAL
continuar: BOOLEANO
```

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	4/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

Es posible declarar más de una variable de un mismo tipo de dato utilizando arreglos, indicando la cantidad de variables que se requieren, su sintaxis es la siguiente:

```
<nombreVariable>[cantidad]:<tipoDeDato>
```

#### Ejemplo

```
contador[5]: ENTERO    → 5 variables de tipo entero
division[3]: REAL      → 3 variables de tipo real
bandera[6]: BOOLEANO  → 6 variables de tipo booleano
```

Existe un tipo de dato compuesto, es decir, que puede contener uno o más tipos de datos simples diferentes. Este tipo de dato se conoce como registro o estructura y su sintaxis es la siguiente:

```
<nombreRegistro>:REG
  <nombreVariable_1>:<tipoDeDato>
  ...
  <nombreVariable_N>:<tipoDeDato>
FIN REG
```


Para crear una variable tipo registro se debe indicar el nombre del registro y el nombre de la variable. Para acceder a los datos del registro se hace uso del operador punto (.).

#### Ejemplo

```
domicilio:REG
  calle: CADENA
  número: ENTERO
  ciudad: CADENA
FIN REG
```

```
usuario:REG domicilio → variable llamada usuario de tipo registro
usuario.calle := "Av. Imán"
usuario.número := 3000
usuario.ciudad := "México"
```

Es posible crear variables constantes con la palabra reservada CONST, la cual indica que un identificador no cambia su valor durante todo el pseudocódigo. Las

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	5/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

constantes (por convención) se escriben con mayúsculas y se deben inicializar al momento de declararse.

### Ejemplo

```
NUM_MAX := 1000: REAL, CONST
```

- Operadores aritméticos: Se tiene la posibilidad de utilizar operadores aritméticos y lógicos:

Operadores aritméticos: suma (+), resta (-), multiplicación (\*), división (/), división entera (div) **esto ya es en programación hay que omitirla**, módulo (mod), exponenciación (^), asignación (:=).

Operadores lógicos: igualdad (=), Y-lógica o AND (&), O-lógica u OR (|), negación o NOT (!), relaciones de orden (<, >, <=, >=) y diferente (<>).

La tabla de verdad de los operadores lógicos Y (AND), O (OR) y Negación (NOT) se describe en la Tabla 1.


Tabla 1: Tabla de verdad de operadores lógicos

A	B	A & B	A   B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

NOTA: A y B son dos condiciones, el valor 0 indica falso y el valor 1 indica verdadero.

- Notación de camello. Para nombrar variables y nombres de funciones se debe hacer uso de la notación de camello.

En la notación de camello (llamada así porque parecen las jorobas de un camello) los nombres de cada palabra empiezan con mayúscula y el resto se escribe con minúsculas. Existen dos tipos de notaciones de camello: lower camel case que en la cual la primera letra de la variable inicia con minúscula y upper camel case en la cual todas las palabras inician con mayúscula. No se usan puntos ni guiones para separar

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	6/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

las palabras (a excepción de las constantes que utilizan guiones bajos). Además, para saber el tipo de variable se recomienda utilizar un prefijo.

#### Ejemplo

```

realAreaDelTriangulo: REAL    → lower camel case
EnteroRadioCirculo: REAL     → upper camel case

calcularArea()
obtenerPerimetro()

```

## Estructuras de control de flujo

Las estructuras de control de flujo permiten la ejecución condicional y la repetición de un conjunto de instrucciones.

Existen 3 estructuras de control: secuencial, condicional y repetitivas o iterativas.

### Estructura de control secuencial

Las estructuras de control secuenciales son las sentencias o declaraciones que se realizan una a continuación de otra en el orden en el que están escritas.

#### Ejemplo

```


INICIO
    x : REAL
    x := 5.8
    x := x * 2
FIN

```

## Estructuras de control condicionales (o selectivas)

Las estructuras de control condicionales permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa) y, dependiendo del resultado, se realiza uno u otro flujo de instrucciones. Estas estructuras son mutuamente excluyentes (o se realiza una acción o se realiza la otra)

La estructura de control de flujo más simple es la estructura condicional SI, su sintaxis es la siguiente:

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	7/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

SI condición ENTONCES

[Acciones]

FIN SI

Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se realizan las instrucciones del bloque [Acciones]. Si no se cumple la condición, se continúa con el flujo del pseudocódigo descartando las [Acciones].

### Ejemplo

INICIO

a,b: ENTERO

a := 3

b := 2

SI a > b ENTONCES

ESCRIBIR "a es mayor"

FIN SI

FIN

Prueba de escritorio:

Instrucción	a	b	salida
a := 3	3		
b := 2		2	
a > b			
			a es mayor

Además de la estructura de control condicional simple SI, existe la estructura condicional completa nombrada SI-DE LO CONTRARIO.

SI *expresión lógica*

ENTONCES


[Acciones ENTONCES]

DE LO CONTRARIO

[Acciones DE LO CONTRARIO]

FIN DEL SI

Se evalúa la *expresión lógica* y si se cumple (si es verdadera) se realizan las instrucciones del bloque SI [Acciones ENTONCES]. Si no se cumple la *expresión lógica* (si es falsa) se realizan las instrucciones del bloque DE LO CONTRARIO [Acciones DE LO CONTRARIO]. Al final de la estructura condicional, el flujo continúa con las estructuras que le sigan.

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	8/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Ejemplo

INICIO

a,b:ENTERO

a := 3

b := 5

SI a > b

ENTONCES

ESCRIBIR "a es mayor"

DE LO CONTRARIO

ESCRIBIR "b es mayor"

FIN DEL SI

FIN

Prueba de escritorio:

Instrucción	a	b	salida
a := 3	3		
b := 2		5	
a > b			
			b es mayor

Otra estructura de control condicional es SELECCIONAR-CASO, la cual valida el valor de la variable que está entre paréntesis y comprueba si es igual al valor que está definido en cada caso. Si la variable no tiene el valor de ningún caso se va a la instrucción por defecto (DEFECTO).

SELECCIONAR (variable) EN

CASO valor1 ->

[Acciones]

CASO valor2 ->

[Acciones]

CASO valor3 ->


[Acciones]

DEFECTO ->

[Acciones]

FIN SELECCIONAR



	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	9/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Ejemplo

```

INICIO
  a : ENTERO
  a := 1
  SELECCIONAR (a) EN
    CASO 1 ->
      ESCRIBIR "Iniciar sesión."
    CASO 2 ->
      ESCRIBIR "Registrarse."
    CASO 3 ->
      ESCRIBIR "Salir."
    DEFECTO ->
      ESCRIBIR "Opción inválida."
  FIN SELECCIONAR
FIN

```

Prueba de escritorio:

Instrucción	a	salida
a := 1	1	
		Iniciar sesión

## Estructuras de control iterativas o repetitivas

Las estructuras de control de flujo **iterativas o repetitivas** (también llamadas cíclicas) permiten realizar una serie de instrucciones mientras se cumpla la expresión lógica. Existen dos tipos de expresiones cíclicas **MIENTRAS** y **HACER- MIENTRAS**.


La estructura **MIENTRAS** (WHILE en inglés) primero valida la condición y si ésta es verdadera procede a realizar el bloque de instrucciones de la estructura [Acciones] y regresa a validar la condición, esto lo realiza mientras la condición sea verdadera; cuando la condición es Falsa (no se cumpla) se rompe el ciclo y continúa el flujo normal del pseudocódigo.

```

MIENTRAS condición ENTONCES
  [Acciones]
FIN MIENTRAS

```

El final de la estructura lo determina la etiqueta **FIN MIENTRAS**.

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	10/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Ejemplo

```

INICIO
    valorInicial,valorFinal:ENTERO
    valorInicial:=0
    valorFinal:=3
    MIENTRAS valorInicial < valorFinal
        ESCRIBIR valorInicial
        valorInicial := valorInicial + 1
    FIN MIENTRAS
FIN

```

Prueba de escritorio:

Instrucción	valorInicial	valorFinal	salida
valorInicial := 0	0		
valorFinal := 3		3	
valorInicial < valorFinal			
			0
valorInicial := valorInicial +1	1		
valorInicial < valorFinal			
			1
valorInicial := valorInicial +1	2		
valorInicial < valorFinal			
			2
valorInicial := valorInicial +1	3		
valorInicial < valorFinal			


La estructura HACER-MIENTRAS primero realiza las instrucciones descritas en la estructura y después valida la expresión lógica.

```

HACER
    [Acciones]
MIENTRAS condición

```

Si la condición se cumple vuelve a realizar las instrucciones de la estructura, de lo contrario rompe el ciclo y sigue el flujo del pseudocódigo. Esta estructura asegura que, por lo menos, se realiza una vez el bloque de la estructura, ya que primero realiza las instrucciones del bloque y después pregunta por la condición.

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	11/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

### Ejemplo


```

INICIO
    valorInicial,valorFinal:ENTERO
    valorInicial:=0
    valorFinal:=3
    HACER
        ESCRIBIR valorInicial
        valorInicial := valorInicial + 1
    MIENTRAS valorInicial < valorFinal
FIN

```

Prueba de escritorio:

Instrucción	valorInicial	valorFinal	salida
valorInicial := 0	0		
valorFinal := 3		3	
			0
valorInicial := valorInicial +1	1		
valorInicial < valorFinal			
			1
valorInicial := valorInicial +1	2		
valorInicial < valorFinal			
			2
valorInicial := valorInicial +1	3		
valorInicial < valorFinal			

	<b>Manual de prácticas del Laboratorio de Fundamentos de programación</b>	Código:	MADO-17
		Versión:	03
		Página	12/12
		Sección ISO	8.3
		Fecha de emisión	26 / agosto / 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de computación salas A y B	
La impresión de este documento es una copia no controlada			

## Bibliografía

- Metodología de la programación. Osvaldo Cairó, tercera edición, México D.F., Alfaomega 2005.



- Metodología de la programación a través de pseudocódigo. Miguel Ángel Rodríguez Almeida, primera edición, McGraw Hill

