Métodos computacionales en la Ingeniería Química

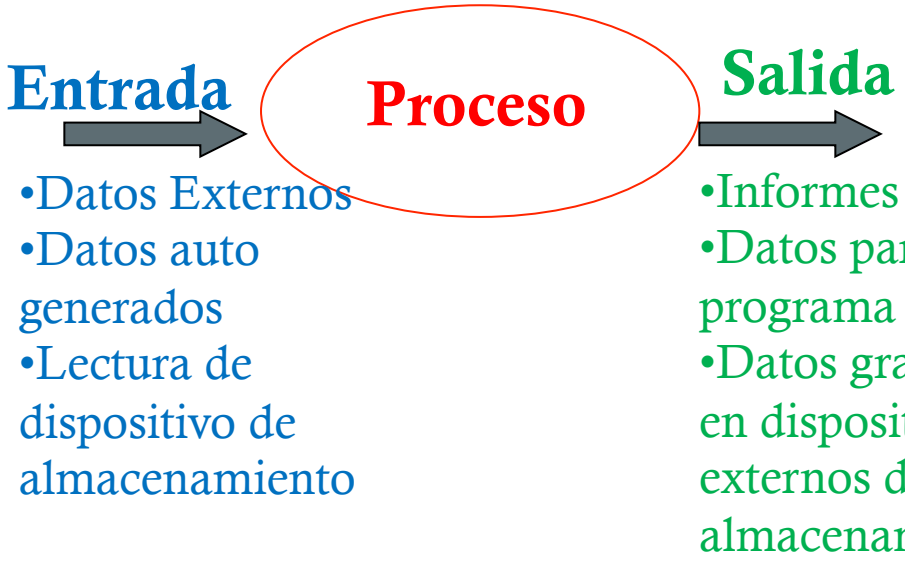
Luis Alejandro Benavides Vázquez

Correo: **luisbv.89@gmail.com**

¿Qué es un problema?

Definición: es una situación concreta sobre la cual se quiere implementar una solución.

Solución: procedimiento que nos lleva a satisfacer ciertos requerimientos



Pasos para resolver problema

1. Entender el problema
2. Buscar soluciones
3. Elegir solución
4. Diseñar solución
5. Implementar solución
6. Validar solución

¿Qué es un algoritmo?

Definición: Procedimiento detallado para resolver un problema en pasos y en un tiempo finito.

Se especifican en base a operaciones básicas que controlan las variables y el flujo del algoritmo

El algoritmo lleva desde un **estado inicial** a un **estado final**

El algoritmo recibe **Entradas** y entrega **Salidas**

Tipos de Algoritmos

Cualitativos: Se describen los pasos utilizando palabras.

Cuantitativos: Se utilizan cálculos numéricos para definir los pasos del proceso.

Gráfico: Es la representación gráfica de las operaciones que realiza un algoritmo (diagrama de flujo).

No gráficos: Representa de forma descriptiva las operaciones que debe realizar un algoritmo (pseudocódigo).

Diseño del algoritmo

Las características de un buen algoritmo son:

1. Debe tener un punto particular de inicio.
2. Debe ser definido, no debe permitir dobles interpretaciones.
3. Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
4. Debe ser finito en tamaño y tiempo de ejecución

Codificación

Definición: operación de escribir la solución del problema (de acuerdo a la lógica del diagrama de flujo o pseudocódigo), en una serie de instrucciones detalladas, en un código reconocible por la computadora.

La serie de instrucciones escritas para un programa se les conoce como **código fuente** y se escriben en un **lenguaje de programación** que puede ser de bajo, medio o alto nivel.

Conceptos Básicos de Algoritmos

La forma en que se ejecutan las operaciones básicas en una computadora, es similar a lo que ocurre en nuestro cerebro.

Por ejemplo, para sumar dos valores:

- Primero debemos pedirle a alguien que nos diga el primer valor.
- Luego de que conocemos este valor, debemos almacenarlo (para recordarlo después) en una neurona (Suponemos que un valor se puede almacenar en una neurona).

Ya conocemos el primer valor y está almacenado en nuestro cerebro.

Conceptos Básicos de Algoritmos (cont.)

- Ahora debemos pedir el segundo valor.
- Una vez conocido, lo almacenamos en otra neurona distinta de la anterior. ¿ Por qué?
- Ahora que conocemos los dos valores procedemos a sumarlos, y dicho resultado lo almacenamos en otra neurona distinta de las anteriores.
- Por último, le decimos el resultado a la persona que nos entrego los números.

Conceptos Básicos de Algoritmos (cont.)

- Al menos necesitamos 3 neuronas para sumar dos números.
- Le pedimos explícitamente que nos dijeran dichos valores.
- Le asignamos dichos valores a las neuronas
- La suma la realizó nuestro cerebro de forma mecánica. Note que no existen detalles de la operaciones básicas (*,/,+, -).
- Finalmente se da el resultado

Ejemplo 1

Algoritmo para sumar dos números:

1. Definimos tres neuronas
2. Pedimos el primer valor
3. Almacenamos ese valor en la neurona 1.
4. Pedimos el segundo valor
5. Almacenamos ese valor en la neurona 2.
6. Almacenamos la suma de las neuronas 1 y 2 en la neurona 3
7. Entregamos el resultado que se encuentra en la neurona 3.

Conceptos Básicos de Algoritmos

- Sin embargo, en los lenguajes de programación no se pueden usar **neuronas**, pero podemos definir **variables**.
- En lugar de usar neurona 1 y neurona 2, se utilizan espacios de memoria que llamaremos "var1" y "var2", y así sucesivamente. También las podemos llamar "x1" y "x2" ó "x" e "y" ….

Ejemplo 2

Algoritmo para multiplicar tres números:

1. Definimos cuatro variables (**var1**, **var2**, **var3**, **var4**).
2. Pedimos el primer valor.
3. Almacenamos ese valor en **var1**.
4. Pedimos el segundo valor.
5. Almacenamos ese valor en **var2**.
6. Pedimos el tercer valor.
7. Almacenamos ese valor en **var3**.
8. Almacenamos la multiplicación de las variables en **var4**.
9. Entregamos el resultado que se encuentra en **var4**.

Ejercicios Algoritmos

- Crear un algoritmo que multiplica tres números y muestre el resultado
- Crear un algoritmo que divida dos números y muestre el resultado
- Crear un algoritmo que muestre en forma ordenada tres número enteros ingresados desde teclado.
- Crear un algoritmo donde una persona ingrese su edad y muestre por pantalla si es mayor de edad (mayoría de edad 18 años).
- Crear un algoritmo que permite sumar “n” números y muestre el resultado. El valor de “n” debe ser ingresado por teclado al igual que los números que se sumarán.
- Crear un el algoritmo que encuentre el número mayor de N números enteros positivos ingresados por teclado

Pseudocódigo

- **El pseudocódigo** es una manera de escribir algoritmos de forma poco estricta (con una sintaxis relajada), pero intentando acercar las ideas del algoritmo a estructuras y sintaxis parecidas a las de los lenguajes de alto nivel en los que vamos a programar el algoritmo.
- Es un lenguaje de especificación de algoritmos, pero muy parecido a cualquier lenguaje de programación, por lo que luego su traducción al lenguaje de programación es muy sencillo. Nos centramos más en la lógica del problema.
- El pseudocódigo también va a utilizar una serie de palabras claves o palabras especiales que va indicando lo que significa el algoritmo.

Pseudocódigo: Sintaxis Utilizada.

1.INICIO y FIN: Por donde empieza y acaba el algoritmo.

2.DATOS: Aquí se declaran e inicializan las variables que utilizará el algoritmo.

3.ALGORITMO: En esta sección se escribe el algoritmo.

Pseudocódigo de un algoritmo genérico:

```
INICIO.
        DATOS:                          ** esto es un comentario **
                entero a ;              ** declaración de una variable entera **
                real b = 0 ;            ** declaración e inicialización de una variable **

        ALGORITMO:
                leer a ;
                b = a + 5 ;
                escribir b ;
FIN.
```

Ejemplo: *Calcular la altura en pulgadas (1 pulgada=2.54 cm) y pies (1 pie=12 pulgadas), a partir de la altura en centímetros, que se introduce por el teclado.*

Inicio
1. **IMPRIMIR** 'Introduce la altura en centimetros: '
2. **LEER**: altura
3. **CALCULAR** pulgadas=altura*2:54
4. **CALCULAR** pies=pulgadas/12
5. **IMPRIMIR** 'La altura en pulgadas es: ', pulgadas
6. **IMPRIMIR** 'La altura en pies es : ', pies

Fin

Ejercicios pseudocódigo

- Crear un pseudocódigo que multiplica tres números y muestre el resultado
- Crear un pseudocódigo que divida dos números y muestre el resultado
- Crear un pseudocódigo que muestre en forma ordenada tres número enteros ingresados desde teclado.
- Crear un pseudocódigo donde una persona ingrese su edad y muestre por pantalla si es mayor de edad (mayoría de edad 18 años).
- Crear un pseudocódigo para el algoritmo que permite sumar “n” números y muestre el resultado. El valor de “n” debe ser ingresado por teclado al igual que los números que se sumarán.
- Crear un pseudocódigo para el algoritmo que encuentre el número mayor de N números enteros positivos ingresados por teclado

Diagramas de flujo

El uso de **diagramas de flujo** como herramienta de programación tiene beneficios que resumidamente se detallan:

Rápida comprensión de las relaciones.

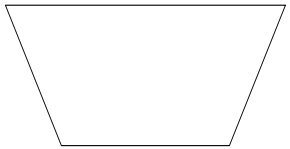
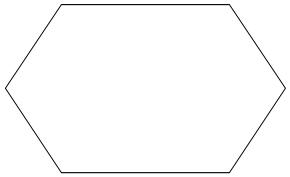
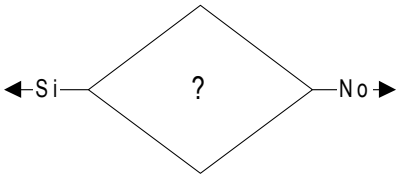
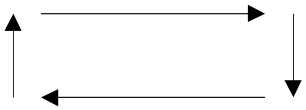
Se pueden usar como modelos de trabajo para el diseño de nuevos programas.

Documentación adecuada de los programas.

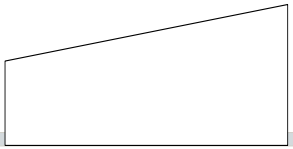
Produce una codificación eficaz en los programas.

Depuración y pruebas ordenadas de programas.

Fácil de traducir a cualquier lenguaje de programación.

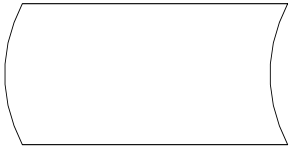
SIMBOLOS	FUNCION
	Represena el Fin y Comienzo del diagrama
	Entrada /salida - Cualquier tipo de introducción de datos en la memoria desde los periféricos de entrada o registro de la información procesada en un periférico de salida (no interesa el soporte)
	Operación o proceso - acciones a realizar(sumar dos números, calcular raíz cuadrada, asignaciones, etc.-
	Subrutina - llamada a un subprograma que es un módulo independiente del programa principal que realiza una determinada tarea y regresa a la siguiente instrucción de donde fue llamada.
	Decisión - operaciones lógicas o de compración entre datos y en función del resultado determina cual de los dos distintos caminos alternativos del programa se debe seguir. Normalmente tiene dos respuestas SI o NO
	Conector - enlaza dos partes cualesquiera del diagrama mediante un conector de salida y otro de entrada. Siempre dentro de la misma página
	Línea de FLujo - indica el sentido de la ejecución de las operaciones

Conector - idem al conector anterior pero usando distitnas páginas

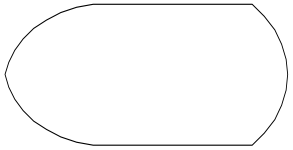


Teclado - introducción manual de datos desde el teclado

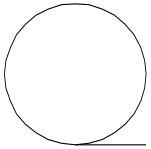
Impresora - salida de datos en forma impresa



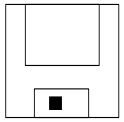
Disco Magnético - para lectura o grabación de datos



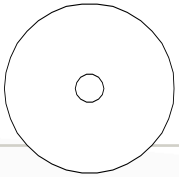
Pantalla - Entrada / Salida de datos por pantalla



Cinta Magnética - para lectura o grabación de datos



Disquete o disco flexible - para lectura o grabación de datos



CD - Disco Compacto - para lectura o grabación de datos

Estructuras Básicas

Las estructuras básicas son las tres siguientes:

Secuencia
Alternativa o Selectiva
Iteración o Repetitiva

Secuencia

Se compone de un grupo de acciones que se realizan todas y en el orden en que están escritas, sin posibilidad de omitir ninguna de ellas.

Las tareas se suceden de forma tal que la salida de una de ellas es la entrada de la siguiente y así sucesivamente hasta el final del proceso.

Pseudocódigo:

```
INICIO
   Levante la bocina
   Espere tono
   Marque el número
   Espere que contesten
   Hable con la otra persona
   Cuelgue la bocina
FIN
```

Diagrama de flujos:

Alternativa o Selectiva

Permite la selección entre dos grupos de acciones dependiendo de que una determinada condición se cumpla o no.

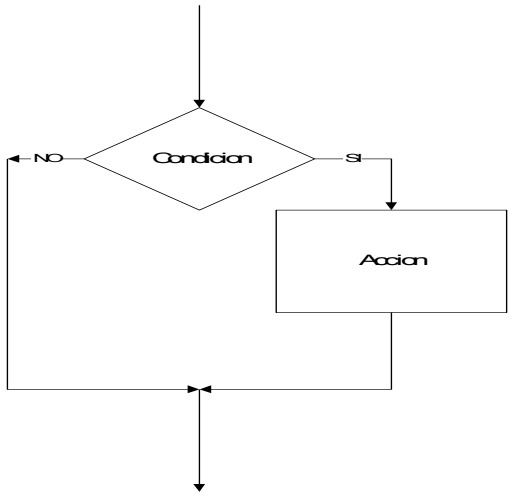
Estas estructuras se utilizan para tomar decisiones lógicas; por ello recibe también el nombre de *estructuras de decisión o alternativas o condicional*.

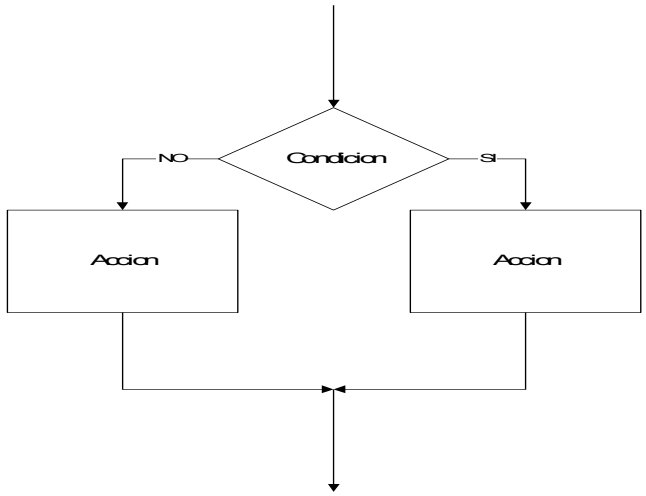
Las condiciones que se especifican usan expresiones lógicas y usan la figura geométrica en forma de rombo. Estas estructuras pueden ser: *Simples o dobles.*

Alternativa o Selectiva (cont.)

Simple: Solo obliga a realizar acciones si se cumple la condición. El "no cumplimiento" de la condición implica que no se realizará ninguna acción.

Doble: El cumplimiento o no de la condición lógica obliga a la ejecución de diferentes grupos de acciones.

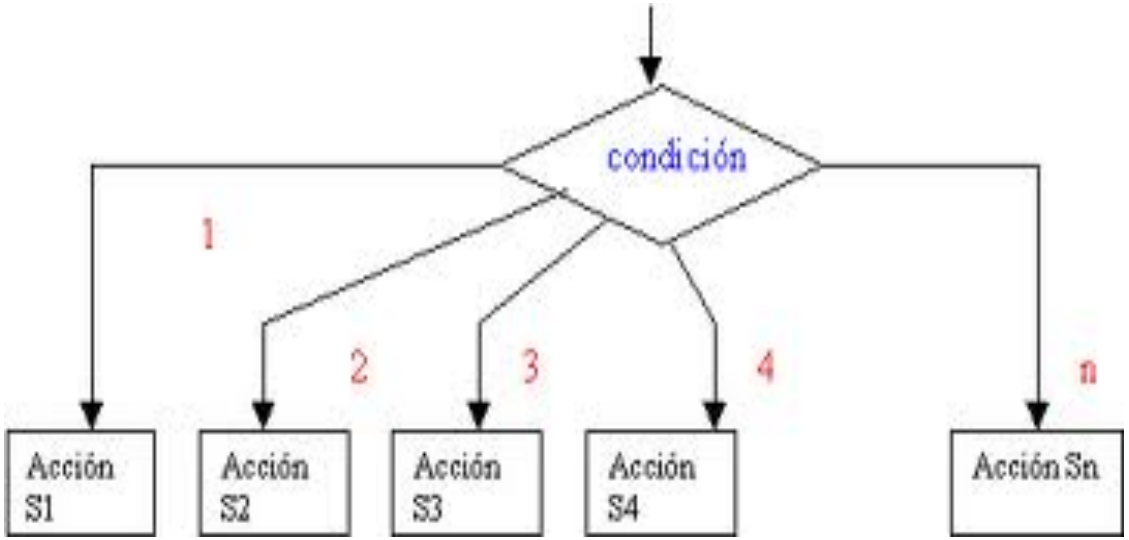




Alternativa o Selectiva (cont.)

Estructura de elección entre varios casos

•Este tipo de estructura permite decidir entre varios caminos posibles, en función del valor que tome una determinada instrucción.



Iteración o Repetitiva

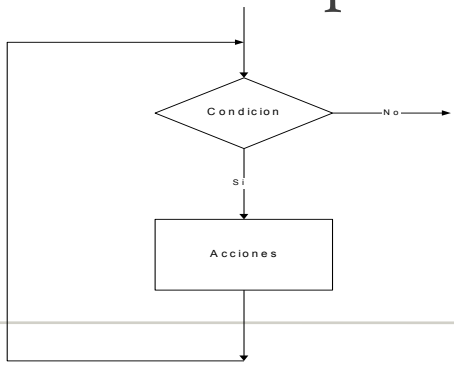
Permite repetir una o varias instrucciones un número determinado de veces que vendrá determinado por una condición. Esta condición se conoce como *condición de salida.*

A estos tipos de estructuras se las conoce también con el nombre de **bucles o rulos** y al hecho de repetir la ejecución de acciones se llama **iteración**.

Iteración o Repetitiva (cont.)

HACER MIENTRAS: Se caracteriza porque la condición de salida del bucle está situada al **comienzo** del mismo, es decir las acciones la hace mientras se cumple determinada condición.

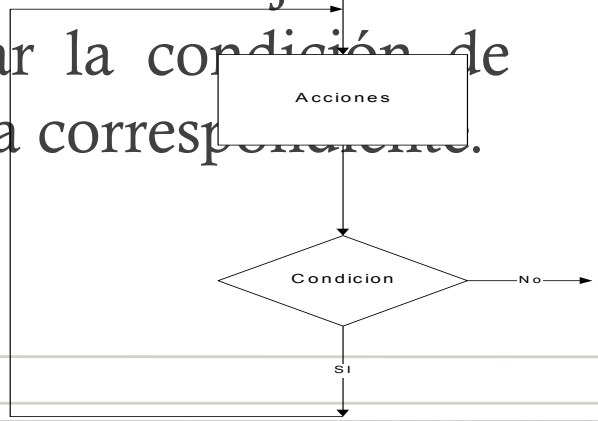
Cuando se ejecuta una estructura de este tipo, lo que primero se hace es evaluar la condición, si la misma es falsa no se realiza ninguna acción. Si la condición resulta verdadera entonces se ejecuta el cuerpo del bucle (acciones de la Figura). Este mecanismo se repite mientras la condición sea verdadera.



Iteración o Repetitiva (cont.)

HACER HASTA: Se caracteriza porque la condición que controla la realización de las acciones del bucle está al **final** del mismo. En este tipo de iteración las acciones se repiten mientras la condición sea falsa, lo opuesto a la estructura **hacer mientras.**

Este tipo de bucle se usa para situaciones en las que se desea que un conjunto de instrucciones se ejecute al menos una vez antes de comprobar la condición de iteración. La figura muestra la gráfica correspondiente.

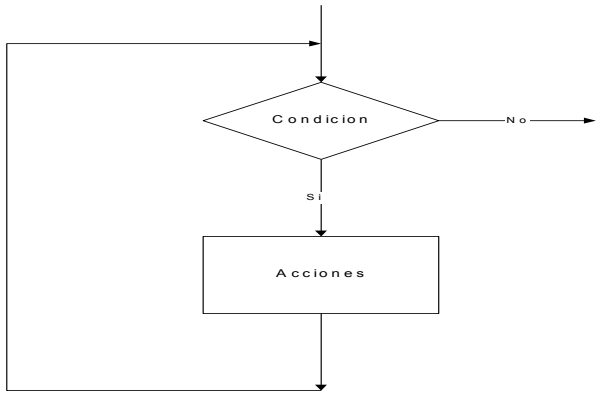


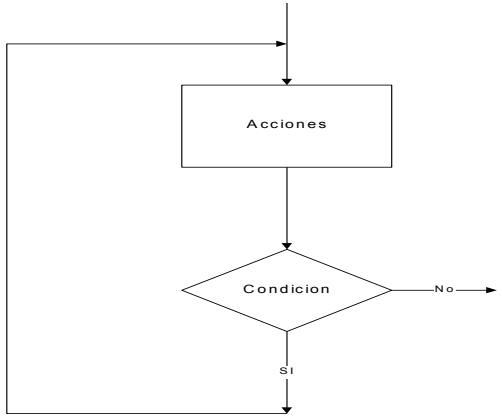
Iteración o Repetitiva

Se puntualizan algunas diferencias entre estas dos estructuras:

- La estructura **mientras** termina cuando la condición es falsa, en cambio la estructura **hasta** termina cuando la condición es verdadera.

En la estructura **hasta** el cuerpo del bucle se ejecuta siempre al menos una vez, en cambio en la estructura **mientras** permite que el cuerpo del bucle nunca se ejecute.





Ejemplo. Hacer un programa que multiplique dos números enteros.

Sabemos que “5 x 3 = 15″ es lo mismo que “5 + 5 + 5 = 15”.

Variables

- multiplicando: entero (nos indica el número que vamos a sumar)
- multiplicador: entero (nos indica el número de veces que lo vamos a sumar)
- resultado: entero (en esta variable asignaremos el resultado)
- indice: entero (nos indicara el número de veces que el número se ha sumado)

Ejemplo. Hacer un programa que multiplique dos números enteros.

Algoritmo

1) Asignamos el número 5 a multiplicando

2) Asignamos el número 3 a multiplicador

3) Asignamos el número 0 a resultado

4) Asignamos el número 0 a indice

5) Sumamos multiplicando y resultado

6) Asignamos a resultado la suma

7) Incrementamos 1 a indice

8) Mientras indice sea menor a multiplicador regresamos al paso 5 de lo contrario continua

9) Muestra el resultado

10) Finalizar

Ejemplo. Hacer un programa que multiplique dos números enteros.

El siguiente paso es…

Prueba de escritorio

La prueba de escritorio es la ejecución manual de nuestro algorítmo

Ponemos a prueba nuestro algoritmo y nos mostrara si tenemos errores (por lo que tendremos que modificar el algoritmo) o si esta bien diseñado. Básicamente es el registro de las variables.

Siguiendo paso a paso nuestro algoritmo, obtendremos la siguiente tabla.

Ejemplo. Hacer un programa que multiplique dos números enteros.

Multiplicado	5			
Multiplicador	3			
Resultado	0	5	10	15
Indice	0	1	2	3

Vemos que el ultimo registro de la variable resultado, es 15, por lo que nuestro algoritmo esta funcionando correctamente. Podemos probar con otros número.

Ejemplo. Hacer un programa que multiplique dos números enteros.

Diagramas de flujo

Una vez que hemos probado muestro algoritmo con la prueba de escritorio y el resultado es el correcto, podemos seguir a diseñar el diagrama de flujo.

Cada paso de nuestro algoritmo en un procedimiento y se representan con un rectángulo. (Podemos agrupar varios procedimientos en un solo rectángulo, pero no es lo indicado)

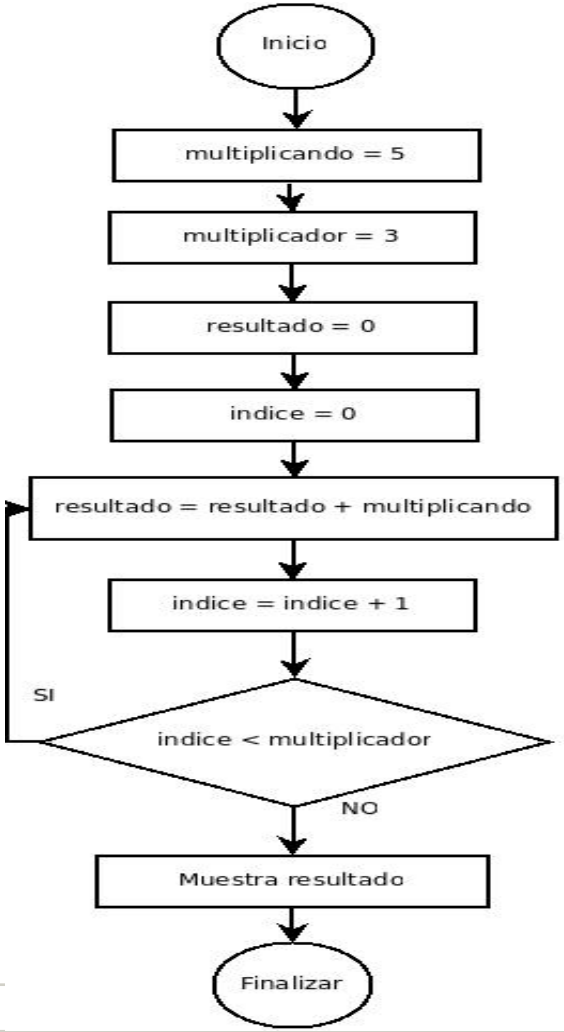
Cada condición como el paso número 8 se representa con un rombo.

Este será el diagrama de flujo de nuestro algoritmo.

Ejemplo. Hacer un programa que multiplique dos números enteros.

1. Inicio
2. multiplicando = 5
3. multiplicador = 3
4. resultado = 0
5. indice = 0
6. hacer
7. resultado = resultado + multiplicando
8. indice = indice + 1
9. mientras indice < multiplicador
10. imprime resultado
11. finalizar

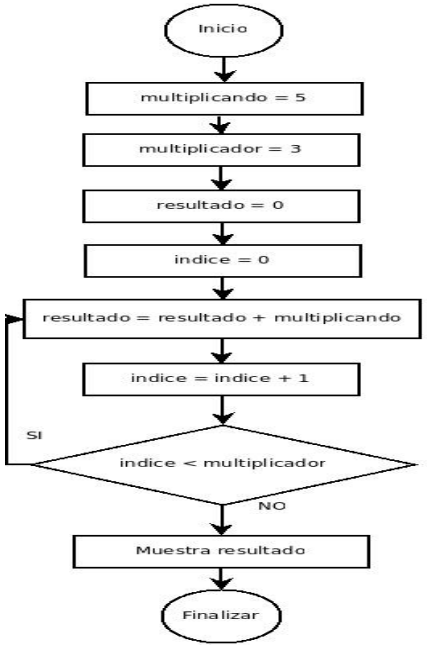
Un programa lo podemos dividir en bloques, por ejemplo; de la línea 6 a la 9 es un bloque, y para identificar cada bloque en el código lo podemos escribir después de unos espacios y así identificar ciertos procesos. Esto nos sirve para en códigos muy grandes.



Ejemplo. Hacer un programa que multiplique dos números enteros.

- **Y finalmente nos pasamos a la computadora y escribimos el código en algun lenguaje de programacion, en nuestro caso C.**

1. Inicio
2. multiplicando = 5
3. multiplicador = 3
4. resultado = 0
5. indice = 0
6. hacer
7. resultado = resultado + multiplicando
8. indice = indice + 1
9. mientras indice < multiplicador
10. imprime resultado
11. finalizar



Inicio
multiplicando= 5
multiplicador = 3
resultado = 0
indice = 0

si indice < multiplicador entonces
resultado = resultado + multiplicando
indice = indice + 1
fin si

mostrar resultado
Fin

Ejercicios Diagrama de Flujo

- Crear un diagrama de flujo que multiplica tres números y muestre el resultado
- Crear un diagrama de flujo que divida dos números y muestre el resultado
- Crear un diagrama de flujo que muestre en forma ordenada tres número enteros ingresados desde teclado.
- Crear un diagrama de flujo donde una persona ingrese su edad y muestre por pantalla si es mayor de edad (mayoría de edad 18 años).
- Crear un diagrama de flujo para el algoritmo que permite sumar “n” números y muestre el resultado. El valor de “n” debe ser ingresado por teclado al igual que los números que se sumarán.
- Crear un diagrama de flujo para el algoritmo que encuentre el número mayor de N números enteros positivos ingresados por teclado

Ejercicios

- Cree algoritmo, pseudocódigo y el diagrama de flujo permita sumar tres números ingresados por teclado y muestre el resultado
- Crear algoritmo, pseudocódigo y el diagrama de flujo para que resuelva la ecuación de primer grado y muestre el resultado
- Cree algoritmo, pseudocódigo y el diagrama de flujo para que resuelva la ecuación de segundo grado, para valores reales e imaginarios, y muestre el resultado