



# INTRODUCCIÓN Y CONSTRUCCIÓN DE ALGORITMOS

**FAVA** - Formación en Ambientes Virtuales de Aprendizaje

**SENA** - Servicio Nacional de Aprendizaje.

## Estructura de contenidos

	Pág.
<b>Introducción</b>	<b>3</b>
<b>Mapa de contenido</b>	<b>4</b>
<b>Desarrollo de contenidos</b>	<b>5</b>
1. Concepto de algoritmos	5
2. Historia del algoritmo	5
3. Resolución de problemas con algoritmos	7
3.1 Análisis del problema	7
3.1.1 Análisis de ejercicios	8
3.1.2 Diseño de un algoritmo	11
3.1.3 Características de los algoritmos	11
4. Diseño de un algoritmo mediante diagrama de flujo	12
4.1 Definición diagrama de flujo	12
4.2 Reglas para la elaboración de un diagrama de flujo	13
4.2.1 Diseño de un algoritmo mediante pseudocódigo	14
4.2.2 Expresar el algoritmo en un lenguaje de programación	15
5. Solución de problemas con diagramas de flujo y lenguaje natural	16
5.1 Variables	16
5.1.1 Declaración de una variable	16
5.1.2 Asignación de valor a una variable	17
5.1.3 Solicitar una variable	17
5.2 Constantes	17
5.3 Contadores	17
5.4 Acumuladores	18
5.5 Identificadores	19
6. Estructuras algorítmicas o de programación	19
6.1 Estructura secuencial	20
6.2 Correspondencia de pseudocódigo a diagrama de flujo	20
6.3 Estructuras de decisión	21
6.3.1 Simples	21
6.3.2 Dobles	22
6.4 Estructuras cíclicas y/o repetitivas	23
6.4.1 Estructura para	23
6.4.2 Estructura mientras	24
6.4.3 Estructura repita	24
Glosario	25
Bibliografía	26
Control del documento	27

## Introducción y construcción de Algoritmos

### Introducción

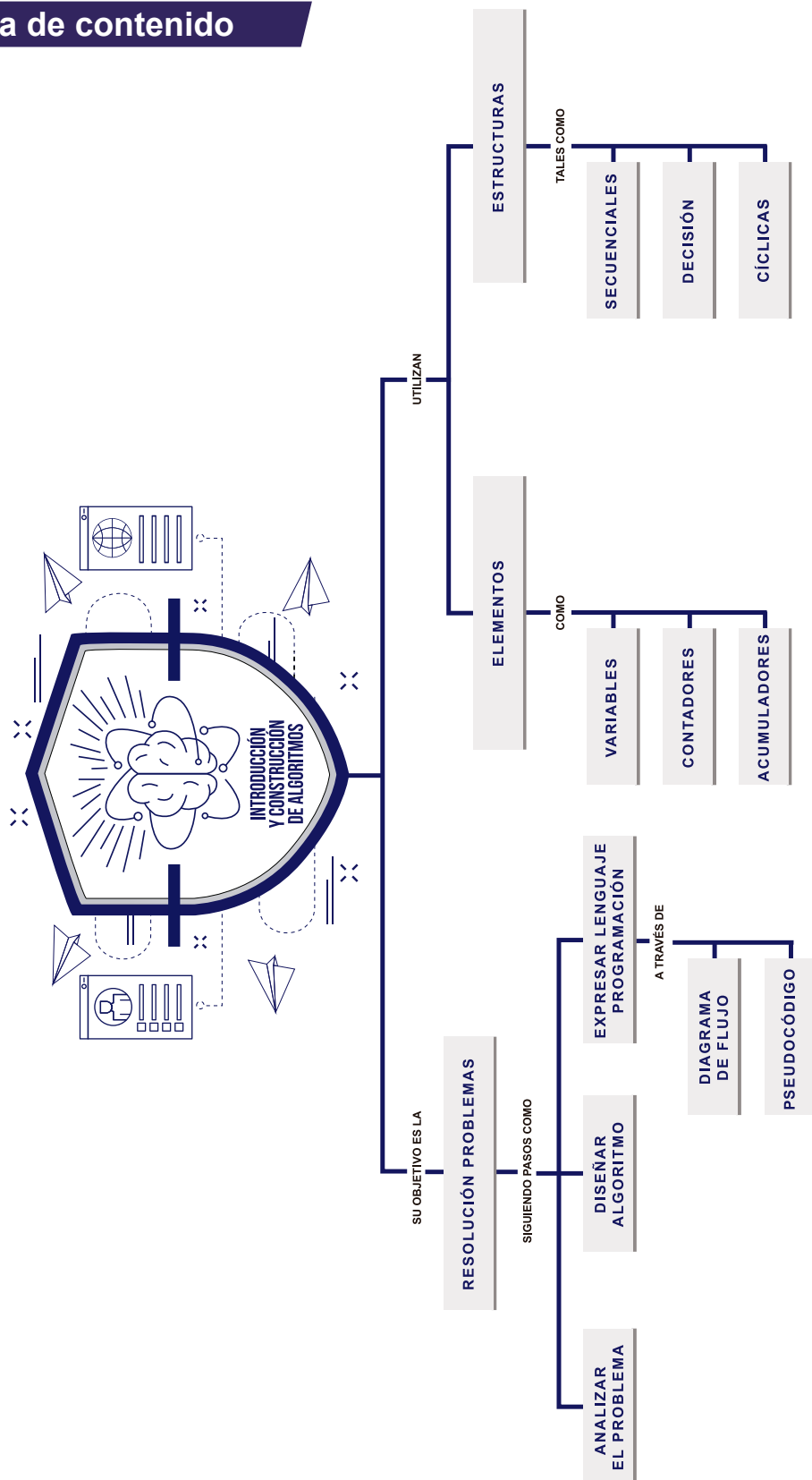
Así como el ser humano cuando va a ejecutar una acción recibe una serie de órdenes por medio de su cerebro indicándole que debe hacer y cómo lo debe hacer, algo parecido sucede con la construcción de un algoritmo donde se tiene como base una serie de pasos con un orden lógico para organizar un diagrama de flujo y así lograr un objetivo.

Hoy en día un computador realiza tareas y maneja datos obedeciendo secuencias de pasos lógicos para lo cual ha sido programado a través de algoritmos, los cuales permiten solucionar problemas por medio de la escritura secuencial (paso a paso) de lenguaje natural y organizado, para luego ser llevados a un programa basado en un lenguaje de programación y simular el ejercicio propuesto.



Es muy importante que usted como aprendiz logre desarrollar y despertar su mentalidad algorítmica, ya que durante todo su proceso estará inmerso en aprender a solucionar problemas durante la etapa lectiva y esto se logra por medio del diseño, creación e implementación de un algoritmo.

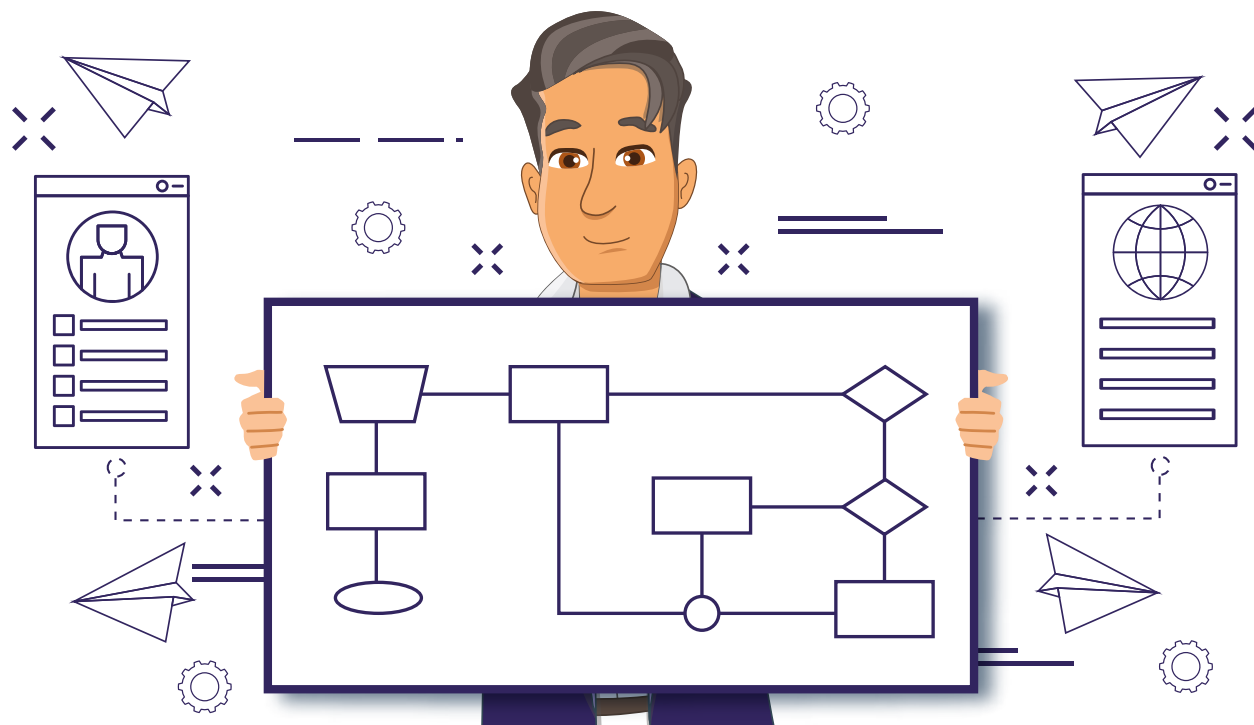
## Mapa de contenido



## Desarrollo de contenidos

### 1. Concepto de algoritmos

Un algoritmo se puede definir como un conjunto de instrucciones que conducen a la solución de un problema determinado, las cuales deben estar relacionadas lógicamente y ordenadamente.





## 2. Historia del Algoritmo

El Algoritmo es originario de Mohammed al-Khwarizmi, gran Matemático Persa (Siglo IX) reconocido como el padre del álgebra y como el introductor del sistema de numeración denominado arábigo. Este ilustre matemático donde solo se conserva la traducción al latín “Algoritmi de numero Indorum”, derivado al término “Algoritmo”, fue una de sus grandes obras y quien fue el primero en pensar en modo algorítmico.

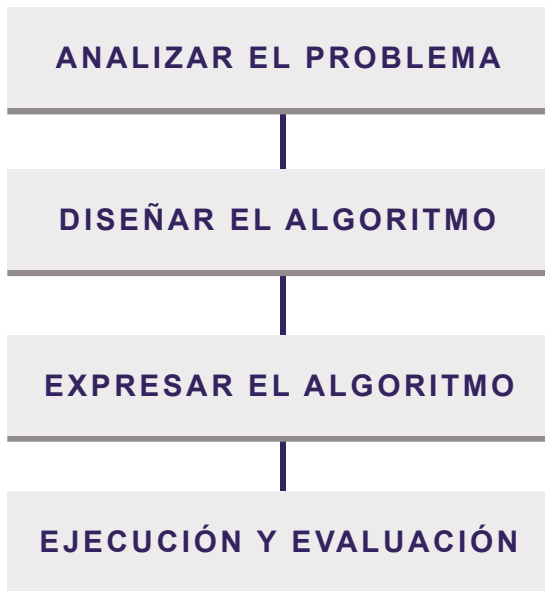
También existen grandes hallazgos como: operaciones con números decimales, Reglas paso a paso para la suma, Solución de ecuaciones entre otros.

Los algoritmos fueron creados con el fin de utilizarlos para resolver problemas, para luego llevarlos a un computador mediante un lenguaje de programación; para realizar un algoritmo se debe establecer una metodología de desarrollo, una de estas es la llamada **resolución de problemas**, la cual permite realizar un diseño organizado del algoritmo.

### 3. Resolución de problemas con algoritmos

Para solucionar un problema mediante un algoritmo es necesario seguir un orden, uno de los primeros pasos es el diseño previo de un algoritmo, con la resolución de problemas se puede seguir este orden de la siguiente manera:

- a. **Analizar el problema:** es analizar la situación que se está presentando y organizar en un orden lógico cada uno de los pasos para así resolver el problema.
- b. **Diseñar el algoritmo:** en este paso se describe la secuencia ordenada de pasos que conduce a la solución del problema citado (diagrama de flujo o pseudocódigo).
- c. **Expresar el algoritmo:** el algoritmo se debe expresar como un programa en un lenguaje de programación adecuado. (Fase de codificación.)
- d. **Ejecución y validación:** se pone en ejecución el programa realizado en un computador.



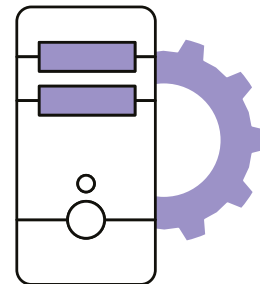
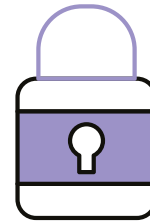
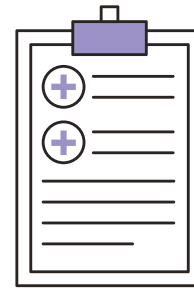
\*Pasos para resolver algoritmos.

#### 3.1 Análisis del problema

- **Reformular el problema:** si el problema que se ha planteado no se encuentra bien formulado, redactado o no se entiende al momento de leerse, debe reformularlo de tal manera que usted lo entienda y tenga claro la situación problema a solucionar.
- **Resultados Esperados:** debe especificar, describir y/o escribir los resultados que espera, por ejemplo (cuál es el producto final que se quiere tener para dar solución al problema, cual es la información a la que se necesita llegar, que se espera del problema citado)



- **Datos disponibles:** identificar la información disponible se resuelve haciéndose las siguientes preguntas: ¿qué información es importante o relevante para solucionar el problema? ¿cuáles son los datos de entrada?, ¿cuál es la incógnita?, ¿qué información me falta para resolver el problema?.
- **Restricciones:** determina las condiciones que plantea el problema para lograr el resultado, lo que está permitido, lo prohibido.
- **Procesos necesarios:** en esta fase debe definir los procesos para poder convertir la información disponible, en resultados esperados que den solución al problema ya que se determinan los procesos que se necesitan, las formulas a utilizar y el orden de lo que se debe realizar.



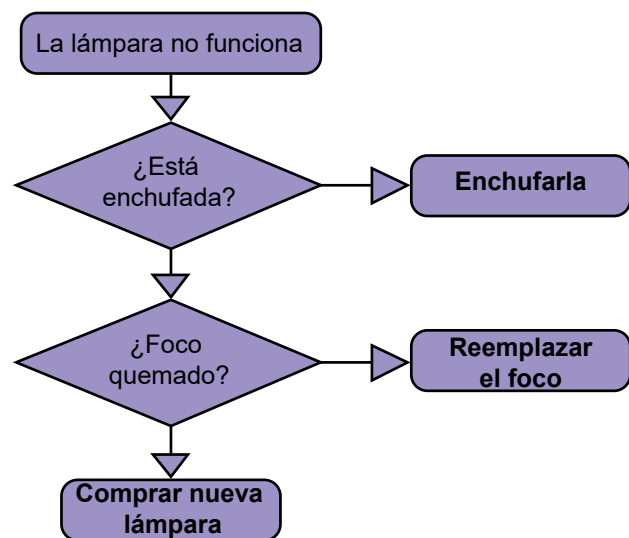
### 3.1.1 Análisis de Ejercicios

#### Ejemplo 1. La lámpara

Fuente: <https://goo.gl/images/pHAOQY>

Como se puede observar en el ejercicio anterior existe un análisis para determinar si la lámpara funciona o no funciona dependiendo la situación y así tomar una decisión con respecto al problema presentado.

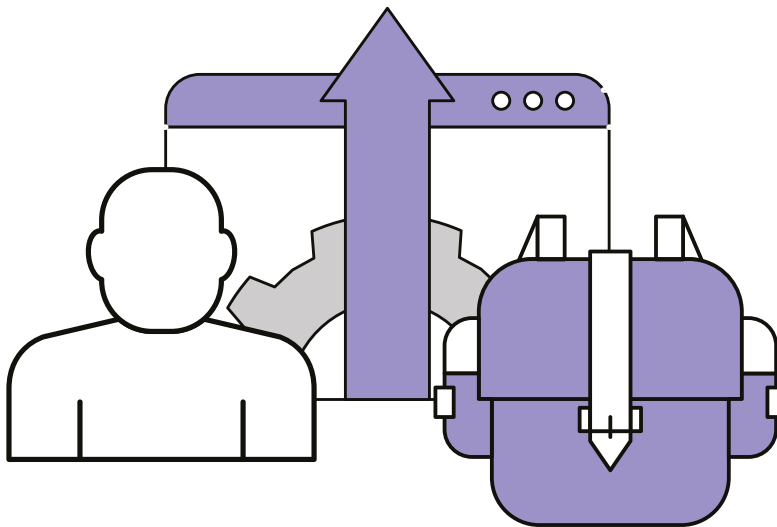
En esto es lo que básicamente consiste un algoritmo, en determinar un orden lógico y una descripción breve para lograr tomar una decisión o lograr un objetivo.



\*Simbología del algoritmo de la lámpara.



## Ejemplo 2



José David, es un muchacho que desea comprar una maleta de \$105.000. El recibe ingresos de diferentes fuentes: en la casa le dan \$6.000 para sus gastos semanales durante 4 semanas, por atender una tienda, tres veces recibió \$12.000. También su hermano lava la piscina una vez al mes por \$13.000 y cuida la tienda por \$10.000. ¿José David tiene ahorrado el dinero suficiente para comprar la patineta o aún le falta?

- **Formular el problema:** ya se encuentra claramente planteado, pero si no entiende la redacción puede redactarlo a su manera, hasta es posible realizar un resumen de lo planteado.
- **Resultados esperados:** saber si José David tiene o no tiene ahorrado el dinero para comprar su maleta, la cual cuesta \$105.000 pesos.
- **Datos disponibles:** los ingresos de José David \$6.000 pesos por 4 semanas + 12.000 pesos por 3, los datos irrelevantes serían: los \$13.000 y \$10.000 pesos que ganó el hermano ya que no aportan información para la solución de este problema y se pueden omitir.
- **Restricciones:** no se encuentra ninguna.
- **Procesos:** calcular el valor ahorrado por José David para saber si le alcanza para comprar la patineta.
- $\text{Valor Ahorrado} = 24.000 + 36.000 = 60.000$  Es decir no le alcanza para la maleta.

### Ejemplo 3

Se necesita calcular el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.

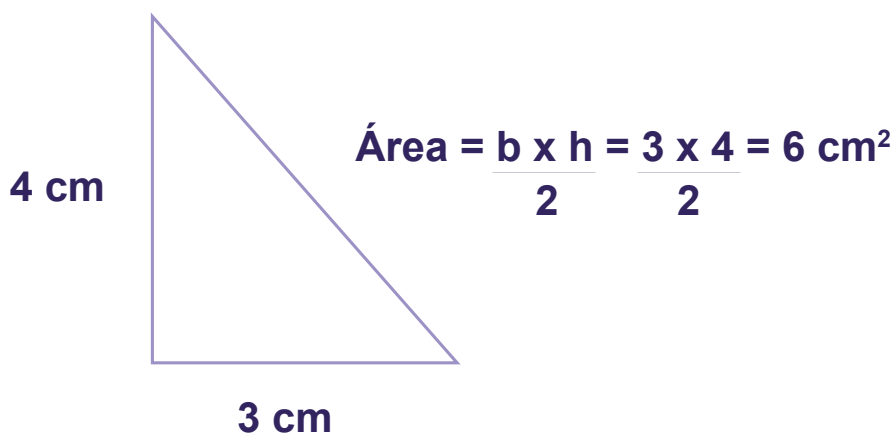
- Formular el problema: ya se encuentra claramente planteado, pero si no entiende la redacción puede redactarlo a su manera, hasta es posible realizar un resumen de lo planteado.
- Resultados esperados: el área de un triángulo rectángulo.
- Datos disponibles: Base, Altura, Hipotenusa, tipo de triángulo.

La incógnita es el área y todos los valores son constantes. El valor de la hipotenusa se puede omitir.

El aprendiz debe preguntarse si sus conocimientos actuales de matemáticas le permiten resolver este problema; de no ser así, debe plantear una estrategia para obtener los conocimientos requeridos.

Determinar las restricciones: utilizar las medidas dadas, y saber que se debe aplicar la fórmula del área de un triángulo rectángulo.

- Procesos necesarios: guardar en dos variables los valores de Base y Altura; Guardar en una constante el divisor 2; aplicar la fórmula  $\text{área} = \text{base} \times \text{altura} / 2$ ; comunicar el resultado (área).



**\*Cómo hallar el área del triángulo.**

Fuente: <https://goo.gl/images/nWcjTf>

### 3.1.2 Diseño de un algoritmo.

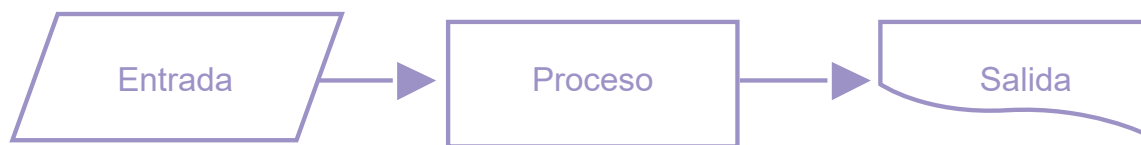
El diseño de un algoritmo se puede realizar mediante un diagrama de flujo o mediante pseudocódigo. Los algoritmos tienen las siguientes características.

### 3.1.3 Características de los algoritmos

Un algoritmo debe:

- Ser preciso e indicar el orden de realización de cada paso.
- Estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Ser finito. Si se sigue un algoritmo, se debe terminar en algún momento.

Un algoritmo debe contener como mínimo las siguientes partes:



**\*Partes de un algoritmo.**

Por ejemplo para realizar una receta de comida por medio de un algoritmo, cada parte podrá estar determinada así:

**Entrada:** insumos y elementos de trabajo, cocineros.

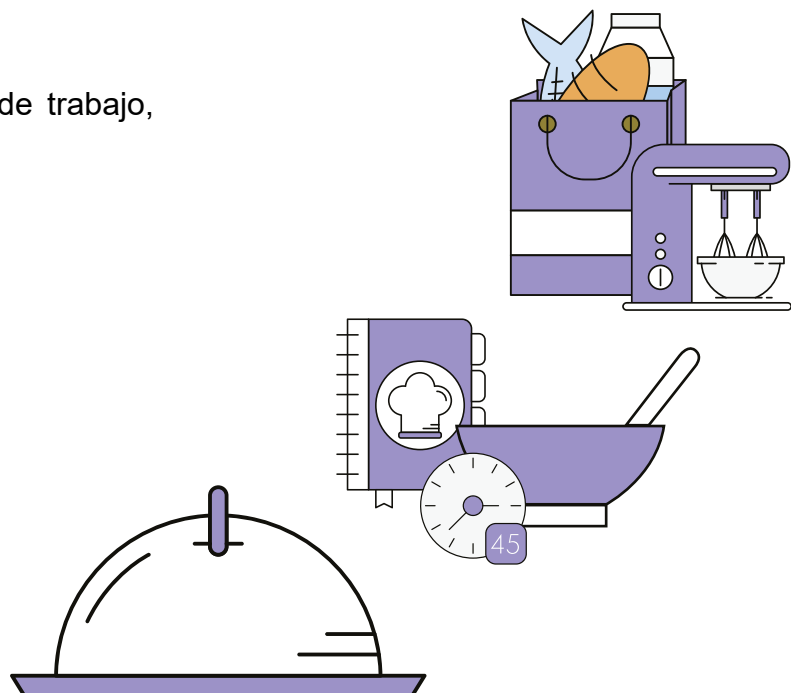
Entrada de un algoritmo.

**Proceso:** elaboración del plato

Proceso Algoritmico

**Salida:** preparación finalizada.

Salida del producto.



#### 4. Diseño de un algoritmo mediante diagrama de flujo

Los algoritmos pueden representarse de varias maneras, entre ellas está la representación de diagramas de flujo. La representación en diagramas de flujo tiene como objetivo seguir paso a paso la solución de un problema mediante símbolos.

##### 4.1 Definición Diagrama de flujo.

Un diagrama de flujo es un conjunto secuencial de figuras geométricas estándar conectadas lógicamente entre sí para dar solución a un problema específico, cada figura tiene un significado propio.

La secuencia lógica se da por medio de flechas llamadas **líneas de flujo** que indican el flujo lógico del algoritmo. Al ser un diagrama gráfico facilita la visión de la ejecución del algoritmo.

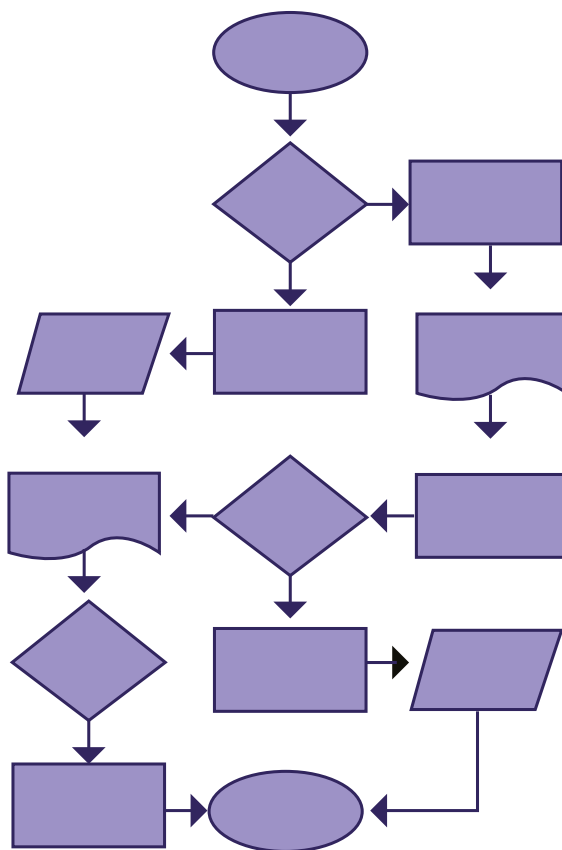
La simbología utilizada en estos diagramas ha sido estandarizada por las organizaciones ANSI (American National Institute) y por ISO (International Standard Organization).

SÍMBOLO	SIGNIFICADO	EXPLICACIÓN
	<b>Paso de tipo operación</b>	Representa cualquier tarea del proceso que lleve implícita una acción física o intelectual (excepto las de inspección o almacenaje).
	<b>Paso de inspección</b>	Se corresponde con tareas de verificación del trabajo realizado en determinada actividad del proceso. Sus acciones más comunes son: clasificar, observar, supervisar, auditar, probar, revisar, verificar, entre otras.
	<b>Paso de decisión</b>	Representa cualquier punto de decisión. Siempre tendrá al menos dos salidas.
	<b>Paso de almacenaje</b>	Se corresponde con una etapa del proceso que sitúa un producto, información o servicio en una zona de conservación (archivo, almacén o refrigerador) o posición (cola) para utilizarlo o proporcionar el servicio más adelante.
	<b>Paso de demora</b>	Corresponde a actividades que implican un retraso o pausa en el flujo del proceso.
	<b>Línea de flujo</b>	Muestra la dirección y sentido del flujo del proceso y representa el progreso de los pasos en la secuencia.
	<b>Documento</b>	Se utiliza con el objetivo de especificar los documentos confeccionados, corregidos o consultados en cada etapa.
	<b>Conector de tareas</b>	Se utiliza el caso de que el diagrama no se pueda hacer en una sola hoja.

Figura 10. Símbolos del Diagrama de Flujo.

Fuente: <https://goo.gl/images/gh7HKA>

Existen herramientas software que permiten realizar los gráficos mencionados anteriormente, una de las más utilizadas es **DFD**, día, entre otros, los cuales son útiles y de libre acceso en la web.



#### 4.2 Reglas para la elaboración de un diagrama de flujo:

- Los diagramas se deben realizar de arriba hacia abajo y de izquierda hacia derecha.
- Los símbolos de inicio y final deben aparecer solo una vez
- La ejecución de un programa siempre empieza en la parte superior del programa.
- La dirección del flujo se debe representar por medio de flechas.
- Todas las líneas de flujo deben llegar a un símbolo o a otra línea.
- Se deben inicializar las variables que se utilicen o permitir la asignación de valores mediante la consulta a un usuario.

### 4.2.1 Diseño de un algoritmo mediante pseudocódigo

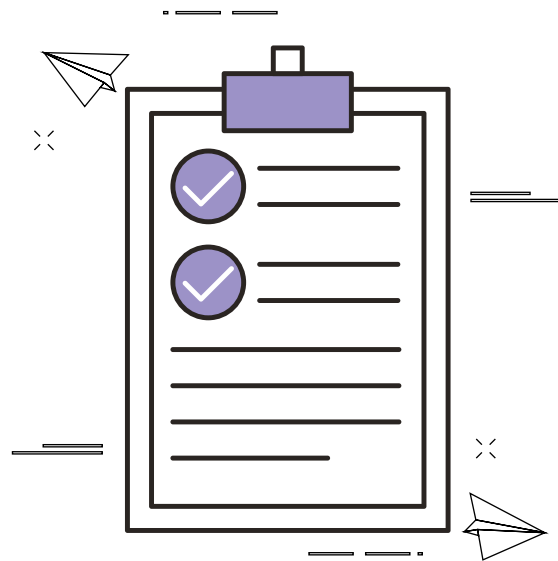
La representación de pseudocódigo sigue paso a paso la solución de un problema con lenguaje natural, pero recuerde que debe realizar la metodología resolución de problemas para poder iniciar su algoritmo con pseudocódigo. Podría recordarse la imagen de resolución de problemas explicada anteriormente.

**Paso 1:** inicio

**Paso 2:** los procesos que se van realizar....

**Paso 3:** si hay más pasos se debe continuar...

**Paso n:** fin



#### Ejemplo1:

Se retoma el ejemplo de José David para llevarlo en forma de algoritmo: José David, se encuentra ahorrando para comprar la maleta que vale 105.000 pesos. En su casa le han dado para sus gastos 24.000 pesos durante 4 semanas. Por atender la tienda recibió \$36.000 pesos. Su hermano Juan Antonio ganó 23.000 pesos por lavar la piscina y cuidar la tienda. ¿José David tiene ahorrado el dinero suficiente para comprar la maleta o aún le falta?

**Paso 1:** inicio

**Paso 2:** asignarle el valor del dinero ahorrado por José David a la Variable ValorAhorrado = 24.000+36.000

**Paso 3:** mostrar el ValorAhorrado, indicando si le alcanza o no para la maleta.

**Paso 4:** fin.

#### Ejemplo 2

Se retoma el ejemplo anterior también por continuidad.

Se necesita calcular el área de un triángulo rectángulo cuya Base mide 3 cm, la Altura 4 cm y la Hipotenusa 5 cm.

**Paso 1:** inicio

**Paso 2:** indicar que la variable Base tiene un valor de 3 cm

**Paso 3:** indicar que la variable Altura es de 4 cm

**Paso 4:** calcular el área  $a = (Bases * Altura) / 2$

**Paso 5:** mostrar el área

**Paso 6:** fin.

### Ejemplo 3

También se pueden resolver problemas cotidianos: por medio de pseudocódigo escriba un algoritmo para poder pasarse los semáforos, una vez usted se encuentra esperando pasar como peatonal.

**Paso 1:** inicio

**Paso 2:** ver el color del semáforo

**Paso 3:** si el semáforo esta en rojo : hay que detenerse

**Paso 4:** mostrar mensaje hay que detenerse.

**Paso 5:** si el semáforo esta en amarillo: alistarse para pasar

**Paso 6:** mostrar mensaje alistarse.

**Paso 7:** si el semáforo esta en verde: pasarse la calle.

**Paso 8:** mostrar mensaje puede pasarse y la persona se pasa la calle.

**Paso 9:** fin.

#### 4.2.2 Expresar el algoritmo en un lenguaje de programación

Una vez el algoritmo este diseñado en diagrama de flujo o en lenguaje natural (pseudocódigo), se pone en marcha la “traducción” de este a un lenguaje de programación específico.

Existen diferentes tipos de lenguajes que permiten “traducir” el algoritmo para que pueda ser entendido por el computador entre los cuales son java, c#, Visual Basic, entre otros, los cuales son tratados específicamente más adelante.



## 5. Solución de problemas con diagramas de flujo y lenguaje natural (Pseudocódigo).

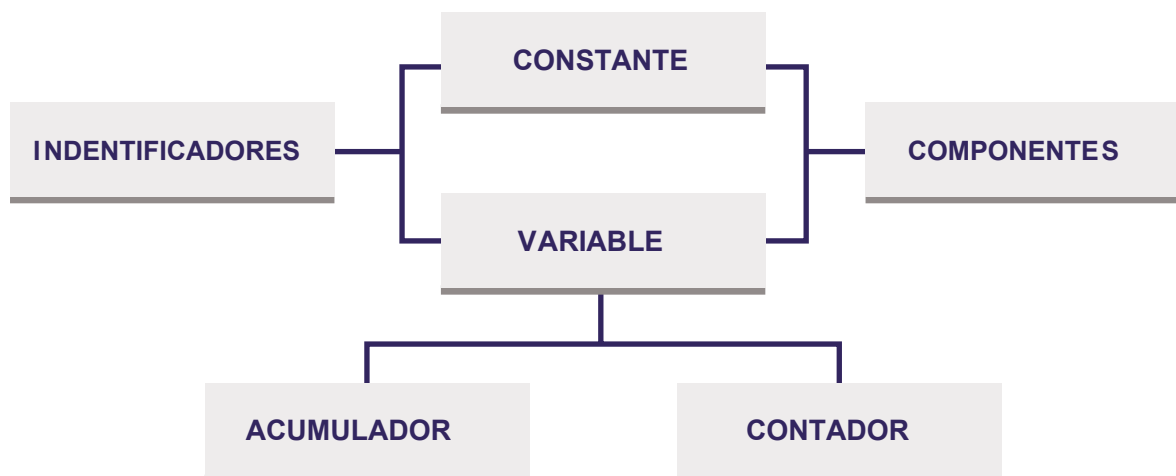


Figura 11. Estructura con identificadores.

Recuerde cuando se plantee un problema ya sabe que puede resolverlo por medio de diagrama de flujo o por lenguaje natural.

### 5.1 Variables.

Cuando se inicia con el análisis del problema se identifican los datos iniciales, estos datos se estructuran o se definen como variables; en ellas se pueden almacenar valores y son nombradas con **identificadores**, es decir nombres para poder identificarlas dentro del algoritmo. Por ejemplo, si en el problema de calcular el área de un triángulo tengo como datos iniciales la base y la altura, estas dos anteriores serían llamadas variables y sus nombres serán “BASE” y “ALTURA”.

Un aspecto importante de las variables es que pueden cambiar su valor durante la ejecución del algoritmo.

Se debe tener en cuenta que **una variable puede ser declarada, asignada o solicitada mediante un algoritmo para poder ser utilizada.**

#### 5.1.1 Declaración de una variable

Para utilizar una variable tanto en pseudocódigo como en diagrama de flujo es necesario siempre declararla, es decir indicarle al algoritmo que va a utilizar una variable por ejemplo Nom\_Persona, quiere decir que usted necesita una variable que almacene los nombres de las personas. Ejemplo Nom\_Persona=Nombre de las personas.



### 5.1.2 Asignación de valor a una variable

Luego de declararla puede usted necesitar que esa variable inicie por defecto con un valor por ejemplo Saldo = 60000, quiere decir que utilizará la variable saldo pero que esta inicializada con un valor de 60000, puede inicializar una variable según el valor que necesite para solucionar el problema.

Asignarle un valor a una variable, constante, acumulador o contador, expresiones complejas o simples, por ejemplo.

Variable = expresión

Nom\_Persona = Rita

Saldo = 4000

Area = (Base\*altura)

### 5.1.3 Solicitar una variable

Cuando se necesita que el valor de la variable pueda ser ingresado por un usuario al computador, entonces debe solicitarse este valor de la variable.

## 5.2 Constantes

Almacenan datos al igual que las variables, pero su gran particularidad es que sus datos no cambian durante la ejecución del algoritmo, es decir siempre el valor de la constante va ser el mismo.

Las constantes se deben declarar e inicializar.



DIAGRAMA DE FLUJO	PSEUDOCÓDIGO
<b>Declarar una constante e inicializarla</b> <div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: fit-content;"> Máximo            100 </div>	<b>Declarar una constante e inicializarla</b>  <b>Constante Máximo 100</b>

Ejemplo de constante.

## 5.3 Contadores

Como su palabra lo dice permiten contar, para poder utilizar un contador es necesario inicializarlo en un valor y luego incrementar su valor de una manera constante para permitir realizar el conteo.

Es una variable cuyo valor se incrementa o decrementa en una cantidad constante cada vez que se produce un determinado suceso o acción. Los contadores se utilizan con la finalidad de contar sucesos o acciones internas de un bucle; deben realizar una operación de inicialización y posteriormente las sucesivas de incremento o decremento del mismo. La inicialización consiste en asignarle al contador un valor. Se situará antes y fuera del bucle.

DIAGRAMA DE FLUJO	PSEUDOCÓDIGO
<b>Declarar una constante e inicializarla</b> 	<b>Declarar una constante e inicializarla</b> <b>cont = 0</b>
<b>Utilizar contador</b> 	<b>Utilizar un contador</b> <b>cont = cont + 1</b>

Representación:

<nombre del contador> = <nombre del contador> + <valor constante>

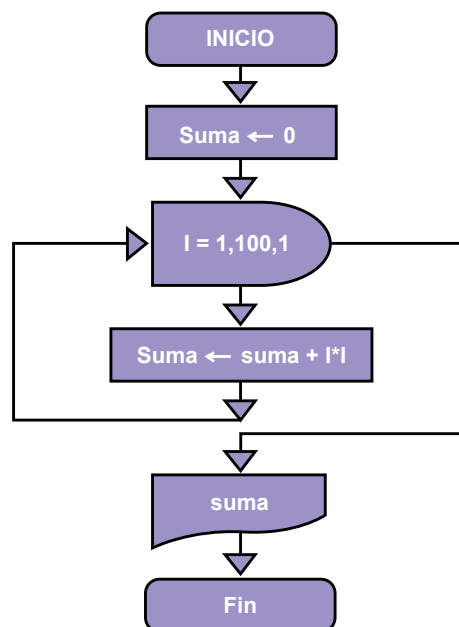
Si en vez de incremento es decremento se coloca un menos en lugar del más.

Ejemplo:  $i = i + 1$

## 5.4 Acumuladores

Como su palabra lo dice permiten acumular el valor de una variable, para poder utilizar un acumulador es necesario inicializarlo en un valor y luego iniciar con la acumulación del valor.

El anterior algoritmo está representado mediante un Diagrama de Flujo y lo que hace es inicializar un valor en 1 hasta 100, hasta imprimir la suma de cada dígito acumulándolo hasta 100. En este caso sería  $1+2+3+4+5+...+100$ .

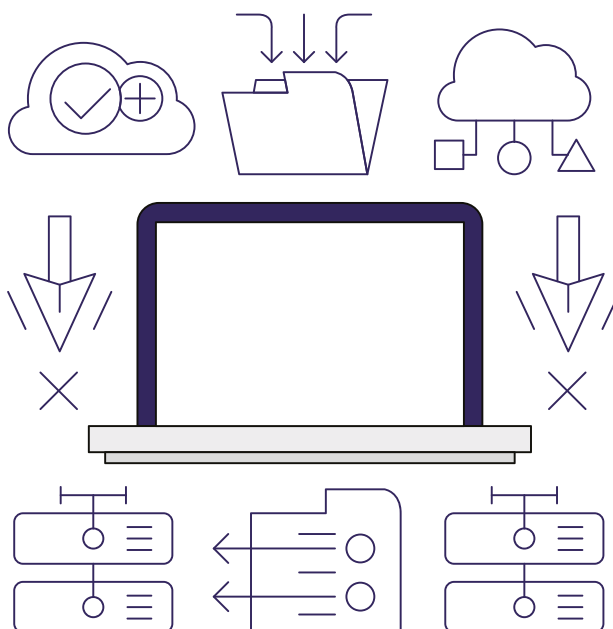


## 5.5 Identificadores

Los identificadores son nombres que se dan a las variables, constantes, acumuladores y contadores para así poder diferenciarlos. Para asignar los nombres se debe tener en cuenta lo siguiente:

- Los nombres pueden estar formados por una combinación de letras y números (saldoMes, salario, fecha2, baseTriángulo, etc).
- El primer carácter de un nombre debe ser una letra.
- La mayoría de los lenguajes de programación diferencian las mayúsculas de las minúsculas.
- Los nombres deben ser nemotécnicos, con solo leerlos se puede entender lo que contienen. Deben ser muy descriptivos; no utilizar abreviaturas, a menos que se justifique plenamente.
- No utilizar caracteres reservados (% , + , / , > , etc).
- No utilizar palabras reservadas por los lenguajes de programación.
- Para cumplir con convenciones ampliamente utilizadas (Jiménez, 2002), los nombres de procedimientos, variables y constantes deben empezar con minúscula. Ejemplo, fecha, suma, etc. Si es un nombre compuesto por varias palabras, cada una de las palabras (con excepción de la primera) debe empezar con mayúscula. Ejemplo: fechaInicial, baseTriángulo, etc.

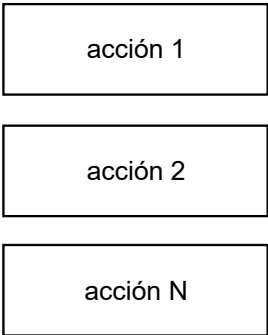
## 6. Estructuras algorítmicas o de programación



Las estructuras algorítmicas están creadas para orientarnos en la forma de diseñar algoritmos, cada una de ellas representa cierto concepto que permite lograr encontrar la solución del problema, entre ellas tenemos:

## 6.1 Estructura secuencial

Es una de las estructuras más sencillas, conocida también como estructura lineal y se compone de instrucciones que deben ejecutarse secuencialmente. Ejemplo:

DIAGRAMA DE FLUJO	PSEUDOCÓDIGO
 <pre> graph TD     A[acción 1] --&gt; B[acción 2]     B --&gt; C[acción N]         </pre>	<p><b>Inicio</b></p> <p>&lt;acción1&gt;</p> <p>&lt;acción2&gt;</p> <p>*</p> <p>*</p> <p>*</p> <p>&lt;acciónN&gt;</p> <p><b>Fin</b></p>

## 6.2 Correspondencia de pseudocódigo a diagrama de flujo.

Consiste en llevar un algoritmo a un diagrama de flujo.

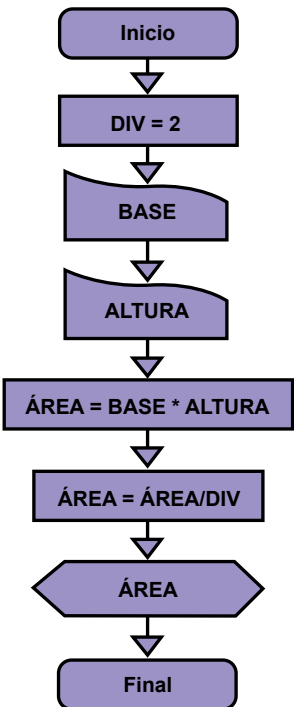
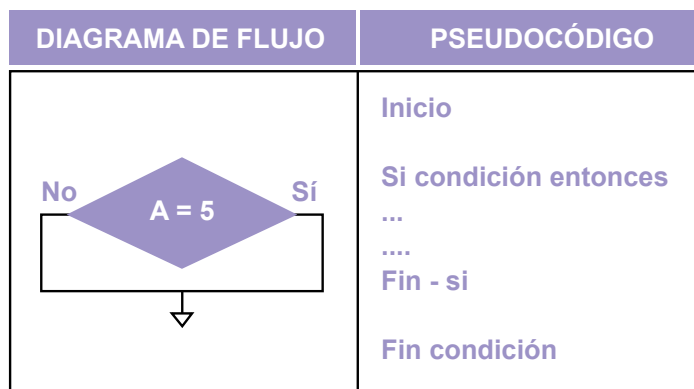
DIAGRAMA DE FLUJO	PSEUDOCÓDIGO
 <pre> graph TD     Inicio([Inicio]) --&gt; DIV[DIV = 2]     DIV --&gt; BASE[/BASE/]     BASE --&gt; ALTURA[/ALTURA/]     ALTURA --&gt; AREA_CALC[ÁREA = BASE * ALTURA]     AREA_CALC --&gt; AREA_DIV[ÁREA = ÁREA / DIV]     AREA_DIV --&gt; AREA_OUT{ÁREA}     AREA_OUT --&gt; Final([Final])         </pre>	<p><b>Paso 1:</b> inicio.</p> <p><b>Paso 2:</b> asignar el número 2 a la constante DIV.</p> <p><b>Paso 3:</b> saber la base del triángulo y guardarlo en la variable "BASE".</p> <p><b>Paso 4:</b> saber la altura del triángulo y guardarla en la variable "ALTURA".</p> <p><b>Paso 5:</b> guardar en la variable "AREA" el valor de <math>BASE * ALTURA</math></p> <p><b>Paso 6:</b> guardar en la variable "AREA" la división entre el "AREA" sobre la variable "DIV".</p> <p><b>Paso 7:</b> mostrar el valor de área.</p> <p><b>Paso 8:</b> fin.</p>

Figura 14. Diagrama de Flujo con Pseudocódigo.

### 6.3 Estructuras de decisión

Son utilizadas para tomar decisiones lógicas, llamadas también estructuras selectivas o alternativas, en ellas se evalúa una condición y en función del resultado de la misma se realiza una opción u otra.

**Ejemplo:**



Las estructuras de decisión pueden ser:

#### 6.3.1 Simples.

Ejecuta una determinada acción cuando se cumple una determinada condición (llamada si-entonces). La selección si-entonces evalúa la condición y si la condición es verdadera, entonces ejecuta la acción; si la condición es falsa entonces no hace nada. Ejemplo:

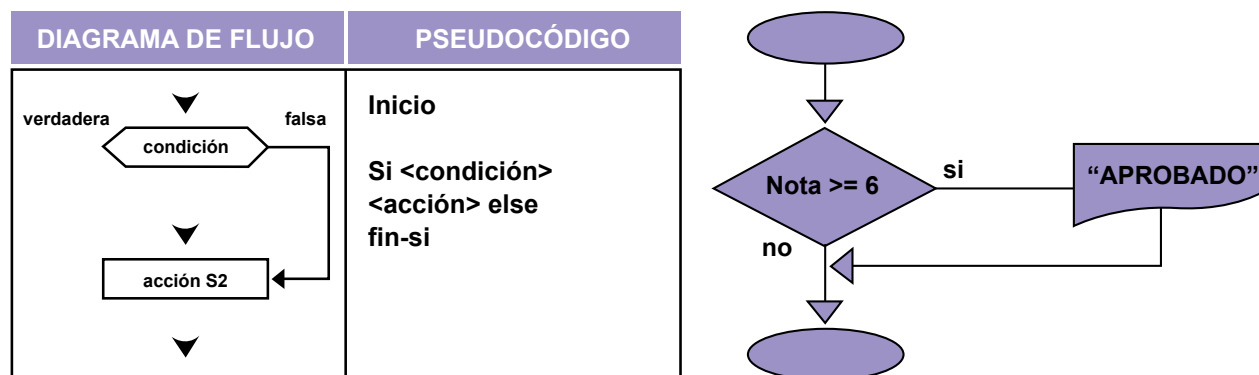
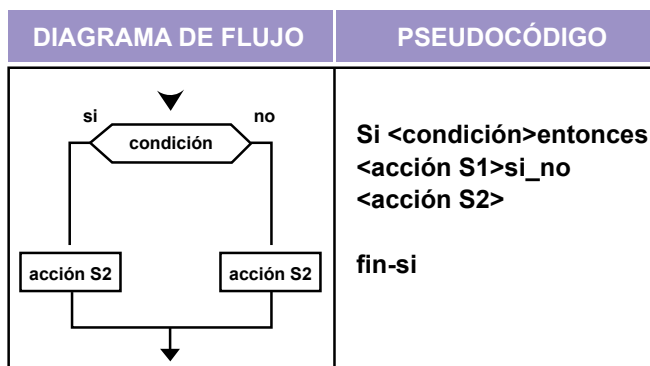


Figura 15. Estructura simple de un condicional.

En el ejemplo anterior existe una estructura de decisión simple dado que si la Nota es superior a 6 es considerado el aprendiz como un logro aprobado.

### 6.3.2 Dobles:

La estructura anterior es muy limitada y normalmente se necesitará una estructura que permita elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición. **Ejemplo:**



En el siguiente ejemplo dados 3 números definir cuál de ellos es mayor e imprimirlo. (Véase Figura 16).

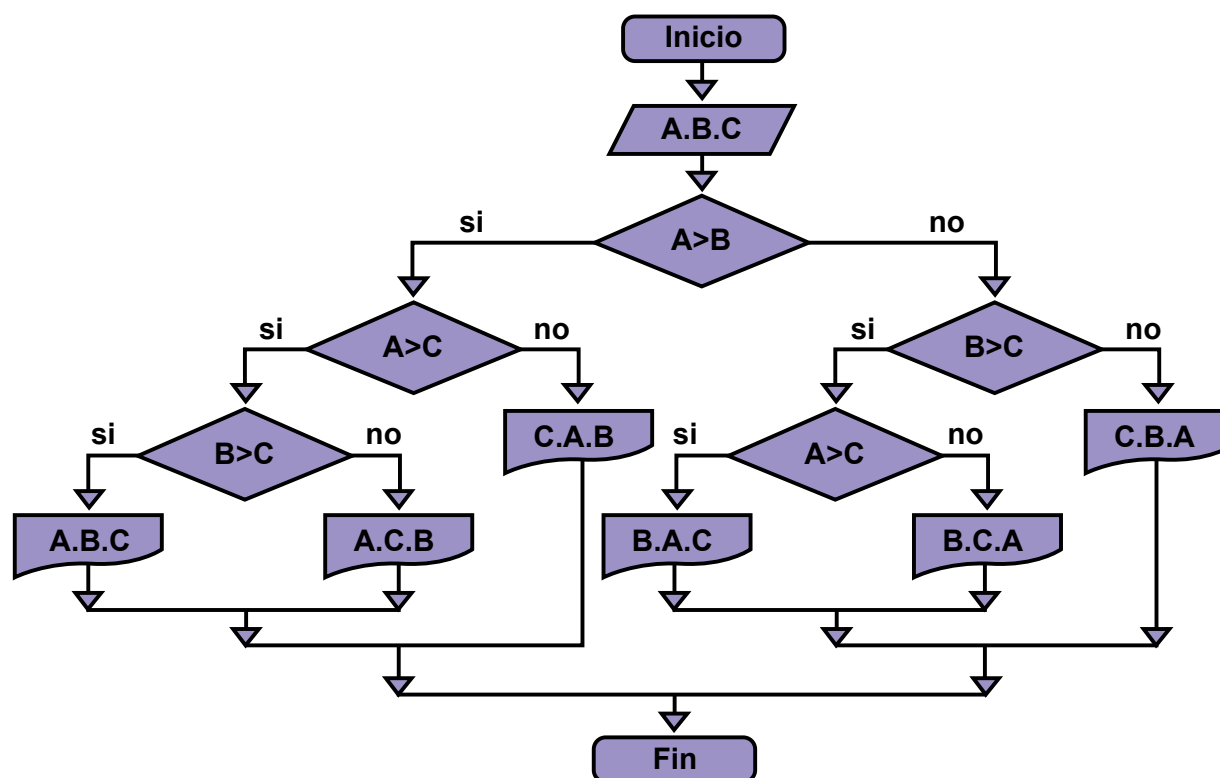


Figura 16. Estructura condicional anidada.

Fuente: <https://goo.gl/images/4J38cR>

## 6.4 Estructuras cíclicas y/o repetitivas

Un tipo muy importante de estructura es el algoritmo necesario para repetir una o varias acciones un número determinado. Las estructuras que repiten una secuencia de instrucciones se denominan bucles, y se llama iteración al hecho de repetir la ejecución de una secuencia de acciones.

Y así sucesivamente para cada número de la lista. En otras palabras, el algoritmo repite muchas veces las acciones.

Las dos principales preguntas a realizarse en el diseño de un bucle son: ¿qué contiene el bucle? y ¿cuántas veces se debe repetir?.

### 6.4.1 Estructura para

En muchas ocasiones se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle. Esta estructura permite ejecutar una o varias instrucciones un determinado de veces finita, fija. (Véase Figura 17).

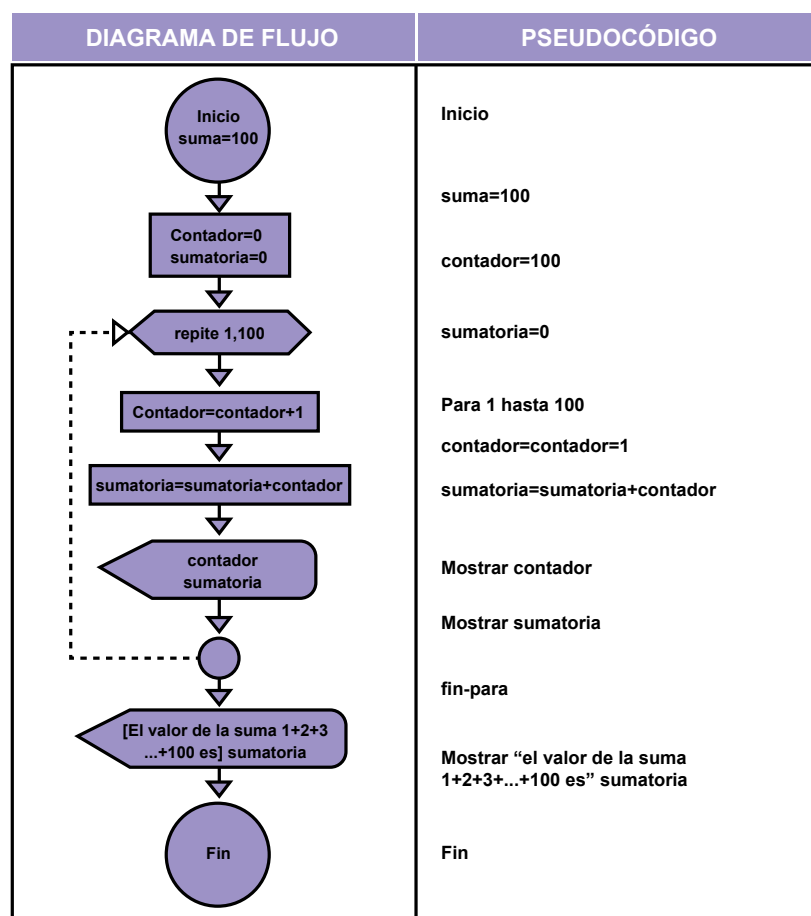


Figura 17. Diagrama con estructura cíclica y/o repetitiva

## 6.4.2 Estructura mientras

Se llama Mientras a la estructura algorítmica que se ejecuta mientras la condición evaluada resulte verdadera. Se evalúa la expresión booleana y, si es cierta, se ejecuta la instrucción especificada, llamada el cuerpo del bucle. Entonces se vuelve a evaluar la expresión booleana, y si todavía es cierta se ejecuta de nuevo el cuerpo. Este proceso de evaluación de la expresión booleana y ejecución del cuerpo se repite mientras la expresión sea cierta.

Cuando se ejecuta la instrucción mientras, la primera cosa que sucede es que se evalúa la condición (una expresión booleana). Si se evalúa falsa, ninguna acción se toma y el programa prosigue en la siguiente instrucción del bucle. Si la expresión booleana es verdadera, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la expresión booleana. Este proceso se repite una y otra vez mientras la expresión booleana (condición) sea verdadera. (Véase Figura 18).

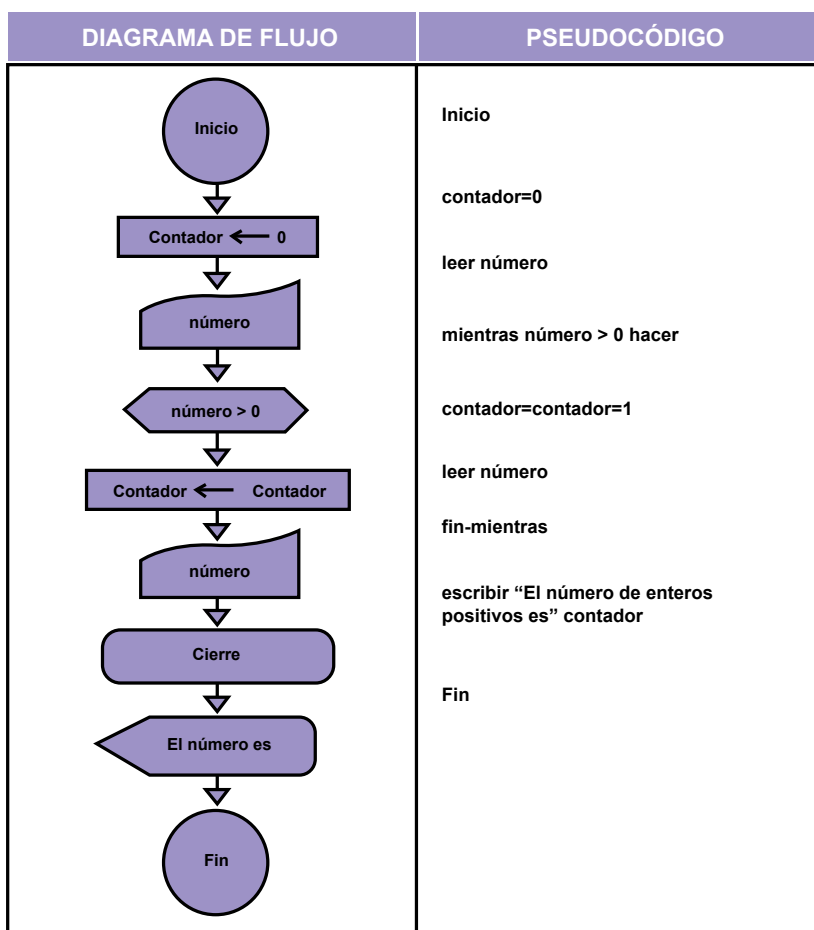
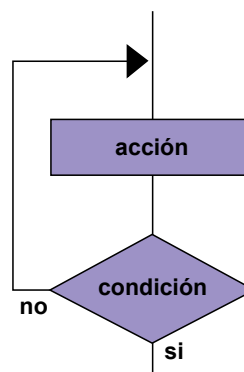


Figura 18. Estructura con la función Mientras.

## 6.4.3 Estructura Repita

La estructura cíclica REPITA, al igual que la estructura cíclica mientras, se ejecuta un número indeterminado de veces, estas dos estructuras tienen un comportamiento similar, presentando su principal diferencia en el lugar de estructura donde se evalúa la condición, dado que la estructura MIENTRAS evalúa la condición del ciclo al inicio del mismo y la estructura REPITA lo hace al final del mismo, de este modo, en la estructura cíclica REPITA, el programador garantiza que el ciclo se ejecuta al menos una vez. (Véase Figura 19).





## Glosario

**Algoritmo:** es un conjunto de instrucciones que conducen a la solución de un problema determinado, las cuales deben estar relacionadas lógicamente y ordenadamente.

**ANSI:** (American National Standards Institute): el Instituto Nacional Estadounidense de Estándares, es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos.

**ISO:** (International Organization for Standardization): la Organización Internacional para la Estandarización, es una federación mundial que agrupa a representantes de cada uno de los organismos nacionales de estandarización (como lo es el ICONTEC en Colombia), y que tiene como objeto desarrollar estándares internacionales que faciliten el comercio internacional entre otros.

**Pseudocódigo:** es una descripción informal de alto nivel de un algoritmo, que utiliza las convenciones estructurales de un lenguaje de programación verdadero, pero que está diseñado para la lectura.

## Bibliografía

Joyanes, L. (2003). *Fundamentos de programación*. Madrid: McGraw-Hill.

Lopez, J. (2007, 2009). *Algoritmos y programación*. Bogotá: Fundación Gabriel Piedrahita Uribe.

Servicio Nacional de Aprendizaje, SENA. (2009, Mayo). *Diseño Interfaz de usuario*. Curso virtual Metodologías de análisis y diseño de Sistemas.

Sommerville, I. (2005). *Ingeniería del Software*. Madrid: Pearson Educación.

## Control del documento

### CONSTRUCCIÓN OBJETO DE APRENDIZAJE



### INTRODUCCIÓN Y CONSTRUCCIÓN DE ALGORITMOS

Centro Industrial de Mantenimiento Integral - CIMI  
Regional Santander

**Líder línea de producción:** Santiago Lozada Garcés

**Asesores pedagógicos:** Rosa Elvia Quintero Guasca  
Claudia Milena Hernández Naranjo

**Líder expertos temáticos:** Rita Rubiela Rincón Badillo

**Experto temático:** Ana Yaqueline Chavarro Parra - V1

**Experto temático:** Rita Rubiela Rincón Badillo - V2

**Diseño multimedia:** Jesús Antonio Vecino Valero

**Programador:** Francisco José Lizcano Reyes

**Producción de audio:** Víctor Hugo Tabares Carreño

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de la licencia que el trabajo original.

© Microsoft Visual Studio. 2017. Todos los derechos reservados.

© Java by Oracle®. 2017. Todos los derechos reservados.



**creative  
commons**

BY NC SA



**Registered trademark**