|  |
| --- |
| Laboratorio de Computación  Salas A y B |
|  |

|  |  |
| --- | --- |
| Profesor: | García Morales Karina |
| Asignatura: | Fundamentos de Programación |
| Grupo: | 1121 |
| No de Práctica(s): | 04 |
| Integrante(s): | Valle Olivas Guillermo |
|  |  |
|  |  |
|  |  |
| No. de Equipo de cómputo empleado: |  |
| Semestre: | 2019-1 |
| Fecha de entrega: |  |
| Observaciones: |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Diagramas de flujo**

**Objetivo:**

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

**Actividades:**

• Elaborar un diagrama de flujo que represente la solución algorítmica de un problema, en el cual requiera el uso de la estructura de control condicional.

• Elaborar la representación gráfica de la solución de un problema, a través de un diagrama de flujo, en el cual requiera el uso de la estructura de control iterativa.

**Introducción**

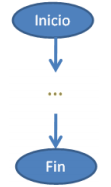
Un diagrama de flujo es la representación gráfica de un proceso, es decir, muestra gráficamente el flujo de acciones a seguir para cumplir con una tarea específica.

Dentro de las ciencias de la computación, un diagrama de flujo es la representación gráfica de un algoritmo. La correcta construcción de estos diagramas es fundamental para la etapa de codificación, ya que, a partir del diagrama de flujo es posible codificar un programa en algún lenguaje de programación.

**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.

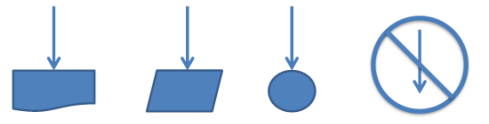
1.- Todo diagrama de flujo debe tener un inicio y un fin.



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.



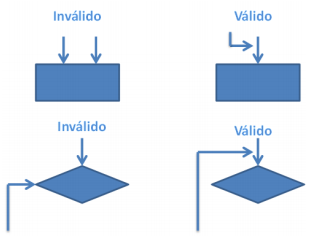
4.- El diagrama debe ser construido de arriba hacia abajo (top-down) y de izquierda a derecha (left to right).

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.

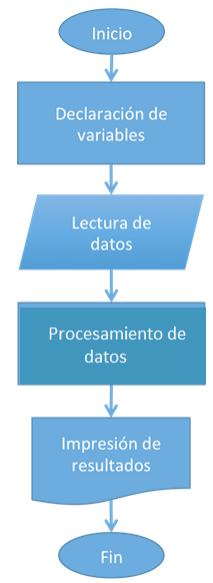


9.- Notación de camello. Para nombrar variables y nombres de funciones se debe hacer uso de la notación de camello. En la notación camello (llamada así porque parecen las jorobas de un camello) los nombres de cada palabra empiezan con mayúscula y el resto se escribe con minúsculas. No se usan puntos ni guiones para separar las palabras. Además, para saber el tipo de variable se utiliza un prefijo.

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 | la que sjo. |

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

La estructura secuencial es la que sigue una serie de pasos, uno tras otro de forma ordenada para llegar a un fin determinado.

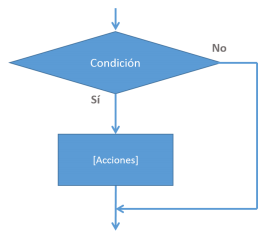
Ejemplo:



Estructuras de control condicionales (o selectivas).

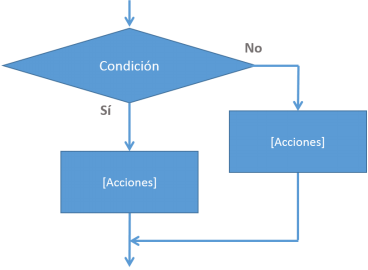
Es la estructura que indica qué camino debe tomar el flujo, es una estructura excluyente por lo cual o ejecuta una acción o ejecuta otra.

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



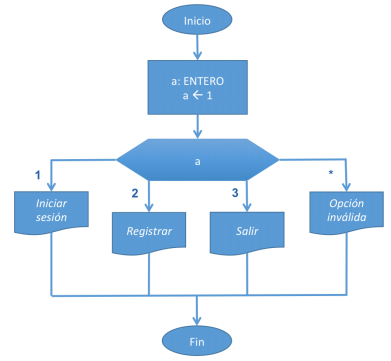
Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque [Acciones]. Si no se cumple la condición, se continúa con el flujo normal del programa.

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.

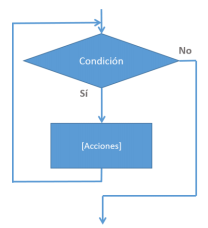
La estructura condicional **SELECCIONAR-CASO** valida el valor de la variable que está en el hexágono y comprueba si es igual al valor que está definido en cada caso (líneas que emanan del hexágono). Si la variable no tiene el valor de algún caso se va a la instrucción por defecto



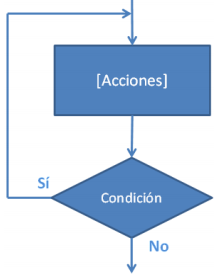
Estructuras de control iterativas o repetitivas

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.



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

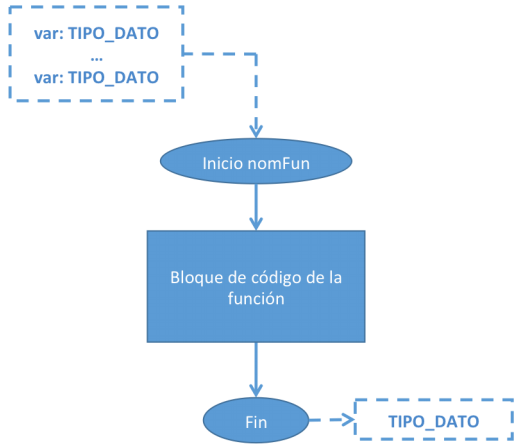


Si la condición se cumple vuelve a ejecutar las instrucciones de la estructura, de lo contrario rompe el ciclo y sigue el flujo del algoritmo. Esta estructura asegura que, por lo menos, se ejecuta una vez el bloque de la estructura, ya que primero ejecuta y después pregunta por la condición.

**Funciones**

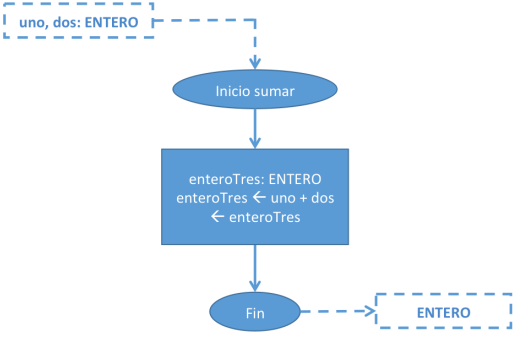
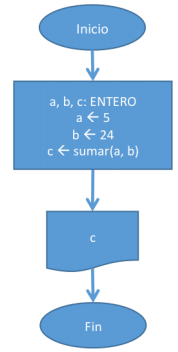
Cuando la solución de un problema es muy compleja se suele ocupar el diseño descendente (divide y vencerás). Este diseño implica la división de un problema en varios subprocesos más sencillos que juntos forman la solución completa. A estos subprocesos se les llaman módulos o funciones.

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:

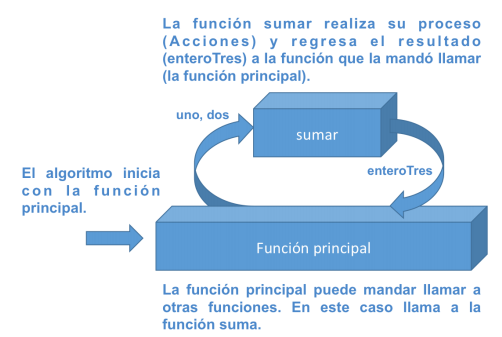


Descripción

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.

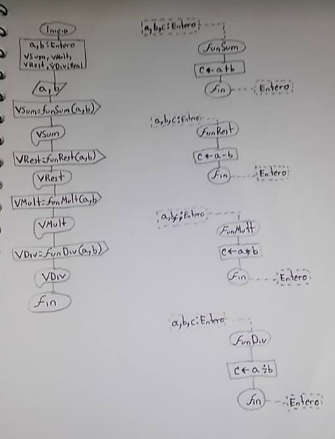
Para la función 'principal' los pasos que realiza la función 'sumar' son transparentes, es decir, solo manda a llamar a la función y espera el parámetro de retorno.

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.

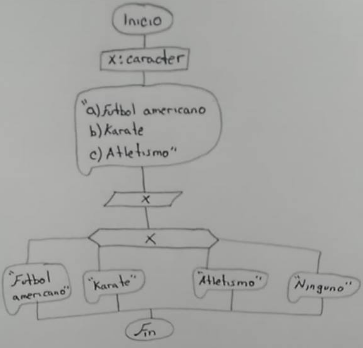


**Ejercicios de tarea**

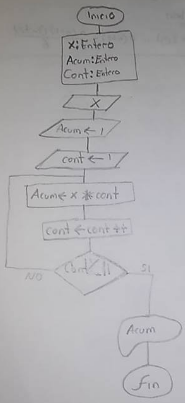
1.- Calculadora para dos variables (+,-,\*,/)



2.-Menú de deportes (3 deportes)



3.- Tablas de multiplicar del 1 al 10, el usuario proporciona el valor a calcular (Para o Mientras)



**Conclusiones**

Con esta práctica ya me resultó más fácil realizar diagramas de flujo, aunque aún me queda algo de duda sobre la condicional simple, me cuesta algo de trabajo darme cuenta como acomodar la condicional para que así el flujo del diagrama no se vea comprometido, dejando de lado eso el resto me pareció algo sencillo

**Bibliografía**

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

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

♣ https://firebird21.wordpress.com/2013/10/03/la-notacion-camello/