



**Universidad Nacional
Autónoma de México**
Facultad de Ingeniería



Fundamentos de Programación (1122)

Laboratorios de computación
salas A y B

Profesor: M.I. Marco Antonio Martínez Quintana
Semestre 2021-1

Practica No. 4

Diagramas de Flujo

Grupo: 1129

No. de Equipo de cómputo empleado: No aplica

No. de Lista o Brigada: No aplica

No. de Lista: 42

Nombre: Adolfo Román Jiménez

Objetivo:

Elaborar diagramas de flujo que representen soluciones algorítmicas vistas como una serie de acciones que comprendan un proceso.

Introducción:

El diagrama de flujo es una herramienta fundamental para la elaboración de un procedimiento, ya que a través de ellos podemos ver gráficamente y en forma consecutiva el desarrollo de una actividad determinada. Los diagramas de flujo representan en forma gráfica la secuencia que siguen las operaciones de un determinado procedimiento y/o el recorrido de las formas o los materiales. Muestran las unidades administrativas (procedimiento general) o los puestos que intervienen (procedimiento detallado) para cada operación descrita, y puede indicar, además, el equipo que se utilice en cada caso. El Diagrama de Flujo de Proceso es una representación esquemática del proceso, sus condiciones de operación normal y su control básico.

Formas de los diagramas de flujo

Los diagramas de flujo poseen símbolos que permiten estructurar la solución de un problema de manera gráfica. A continuación, se muestran los elementos que conforman este lenguaje gráfico.

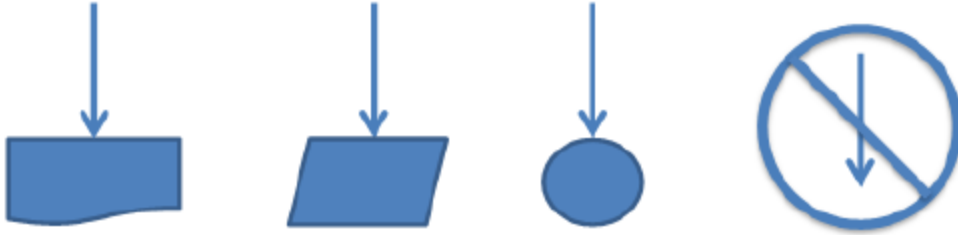
1. Todo diagrama de flujo debe tener un inicio y un fin.



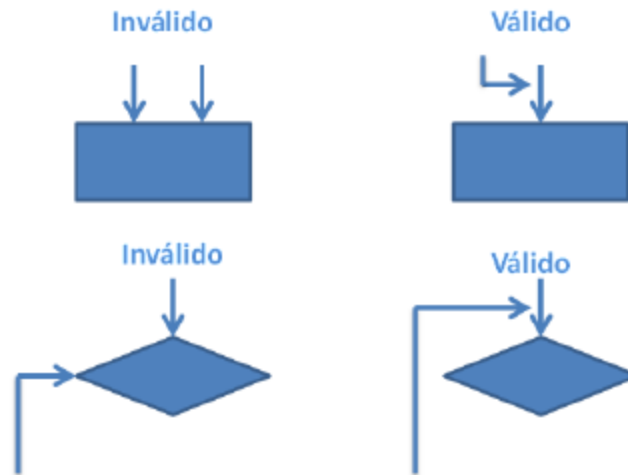
2. Las líneas utilizadas para indicar la dirección del flujo del diagrama deben ser rectas, verticales u horizontales, exclusivamente.



3. Todas las líneas utilizadas para indicar la dirección del flujo del diagrama deben estar conectadas a un símbolo.



4. El diagrama debe ser construido de arriba hacia abajo (top-down) y de izquierda a derecha (left to right).
5. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación en el que se va a codificar la solución.
6. Se recomienda poner comentarios que expresen o ayuden a entender un bloque de símbolos.
7. Si la extensión de un diagrama de flujo ocupa más de una página, es necesario utilizar y numerar los símbolos adecuados.
8. A cada símbolo solo le puede llegar una línea de dirección de flujo.



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

Los diagramas de flujo poseen símbolos que permiten estructurar la solución de un problema de manera gráfica. Por tanto es fundamental conocer los elementos que conforman este lenguaje gráfico.



Representa el inicio o el fin del diagrama de flujo.

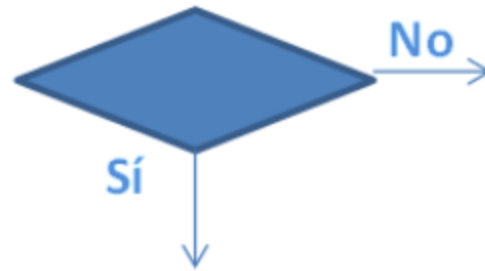
Datos de entrada. Expresa lectura de datos.





Proceso. En su interior se expresan asignaciones u operaciones.

Decisión. Valida una condición y toma uno u otro camino.



Escritura. Impresión del o los resultado(s).

Dirección de flujo del diagrama.



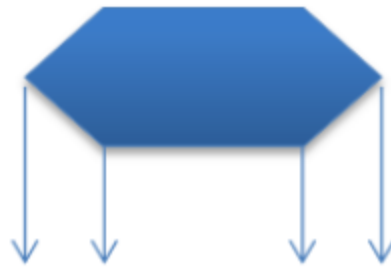
Conexión dentro de la misma página.

Conexión entre diferentes páginas.



Módulo de un problema. Llamada a otros módulos o funciones.

Decisión múltiple. Almacena un selector que determina la rama por la que sigue el flujo.



El diagrama de flujo para construir un diagrama de flujo es el siguiente:



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.

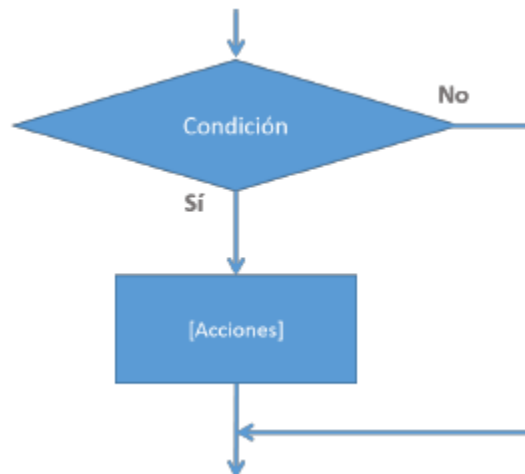
Ejemplo

```
x: REAL
x ← 5.8
x ← x*2
```

Estructuras de control condicionales (o selectivas)

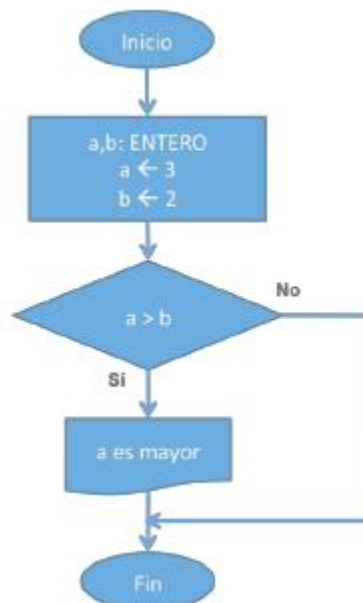
Las estructuras de control condicionales permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa) y, dependiendo del resultado, se realiza uno u otro flujo de instrucciones. Estas estructuras son mutuamente excluyentes (o se ejecuta una acción o se ejecuta la otra).

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



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

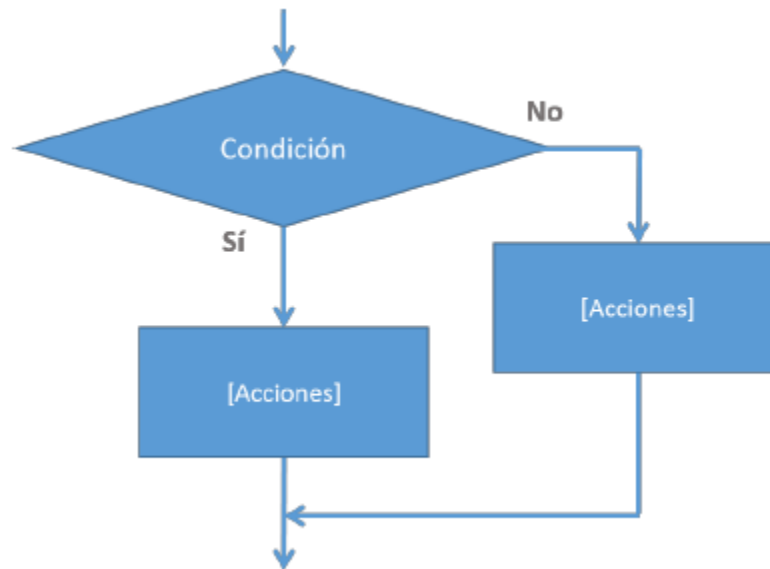
Ejemplo



// >>> a es mayor

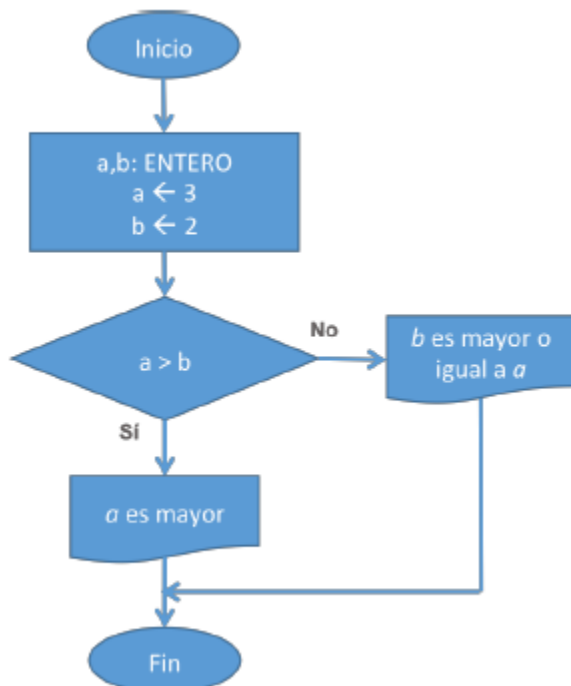
NOTA: La línea // >>> **valor**, indica el resultado que genera el ejemplo.

La estructura condicional completa es SI-DE LO CONTRARIO (IF-ELSE):



Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque Sí. Si no se cumple la condición se ejecutan las instrucciones del bloque No. Al final el programa sigue su flujo normal.

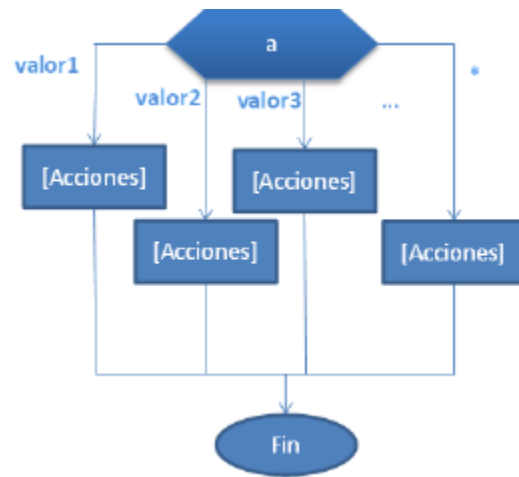
Ejemplo



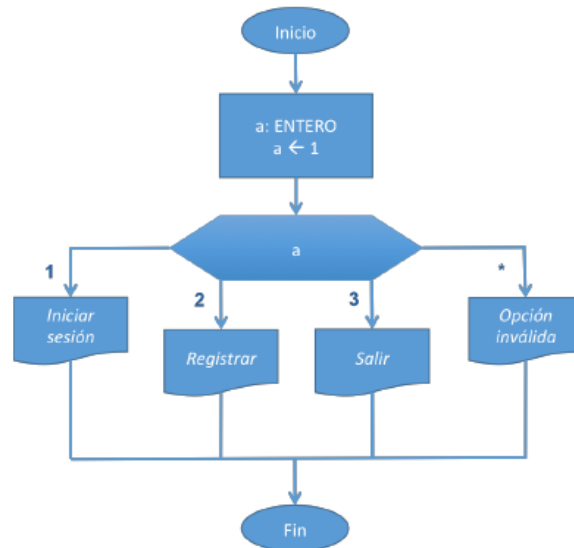
// >>> b es mayor

La estructura condicional SELECCIONAR-CASO valida el valor de la variable que está en el hexágono y comprueba si es igual al valor que está definido en cada caso (líneas que emanan

del hexágono). Si la variable no tiene el valor de algún caso se va a la instrucción por defecto (*).



Ejemplo

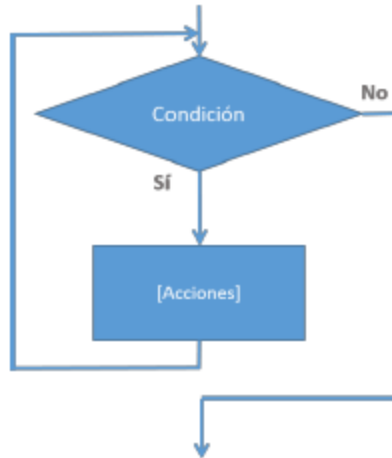


// >>> "Iniciar sesión"

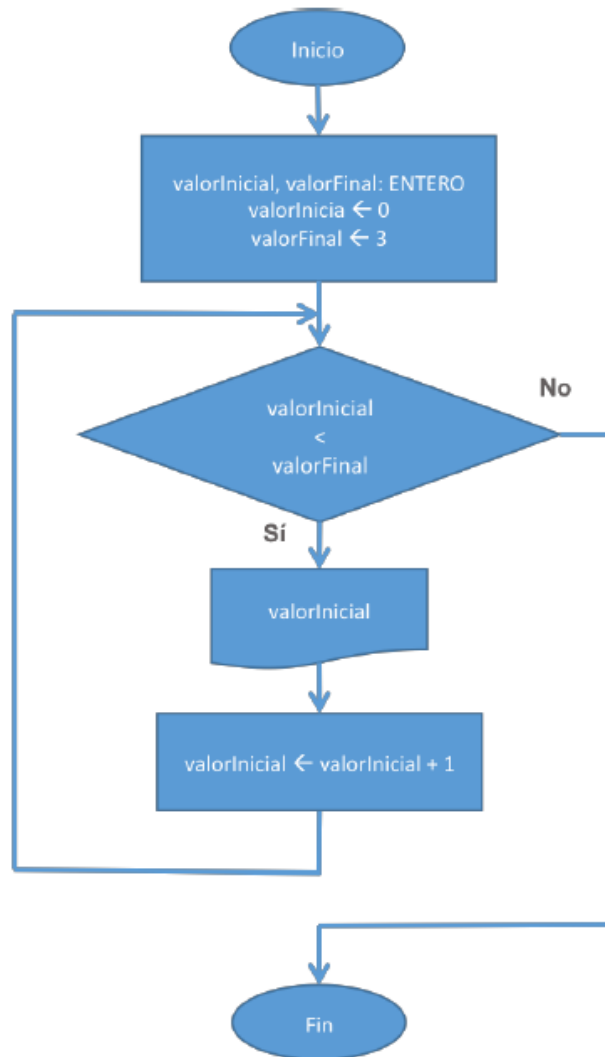
Estructuras de control iterativas o repetitivas

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

La estructura MIENTRAS primero valida la condición y si ésta es verdadera procede a ejecutar el bloque de instrucciones de la estructura, de lo contrario rompe el ciclo y continúa el flujo normal del programa.



Ejemplo

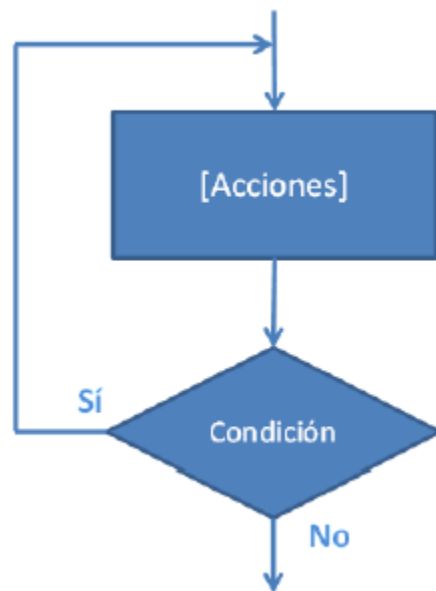


//>>> 0

//>>> 1

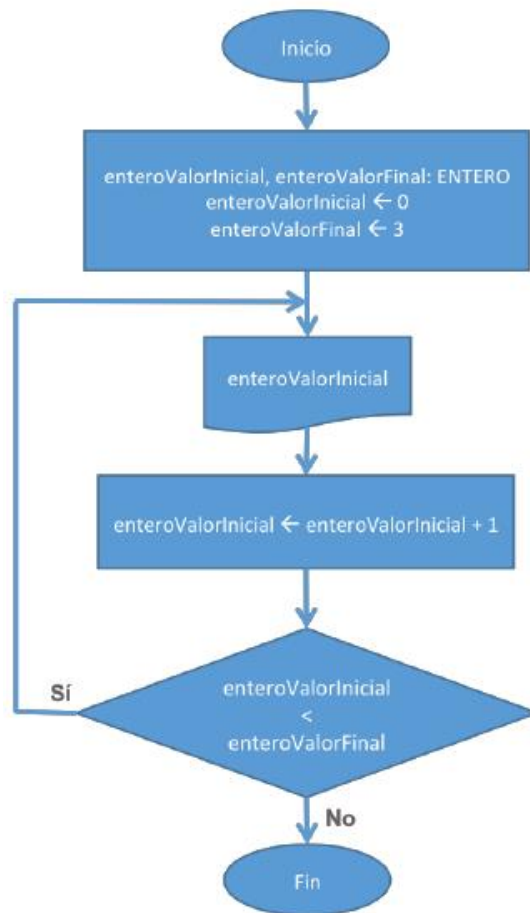
//>>> 2

La estructura HACER-MIENTRAS primero ejecuta las instrucciones descritas en la estructura y al final valida la expresión lógica.



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

Ejemplo



// >>> 0

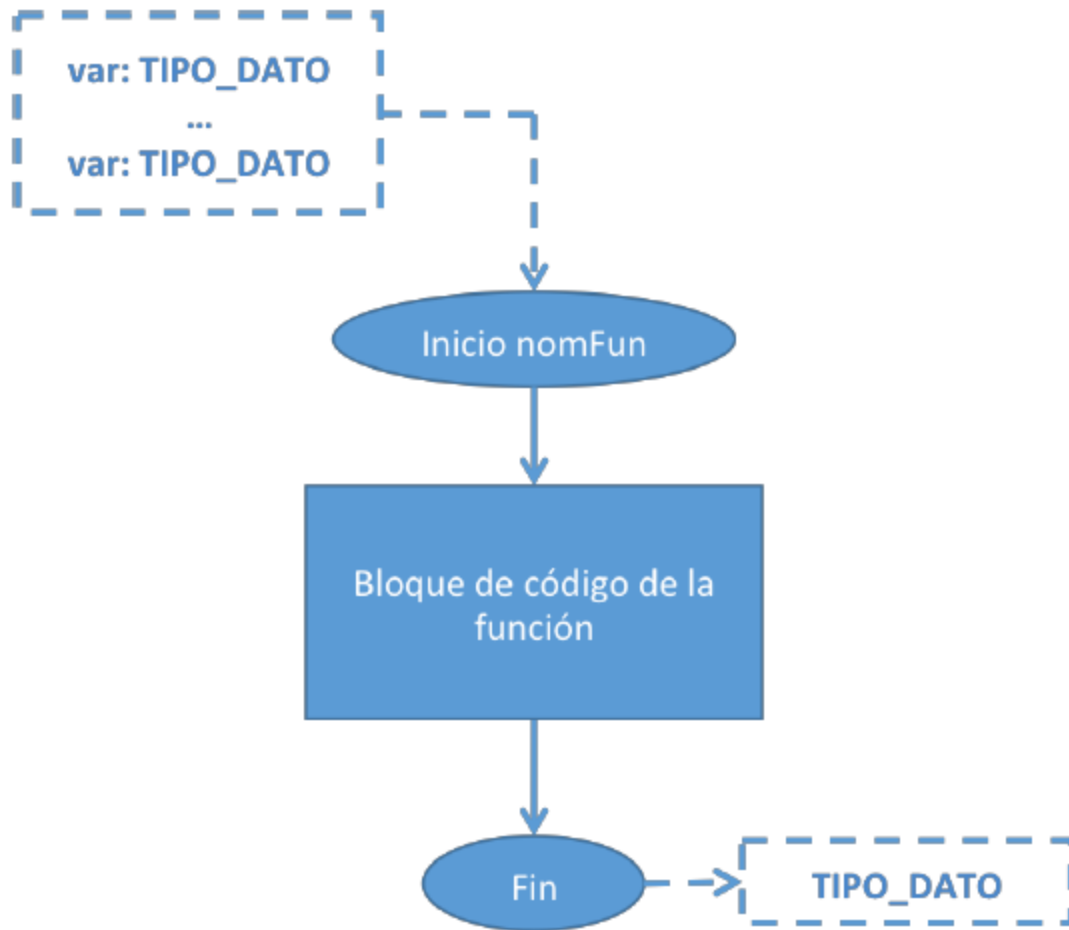
// >>> 1

// >>> 2

Funciones

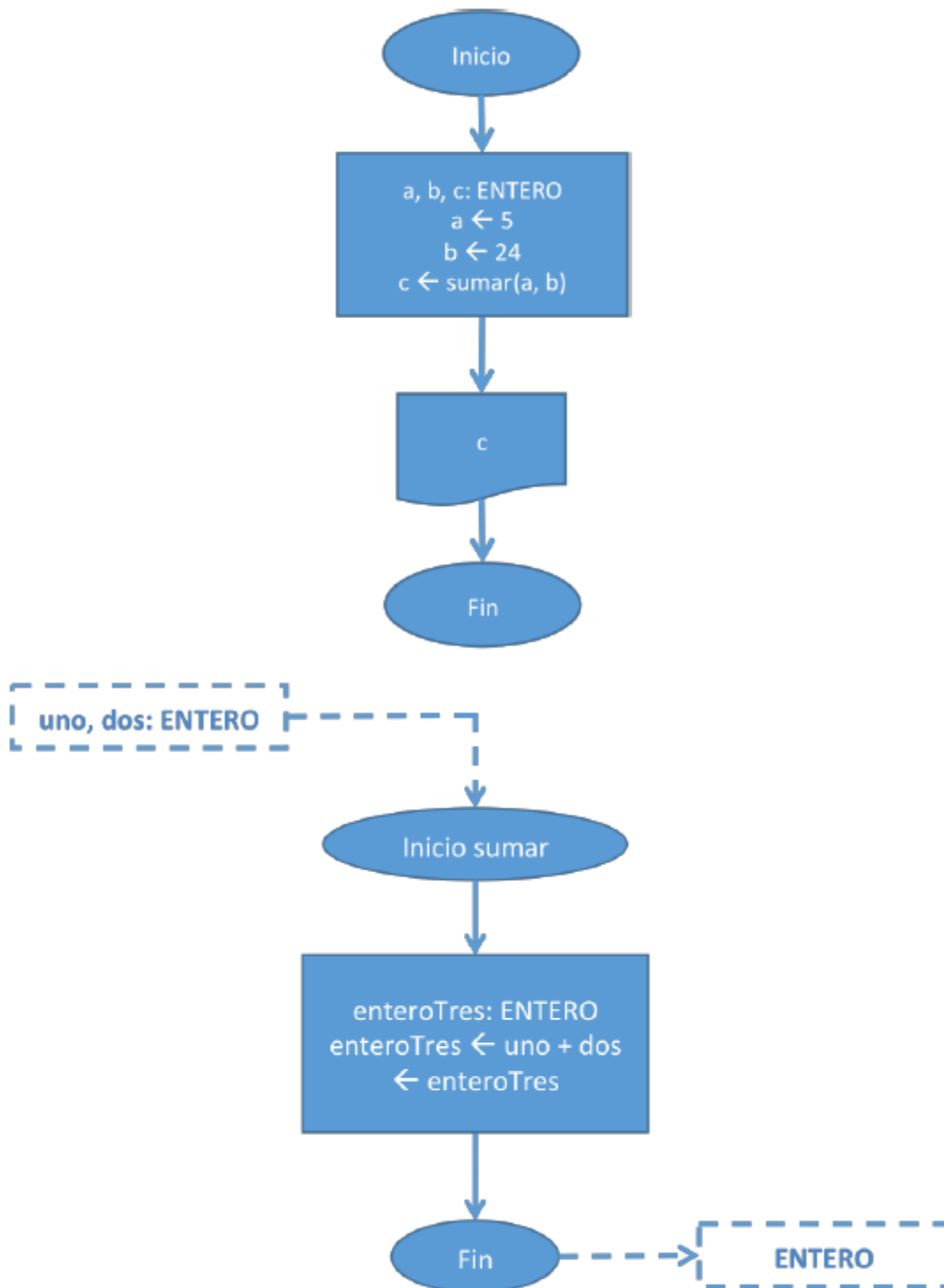
Cuando la solución de un problema es muy compleja se suele ocupar el diseño descendente (divide y vencerás). Este diseño implica la división de un problema en varios subprocesos más sencillos que juntos forman la solución completa. A estos subprocesos se les llaman módulos o funciones.

Una función está constituida por un identificador de función (nombre), de cero a n parámetros de entrada y un valor de retorno:



nomFun es el nombre con el que llama a la función. Las funciones pueden o no recibir algún parámetro (tipo de dato) como entrada, si la función recibe alguno se debe incluir en el recuadro inicial (el que apunta al nombre de la función). Todas las funciones pueden regresar un valor al final de su ejecución (un resultado) para ello se debe definir el dominio del conjunto de salida (tipo de dato).

Ejemplo



// >>> 29

Descripción

La primera función que se ejecuta es 'principal', ahí se crean las variables (uno y dos) y, posteriormente, se manda llamar a la función 'sumar'. La función 'sumar' recibe como

parámetros dos valores enteros y devuelve como resultado un valor de tipo entero, que es la suma de los valores que se enviaron como parámetro.

Para la función 'principal' los pasos que realiza la función 'sumar' son transparentes, es decir, solo manda a llamar a la función y espera el parámetro de retorno.

La siguiente figura permite analizar la función a través del tiempo. El algoritmo inicia con la función principal, dentro de esta función se hace una llamada a una función externa (sumar). Sumar realiza su proceso (ejecuta su algoritmo) y devuelve un valor a la función principal, la cual sigue su flujo hasta que su estructura secuencial llega a su fin.

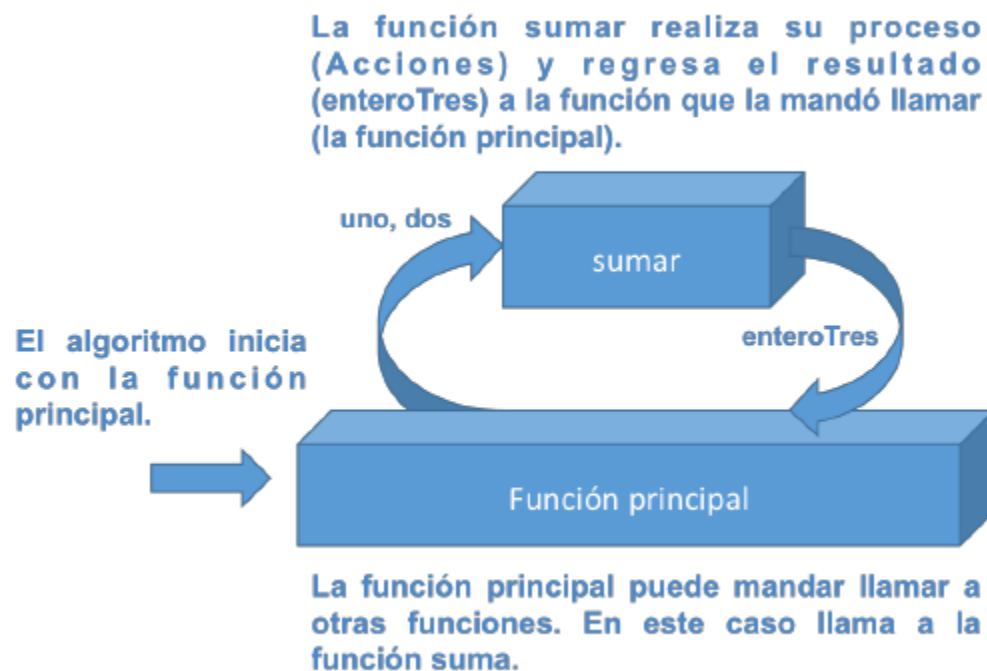


Diagrama 1

Realizar un diagrama de flujo que determine el color del semáforo COVID en base a una muestra de 100 individuos:

- Si hay más de 80 individuos con COVID el color del semáforo es rojo
- Si hay de 51 a 80 individuos con COVID el color del semáforo es naranja
- Si hay de 1 a 50 individuos con COVID el color del semáforo es amarillo
- Si no hay individuos con COVID el color del semáforo es verde

El siguiente diagrama se hizo con el Flowgorithm y representa al flujo de un programa el cual evalúa de una forma sencilla el color del semáforo COVID en función del número que se ingrese como dato del total de enfermos.

Cuando el dato ingresado es NO es mayor a 0, encontramos que el programa se dirige hacia la izquierda y evalúa si el dato es MENOR a 0, si no, entonces es igual a 0 y se entiende que no hay enfermos por lo que vuelve a ir hacia la izquierda para ejecutar el comando que imprime a la pantalla como salida la leyenda "Semaforo verde"

Caso contrario, si el dato ingresado, es un entero menor a 0, entonces la evaluación se vuelve positiva y el programa sigue el camino a la derecha en donde imprime como salida una instrucción al usuario pidiéndole que ingrese un numero positivo o 0, según sea el caso, ya que los enfermos no pueden ser negativos.

De la misma manera cuando el caso de los enfermos es un numero entero positivo, el programa sigue la misma lógica de evaluar condiciones y en función de este dato existen los condicionantes para cada uno de los casos, los cuales regresaran una respuesta de acuerdo al resultado de cada evaluación.

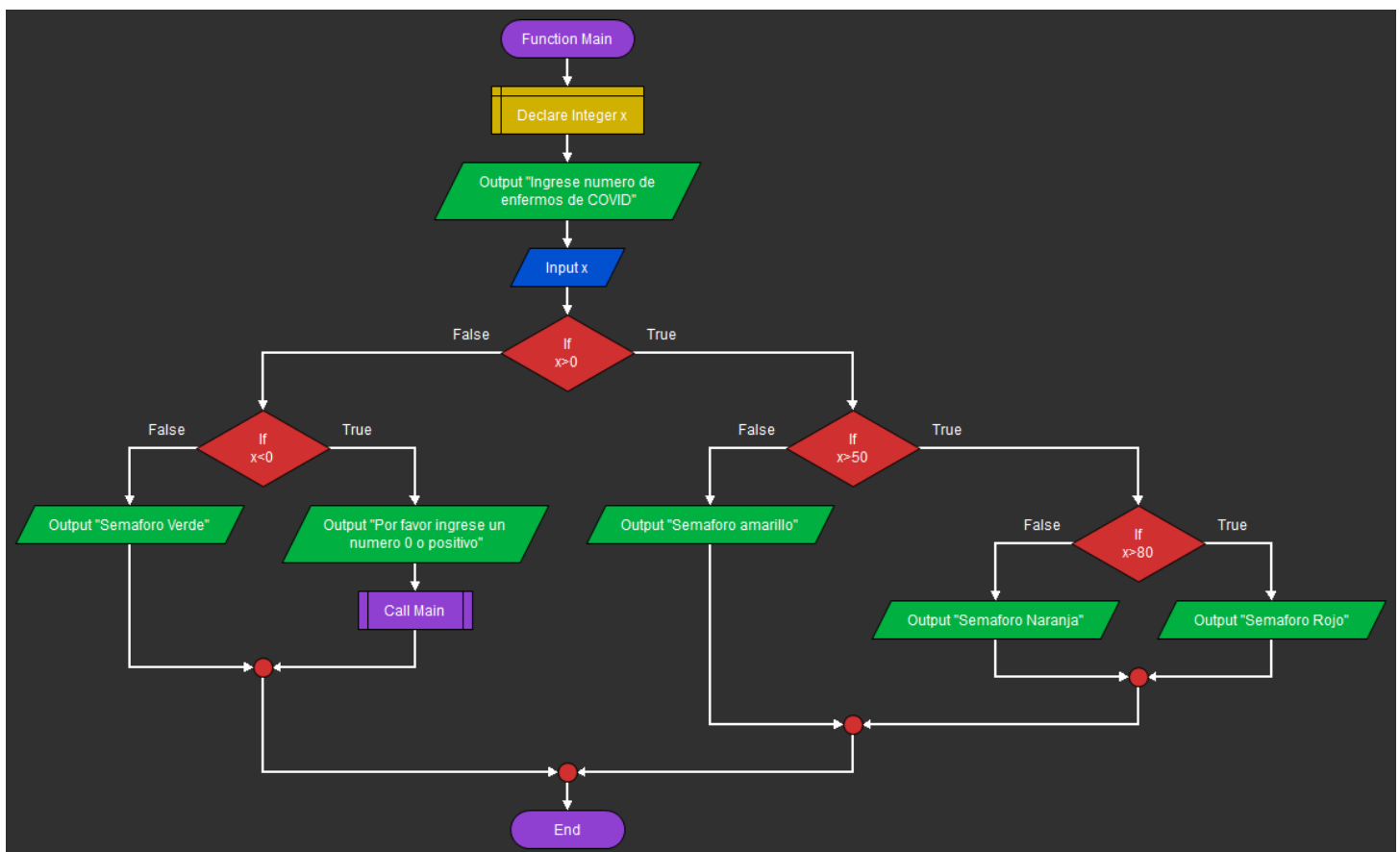


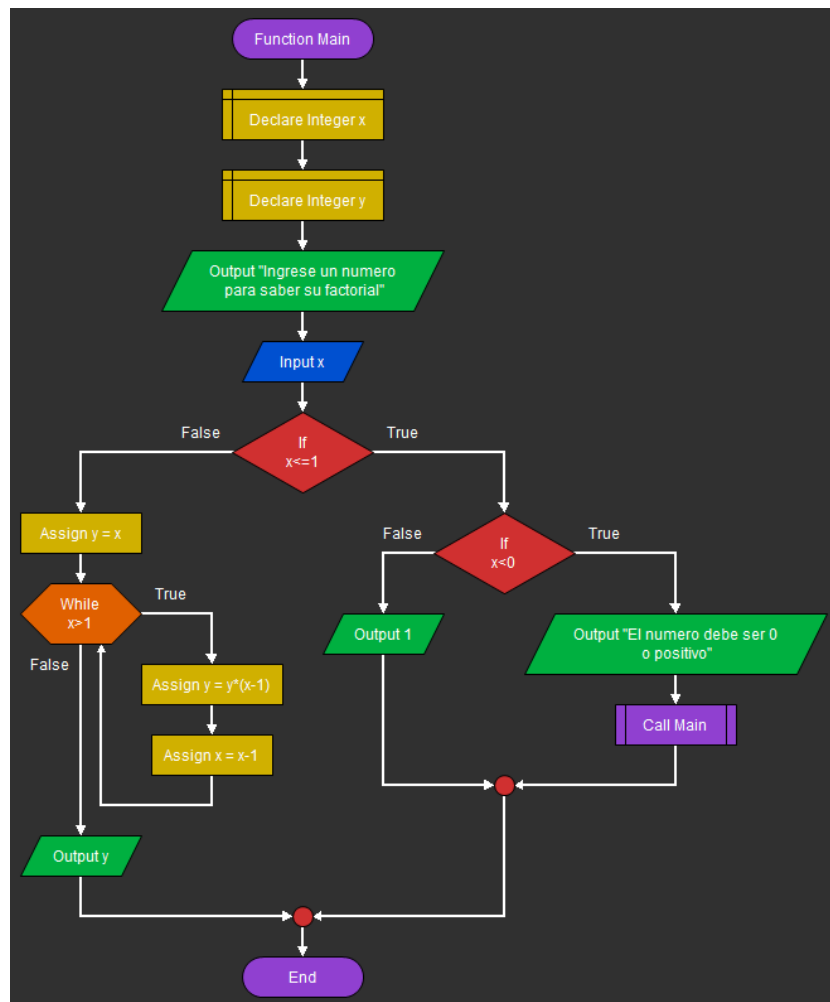
Diagrama 2

Realizar un diagrama de flujo que calcule dado un número el cálculo de su factorial:

Ejemplo:

- $1! = 1$
- $2! = 2$
- $3! = 6$
- $4! = 24$

De la misma forma que el anterior se hizo este programa el cual evalúa el factorial de un número de una manera muy simple. Se asignan dos variables enteras, si el número ingresado es 1 o 0, el programa responde que el factorial de ese número es el número 1, si el número ingresado es mayor, el programa asigna a las dos variables el número ingresado la variable y y multiplica el número de y por $x - 1$, mientras que la variable x va disminuyendo 1 en cada iteración. Cuando x llega a 1, el número de y se imprime en la pantalla.



Conclusiones:

En esta práctica se aprendió a graficar y a elaborar diagramas de flujo como uno de los primeros pasos para el desarrollo de nuevo software que forman parte de la planeación inicial del proyecto y que de igual forma corresponde a la creación del algoritmo principal.

Entendimos que el diagrama de flujo es una serie de pasos que comprende un proceso en su totalidad, es un camino que va siguiendo la información y que va transformándose de acuerdo a las condiciones a las que está sometida esa información.

Aprendimos la simbología que se usa para la creación de diagramas de flujo y también a determinar cuáles son los pasos que ira siguiendo la información dada en el desarrollo del sistema.

Referencias:

- Facultad de ingeniería – UNAM, Manual de prácticas del laboratorio de Fundamentos de programación: http://odin.fi-b.unam.mx/salac/practicasp/MADO-17_FP.pdf