



UNIVERSIDAD DE GUADALAJARA

Red Universitaria de Jalisco

# Diagramas de Flujo

Prof. Sabrina Lizbeth Vega Maldonado

Guadalajara, Jalisco, México

Julio de 2013

# Diagramas de flujo

Imagina que en tu trabajo te encargan realizar un diagrama de flujo que represente a una calculadora. ¡Pero no tienes ni idea de lo que es un diagrama de flujo! ¿Qué hacer entonces?

## ¿Qué son los diagramas de flujo?

Los algoritmos son los pasos a seguir para resolver un problema. "Los diagramas de flujo representan la esquematización gráfica de un algoritmo" (Cairó, 2006). Es decir, un algoritmo puede ser descrito en un diagrama de flujo. Esto permite que se delimita una manera estándar de describir algoritmos, en especial los computables, acercándonos a su representación en un lenguaje de programación para que una computadora resuelva el problema por nosotros.

Hay unas cuantas reglas para construir diagramas de flujo, recomendados por Cairó:

- El diagrama de flujo debe siempre tener un inicio y un fin.
- Las líneas que indican la dirección del flujo en el diagrama deben ser rectas.
- Las líneas siempre deben estar conectando a otros elementos del diagrama.
- La construcción del diagrama siempre debe hacerse de arriba hacia abajo y de izquierda a derecha.

Adicionalmente, se pueden mencionar estas reglas:

- No pueden haber elementos aislados, todo tiene una entrada y una salida (exceptuando el inicio y el fin). Todo debe estar conectado con algo a través de una línea.
- En las entradas y las salidas, se pueden agrupar los pasos.

## Sección 1

### Los cinco símbolos básicos

En esta sección se muestran los cinco símbolos básicos de los diagramas de flujo y su función:

#### Símbolo



#### Función

El **rectángulo** representa un proceso o un paso en el algoritmo. El contenido (el texto que describe lo que ocurre en el proceso) se pone en el centro del rectángulo.



El **rombo** representa una prueba lógica, una valoración o comparación. Combinado y con algunas variaciones, este símbolo se utiliza para diseñar decisiones y repeticiones. El contenido (el texto que describe la prueba lógica) se pone en el centro del rombo.



El **óvalo** representa el inicio y el fin de un diagrama de flujo. La palabra Inicio o Fin se pone en el centro del rombo.



El **romboide** representa la introducción o entrada de valores y datos. El nombre de la variable o dato que se introducirá se pone en el centro del romboide.



Este símbolo, denominado **símbolo de salida**, representa la escritura o salida de valores y datos. El nombre de la variable o dato que se mostrará se pone en el centro del símbolo.

Un símbolo más que se puede mencionar es la flecha o línea. Este símbolo se encarga de unir a los otros símbolos para que, en la posición correcta, representen gráficamente a un algoritmo. Podrás ver ejemplos del uso de éstos símbolos en la sección sobre cómo unir estos símbolos.

## Sección 2

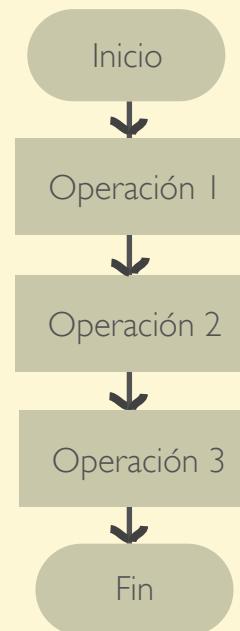
### Cómo unir los símbolos para crear secuencias y decisiones sencillas

Para hacer estructuras secuenciales basta con poner un símbolo tras otro, unidos por líneas. Observa a continuación un ejemplo genérico de esto:

#### Algoritmo genérico

1. Operación 1
2. Operación 2
3. Operación 3

#### Diagrama de flujo genérico



Observa que en el diagrama de flujo, se han agregado los símbolos de Inicio y Fin.

Ahora veamos la estructura de la decisión sencilla. La decisión sencilla es una bifurcación que responde a una condición lógica. Esta bifurcación hace que una operación se lleve a cabo o no dependiendo del resultado lógico de la condición (cierto o falso). A veces, será necesario "hacer una operación si la condición es verdadera" pero también es posible "hacer la operación 1 si la condición es verdadera pero si es falsa, hacer la operación 2". Debido a esto, es posible diagramas dos tipos de decisiones sencillas:

Tipo	Algoritmo genérico	Diagrama de flujo genérico
Decisión simple cierta	Si la condición es cierta entonces realizar la operación 1.	<pre> graph TD     Inicio([Inicio]) --&gt; Condicion{Condición}     Condicion -- Sí --&gt; Operacion[Operación]     Operacion --&gt; Condicion     Condicion -- No --&gt; Fin([Fin])   </pre>
Decisión simple cierta y falsa	Si la condición es cierta entonces realizar la operación 1, de otro modo, realizar la operación 2.	<pre> graph TD     Inicio([Inicio]) --&gt; Condicion{Condición}     Condicion -- Sí --&gt; OperacionI[Operación 1]     OperacionI --&gt; Condicion     Condicion -- No --&gt; OperacionII[Operación 2]     OperacionII --&gt; Fin([Fin])   </pre>



Observa que se agregó de nuevo el Inicio y el Fin, y que en el diagrama de flujo **NO DEBE PONERSE** si la condición es cierta o falsa dentro del rombo, esto es debido a que las líneas que salen del rombo deben estar etiquetadas con las palabras **Sí** y **No**, describiendo hacia dónde va el flujo del diagrama, dependiendo del resultado de la condición lógica.

Para que queden más claras las estructuras de decisión sencilla, se mostrarán algunos ejemplos.

## Ejemplo 1

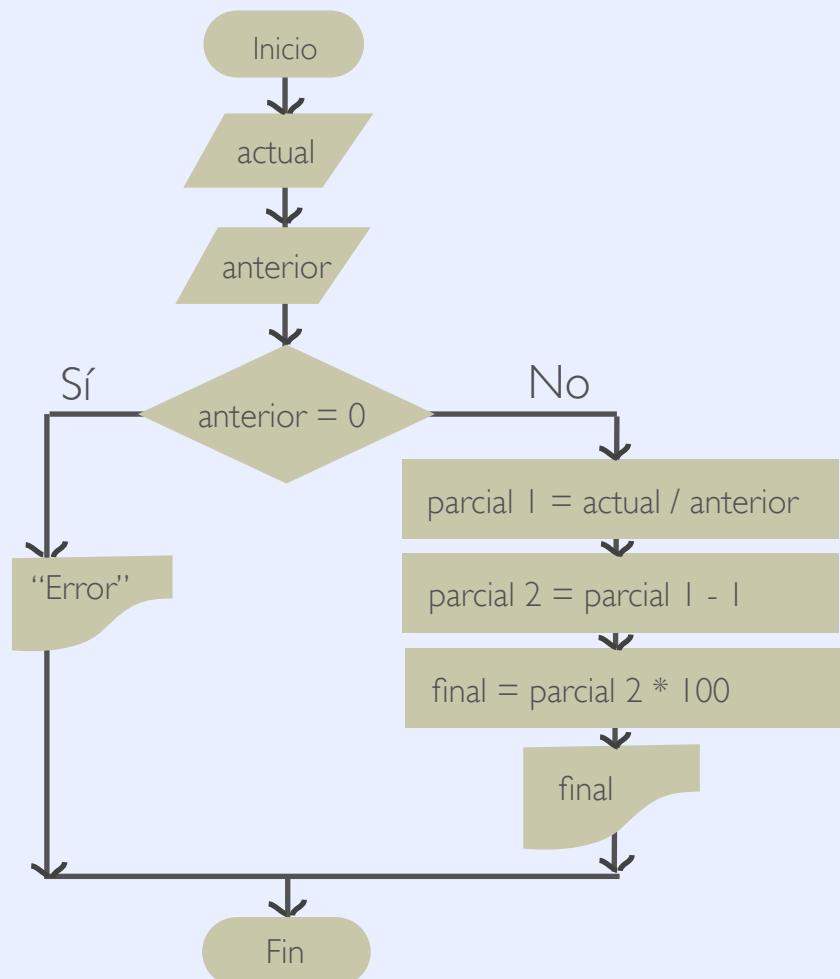
El primer ejemplo es sobre el problema de cómo calcular la variación porcentual de una cifra con respecto a otra, es decir, en qué porcentaje crece o decrece una cifra actual con respecto a una anterior.

Por ejemplo, la variación porcentual de 5 (anterior) a 10 (actual) es del 100% (5 creció en 100% para llegar a 10). Para calcular la variación porcentual se usa la fórmula: a la división del valor actual entre el valor anterior se le resta uno, a ese resultado se le multiplica por cien; siendo válido sólo si el valor anterior es diferente de cero. Entonces, se tiene:

### Algoritmo

1. Obtener el valor actual.
2. Obtener el valor anterior.
3. Dado que se debe hacer una división entre el valor anterior, el valor anterior no debe ser cero, si el valor anterior es cero, se debe mostrar que hay un error y terminar, de otra forma, si el valor anterior no es cero, entonces se debe dividir el valor actual entre el valor anterior, esto es el resultado parcial 1.
4. Al resultado parcial 1 se le debe restar 1, esto es el resultado parcial 2.
5. Al resultado parcial 2 se le debe multiplicar por cien, esto es el resultado final.
6. Se debe mostrar el resultado final.

### Diagrama de flujo



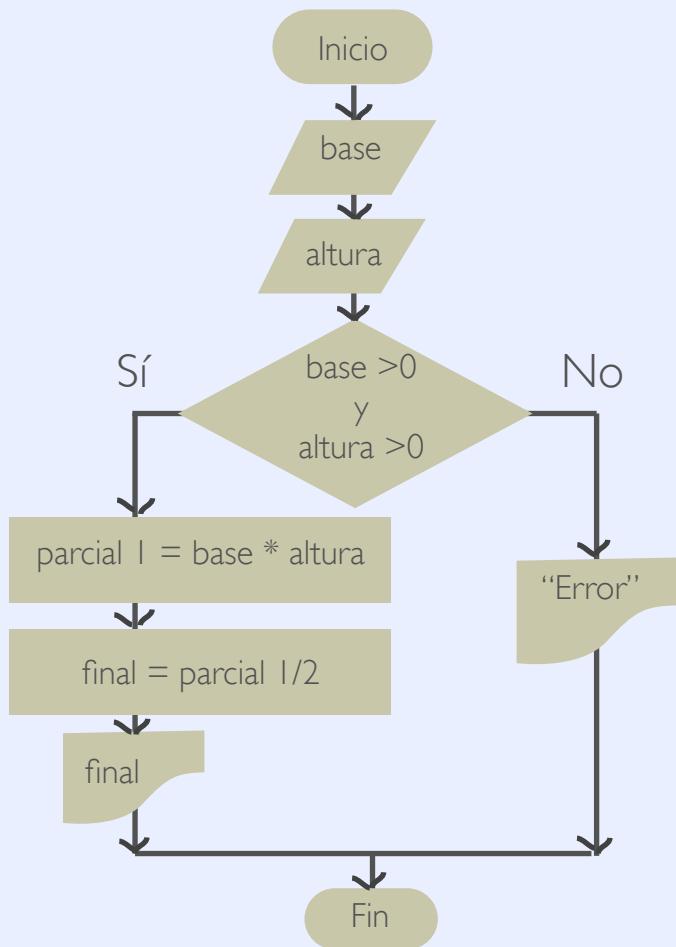
## Ejemplo 2

Este es el ejemplo del cálculo del área de un triángulo. Para calcular el área de esta figura, basta con multiplicar la medida de la base por la altura y a dicho resultado, dividirlo entre dos, siendo todo esto válido sólo cuando la base y la altura son mayores a cero:

### Algoritmo

1. Obtener el valor de la base.
2. Obtener el valor de la altura.
3. Dado que sólo es válido que la base y la altura sean mayores a cero, si ambos valores son cero entonces se debe mostrar que hay un error y terminar, de otra forma, si la base y la altura son mayores a cero, entonces hay que multiplicar la base por la altura y esto es el resultado parcial 1.
4. Hay que dividir el resultado parcial 1 entre 2, esto es el resultado final.
5. Se debe mostrar el resultado final.

### Diagrama de flujo



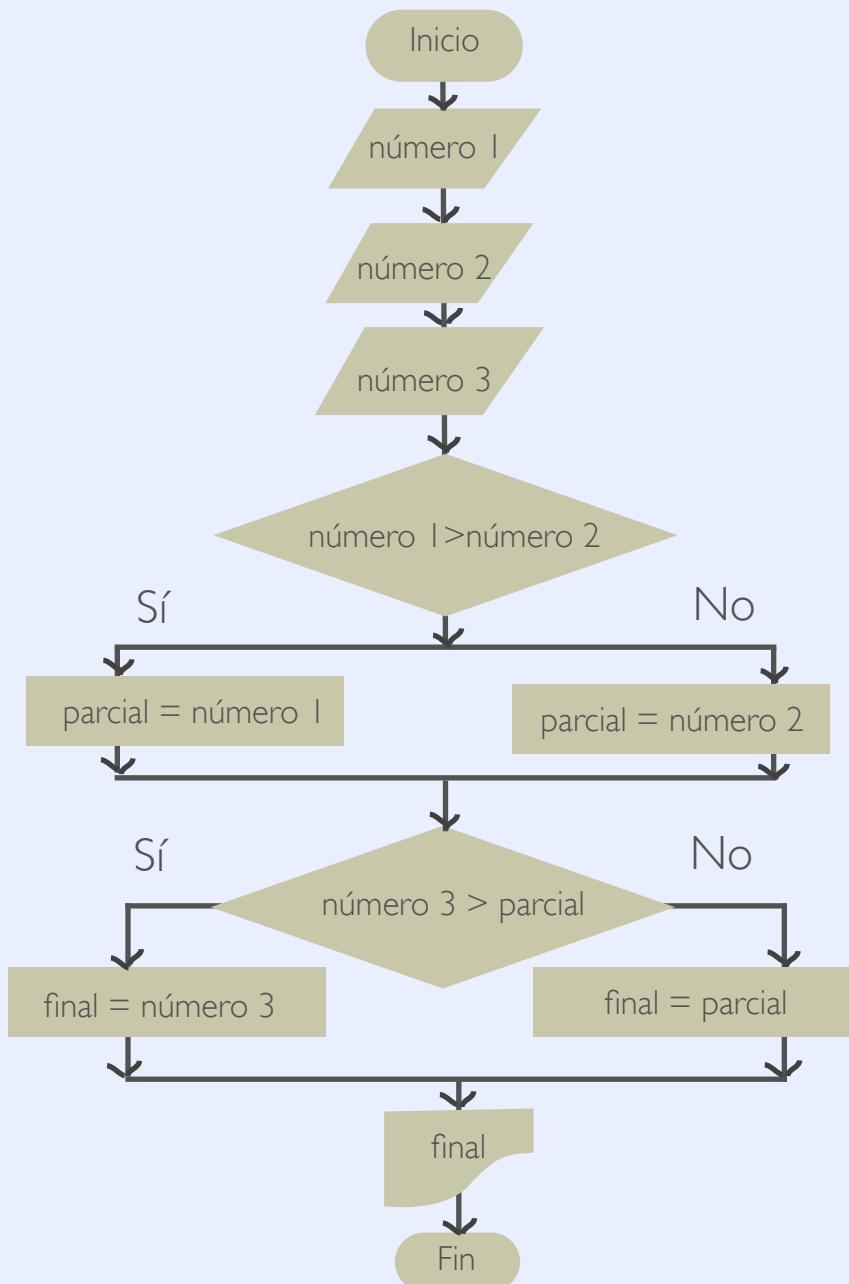
### Ejemplo 3

Ahora se tiene el siguiente problema: Dados tres números, ¿cuál es el mayor?

#### Algoritmo

1. Obtener el primer número.
2. Obtener el segundo número.
3. Obtener el tercer número.
4. Si el primer número es mayor que el segundo número, entonces el número mayor es el primero, de lo contrario, el número mayor es el segundo. Esto es el resultado parcial.
5. Si el tercer número es mayor al resultado parcial, entonces el número mayor es el tercer número, de lo contrario, el número mayor es el resultado parcial. Esto es el resultado final.
6. Mostrar el resultado final.

#### Diagrama de flujo



Puedes observar que el algoritmo no corresponde exactamente al diagrama de flujo, sobretodo en las condiciones "si pasa esto, entonces esto" (los rombos). Pero también observa que el diagrama de flujo debe poder interpretarse como si fuera el algoritmo mismo.

## Sección 3

### Símbolos avanzados

Hay algunos símbolos para funciones más avanzadas. A continuación se muestran esos símbolos junto con su función:

Símbolo	Función
	El <b>hexágono</b> representa una selección múltiple. El nombre de la variable o dato que se comparará con las posibles opciones de la selección múltiple se pone en el centro del hexágono.
	El <b>círculo</b> sirve para conectar elementos dentro de una página, sirve especialmente cuando un diagrama se vuelve complejo. Suele ponérsele una letra o un número en el centro, para identificarlo.
	El <b>pentágono</b> que apunta hacia abajo sirve para conectar elementos en páginas diferentes, sirve especialmente cuando un diagrama se vuelve demasiado complejo y es mejor mostrarlo en páginas diferentes.
	El <b>pentágono</b> que apunta hacia la derecha para expresar la modularidad en un diagrama, es decir, algo que debe ser resuelto antes de continuar con el flujo normal del diagrama.

## Sección 4

### Cómo unir los símbolos para crear decisiones avanzadas, repeticiones y diagramas complejos

Para hacer diagramas más complejos, decisiones avanzadas, repeticiones y construir diagramas complejos, de pueden usar los símbolos que viste en la Sección 3. También se te mostrarán ejemplos que muestran cómo usar y conjugar los símbolos básicos y los avanzados para lograr diseñar diagramas más avanzados.

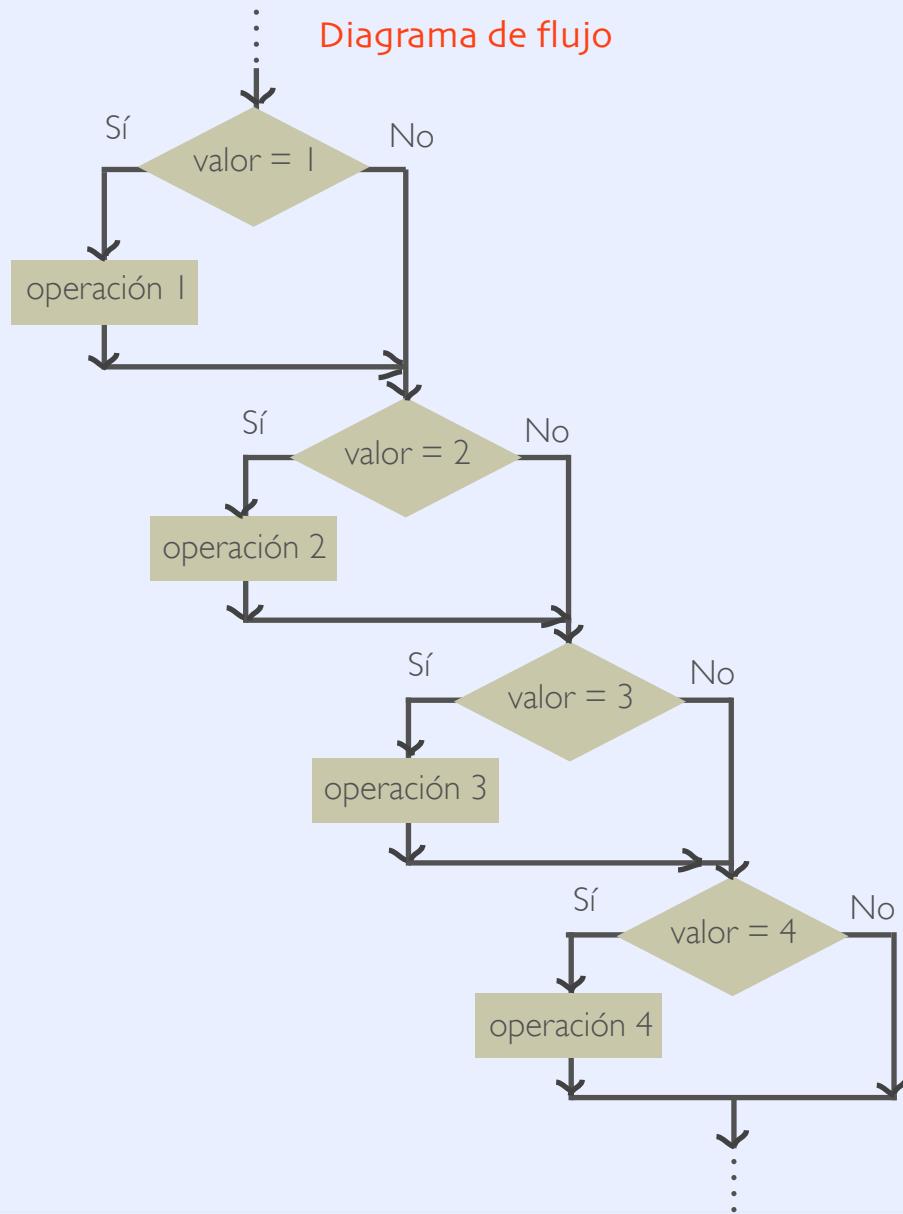
#### Decisiones múltiples

Las decisiones sencillas sólo tienen dos posibles flujos: cuando lo que se está evaluando (comparando, decidiendo) en el rombo es falso o es verdadero. ¿Qué hacer cuando se necesita decidir sobre un rango de posibles valores? Por ejemplo, si se debe hacer algo para cuando un valor sea 1, ó 2, ó 3, ó 4... Diagramar eso con decisiones sencillas, es decir, con rombos solamente, sería muy inadecuado. Observa el siguiente segmento de un algoritmo y su respectivo diagrama de flujo:

#### Algoritmo

1. Si el valor es 1 entonces hacer la operación 1, pero si no es 1, entonces...
2. si el valor es 2 entonces hacer la operación 2, pero si no es 2, entonces...
3. si el valor es 3 entonces hacer la operación 3, pero si no lo es, entonces...
4. si el valor es 4 entonces hacer la operación 4.

#### Diagrama de flujo



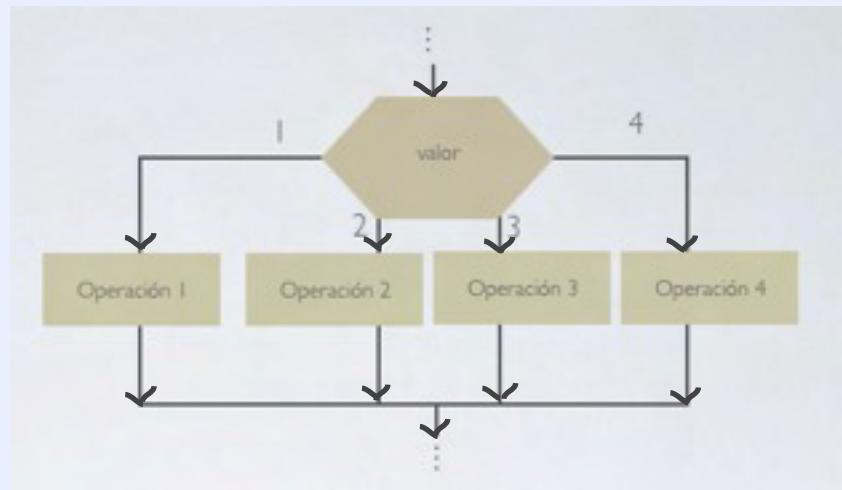
¡Imagínate si fueran veinte posibles opciones! Para casos como este se usa el hexágono, de tal forma que el flujo depende de cada posible valor, por ejemplo:

### Algoritmo

Dependiendo del valor:

- Si el valor es 1 entonces hacer la operación 1.
- Si el valor es 2 entonces hacer la operación 2.
- Si el valor es 3 entonces hacer la operación 3.
- Si el valor es 4 entonces hacer la operación 4.

### Diagrama de flujo

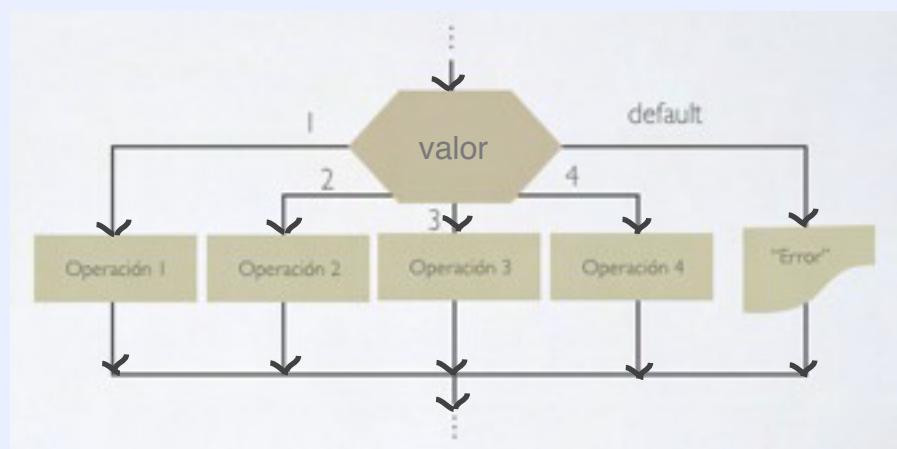


**IMPORTANTE:** Observa cómo el algoritmo está más simplificado, esto es sólo para el ejemplo sobre el uso del hexágono, en realidad se puede usar el mismo algoritmo que en la primera parte. Observa también cómo cada una de las líneas que sale del hexágono está etiquetada con un posible valor. Hay una ventaja extra, si se quiere hacer algo con un valor que no entre en el rango, se usa el "defecto" (default en inglés):

### Diagrama de flujo

Dependiendo del valor:

- Si el valor es 1 entonces hacer la operación 1.
- Si el valor es 2 entonces hacer la operación 2.
- Si el valor es 3 entonces hacer la operación 3.
- Si el valor es 4 entonces hacer la operación 4.
- Si no cae en ningún valor entonces mostrar un error.



Esto último es muy útil para cuando se diagrama un menú de posibles opciones, el defecto puede servir para asegurarse que sólo se elijan las opciones correctas.

## Repeticiones

Las repeticiones son una característica esencial de la Programación Estructurada. Permiten representar la lógica de un algoritmo en menos pasos. Prácticamente, hay tres clases de repeticiones, del inglés For, While y Do... While.

Para todas ellas, se utilizan los símbolos básicos (rectángulo, rombo, línea), pero de tal manera que se represente el bucle o elementos que se repiten. Observa a continuación cada una de las repeticiones, junto con el segmento de algoritmo respectivo. Nota: Se usará un "contador" para exemplificar todas las estructuras, es decir, el valor que decidirá cuándo se debe detener la repetición. Haz clic en cada una de las siguientes pestañas:

### For

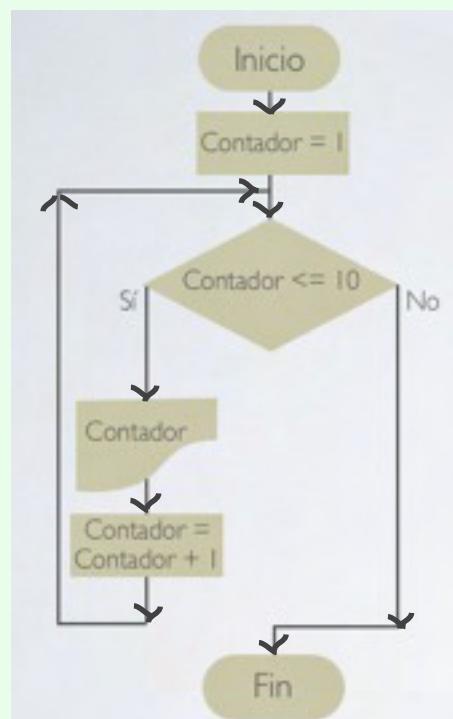
Esta estructura de repetición es especial para recorrer conjuntos, arreglos y cuando **se sabe exactamente la cantidad de veces que un bucle debe repetirse**. A continuación se muestra el algoritmo y el diagrama de flujo de ejemplo para esta estructura de repetición.

#### Algoritmo

Cuando un contador debe ir desde 1 hasta 10, hacer el siguiente bucle:

- Mostrar el contador

#### Diagrama de flujo



#### Nótese:

- El hecho de que el contador sea mayor o igual a 1 y menor o igual a 10 (todo esto es lo mismo que decir que el contador debe ir de 1 a 10) es la condición lógica y esto se pone en un rombo.
- En el algoritmo, el bucle es sólo una operación, pero en el diagrama de flujo debe agregarse el incremento del contador.
- En el diagrama de flujo, el valor inicial del contador debe ponerse antes de la condición lógica.

Observa que en este caso, el algoritmo y el diagrama no corresponden exactamente, pero también es importante que el diagrama deba interpretarse como la representación gráfica del algoritmo.

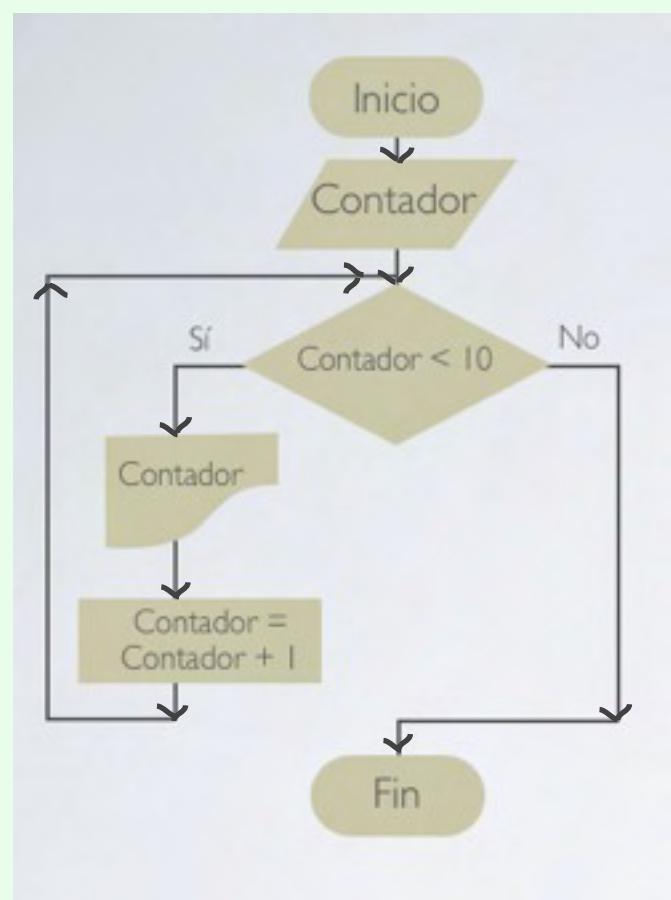
## While

En esta estructura de repetición lo primero que ocurre es que se evalúa la condición lógica, por lo que es posible que el bucle nunca se lleve a cabo. Es especial para cuando no se sabe cuándo debe detenerse la repetición pero **se debe estar seguro al principio de que se cumple la condición**. A continuación se muestra el algoritmo y el diagrama de flujo de ejemplo de esta estructura de repetición.

### Algoritmo

1. Obtener el contador
2. Mientras el contador sea menor de 10, hacer el siguiente bucle:
  - a. Mostrar el contador
  - b. Incrementar el contador en 1

### Diagrama de flujo



#### Nótese:

- El hecho de que el contador sea menor a 10 es la condición lógica y esto se pone en un rombo.

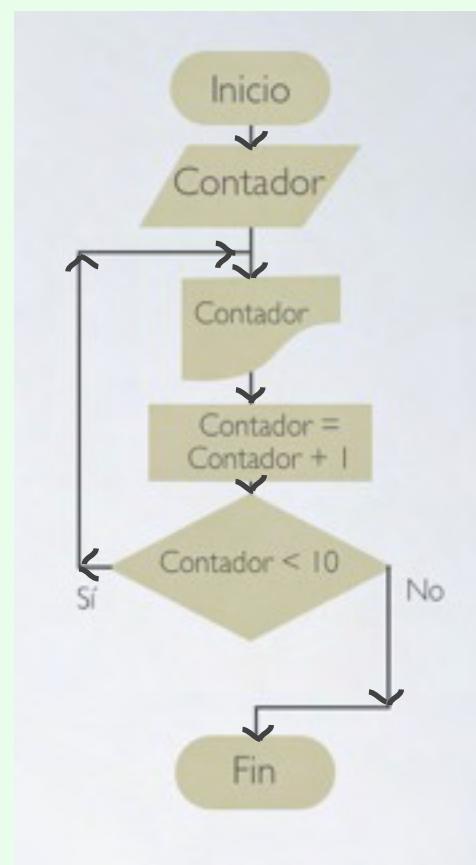
## Do..While

En esta estructura de repetición lo primero que ocurre es el bucle al menos por una vez puesto que la condición lógica se evalúa al terminar el bucle. Es por ello que la repetición puede terminar (ya que la condición puede ser falsa). Es especial para cuando no se sabe cuándo debe detenerse pero **se debe realizar al menos una vez el bucle**. A continuación se muestra el algoritmo y el diagrama de flujo de ejemplo de esta estructura de repetición.

### Algoritmo

1. Obtener el contador
2. Hacer el siguiente bucle...
  - a. Mostrar el contador
  - b. Incrementar el contador en 1
3. ...mientras el contador sea menor de 10

### Diagrama de flujo



### Nótese:

- El hecho de que el contador sea menor a 10 es la condición lógica y esto se pone en un rombo.

## Uso del círculo

El círculo es un símbolo que sirve para describir el flujo de un diagrama complejo. Sirve para evitar diagramas muy difíciles de comprender por cruce de líneas o saturación de símbolos. Es importante recalcar que:

- El círculo no remplaza la funcionalidad del pentágono hacia la derecha (modularidad).
- La funcionalidad del círculo es similar que la del pentágono hacia abajo (conexión a otra página), pero se usa en la misma página y también en páginas diferentes.
- La funcionalidad del círculo es sólo de estructura y sirve para simplificar el diagrama, no sustituye a la decisión ni a la repetición ni a ningún otro símbolo.

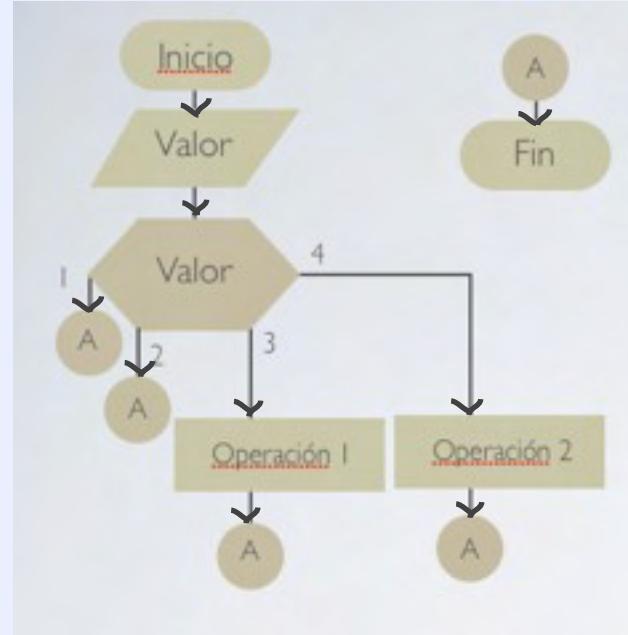
Debido a que es posible que puedan haber más de un círculo en un mismo diagrama, suele ponerse una letra o un número en el centro de ellos, para diferenciar a dónde se debe fluir en el diagrama, inclusive entre páginas diferentes.

Observa los siguientes ejemplos y en especial fíjate que no se pierde el flujo:

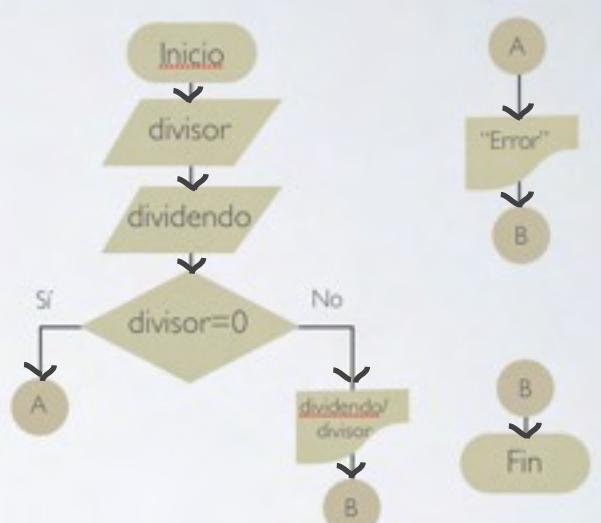
### Ejemplo

### Diagrama de flujo

1.



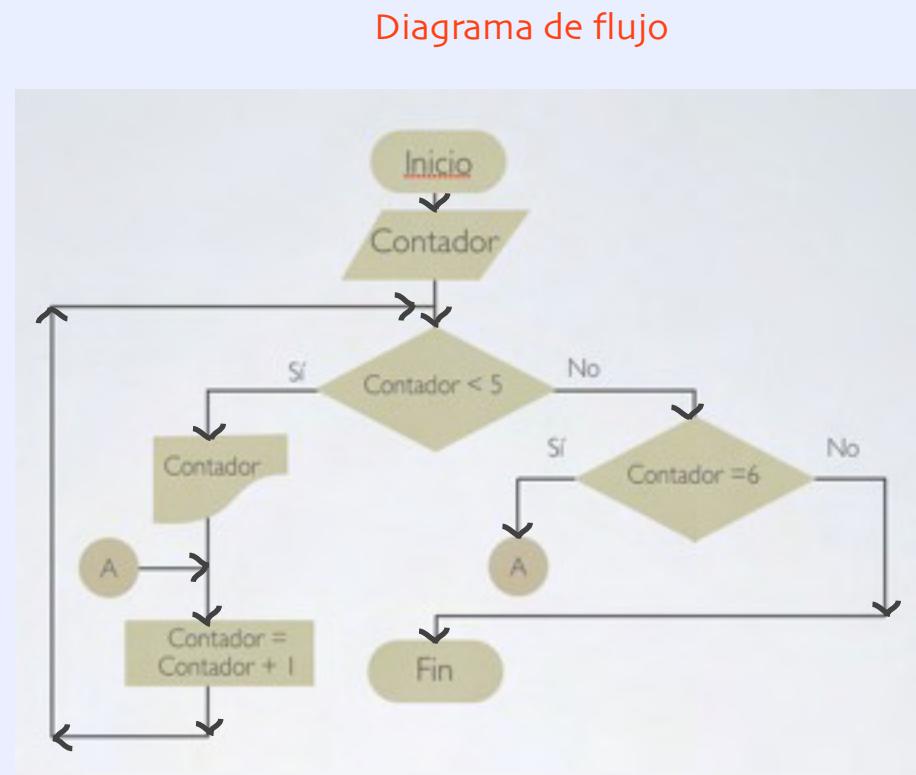
2.



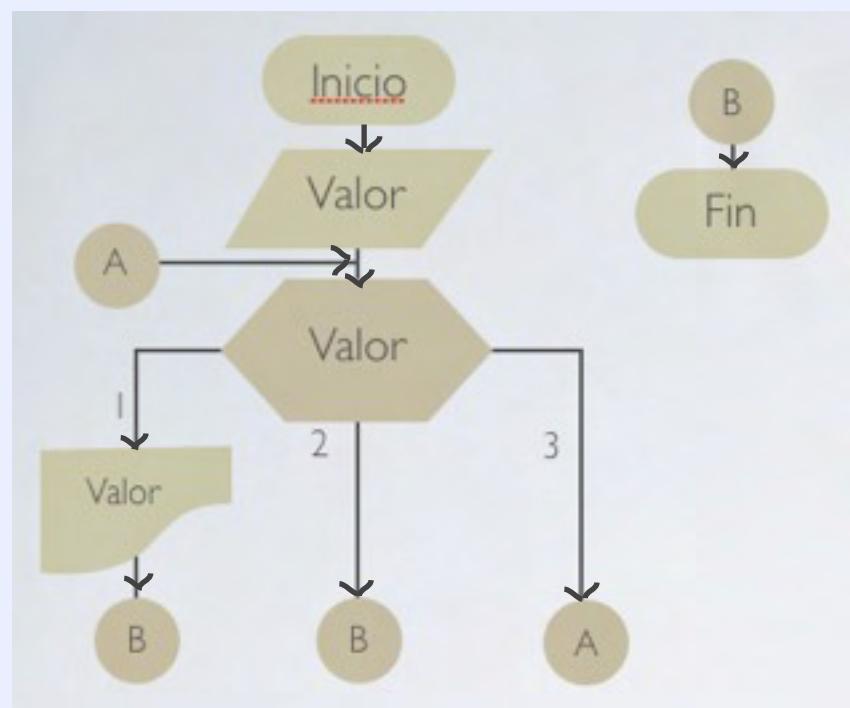
Ahora observa estos **ejemplos incorrectos**:

### Ejemplo

1.



2.



## Conexiones de páginas

El pentágono hacia abajo tiene la misma funcionalidad que el círculo, con la diferencia que se hace para relacionar diagramas que ocupan dos páginas o secciones diferentes en un documento o imagen que contiene un mismo diagrama de flujo. Esto es muy útil para diagramas sumamente largos o complejos. Es importante recalcar que:

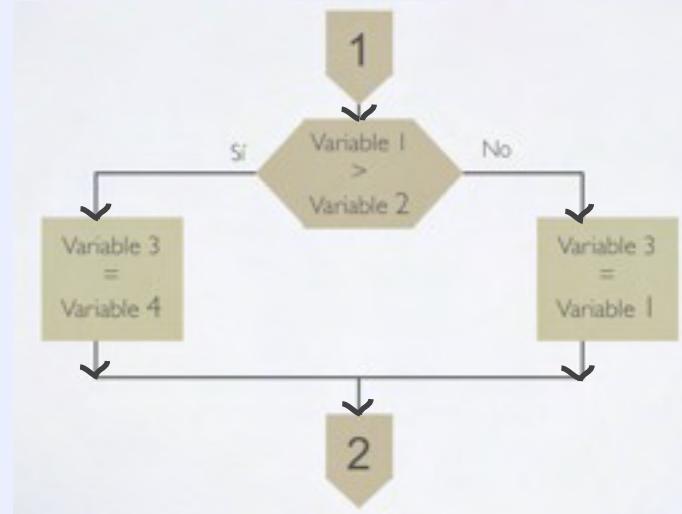
- El pentágono hacia abajo no remplaza la funcionalidad del pentágono hacia la derecha (modularidad).
- La funcionalidad del pentágono hacia abajo es similar que la del círculo, pero se usa solamente entre distintas páginas.
- La funcionalidad del pentágono hacia abajo es sólo de estructura y sirve para interconectar páginas, no sustituye a la decisión ni a la repetición ni a ningún otro símbolo.

Observa el siguiente ejemplo:

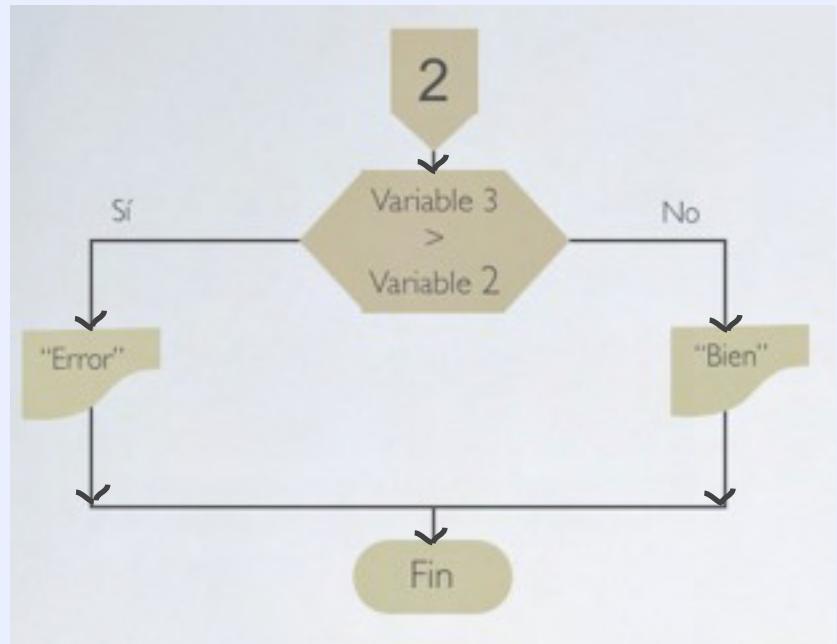
Página 1 de un mismo diagrama:



Página 2 de un mismo diagrama:



Página 3 de un mismo diagrama:



## Modularidad

El pentágono hacia la derecha representa una subrutina que debe realizarse antes de que el flujo principal continúe. Esto es el principio de la modularidad. Es importante recalcar que:

- Cuando se dispone un pentágono a la derecha, el flujo principal se interrumpe.
- La funcionalidad del pentágono hacia la derecha no puede ser remplazado por el círculo, el pentágono hacia abajo ni ningún otro símbolo.
- El flujo secundario puede ponerse en la misma página o en una página diferente.

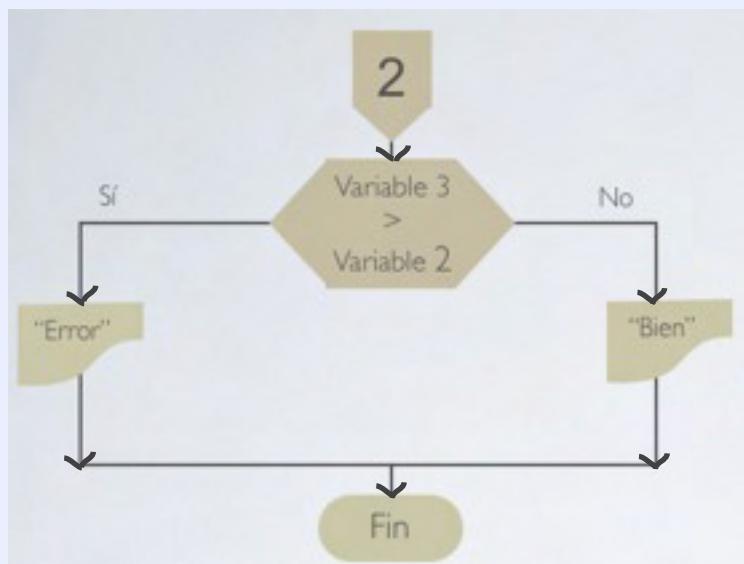
Debido a que es posible que puedan haber más de un pentágono hacia la derecha en un mismo diagrama, suele ponerse una letra o un número en el centro de ellos, para diferenciar a dónde se debe fluir en el diagrama.

Observa los siguientes ejemplos:

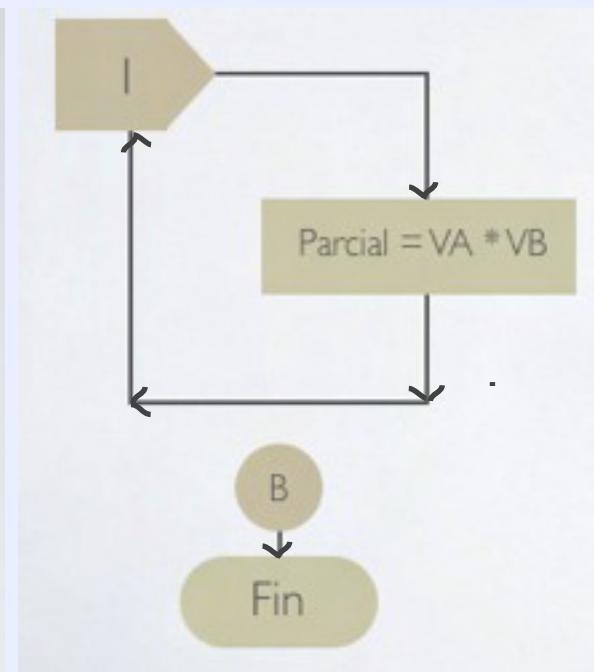
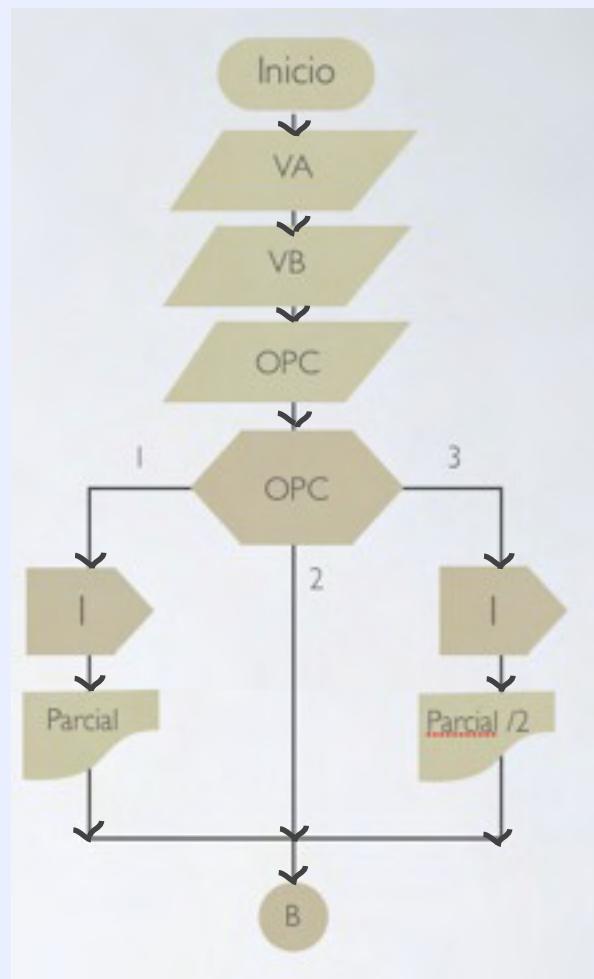
## Ejemplo

## Diagrama de flujo

1.



2.



## referencias

Cairó - Battistutti, O. (2006).

Fundamentos de programación: piensa en C. Pearson Educación.