Diagramas de Flujo y Pseudocódigo

1. Lenguajes de programación.

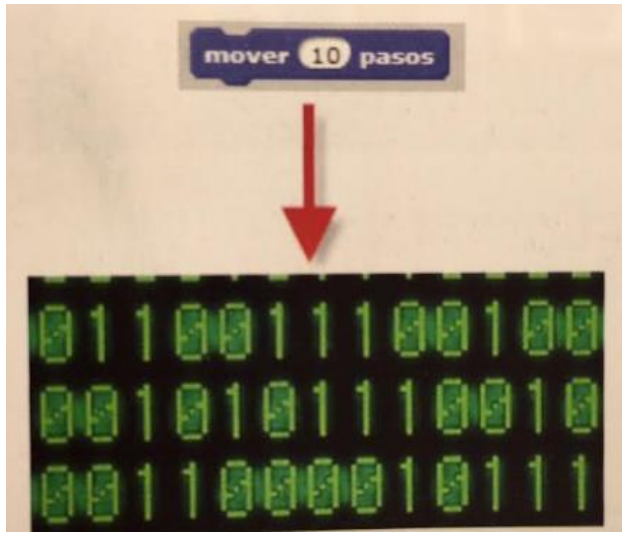
Lenguaje de programación: Es un lenguaje que sirve para describir un conjunto de acciones que deben ser ejecutadas por las máquinas, es decir, son las instrucciones que damos a los equipos. Por medio de ordenes construimos **programas**.

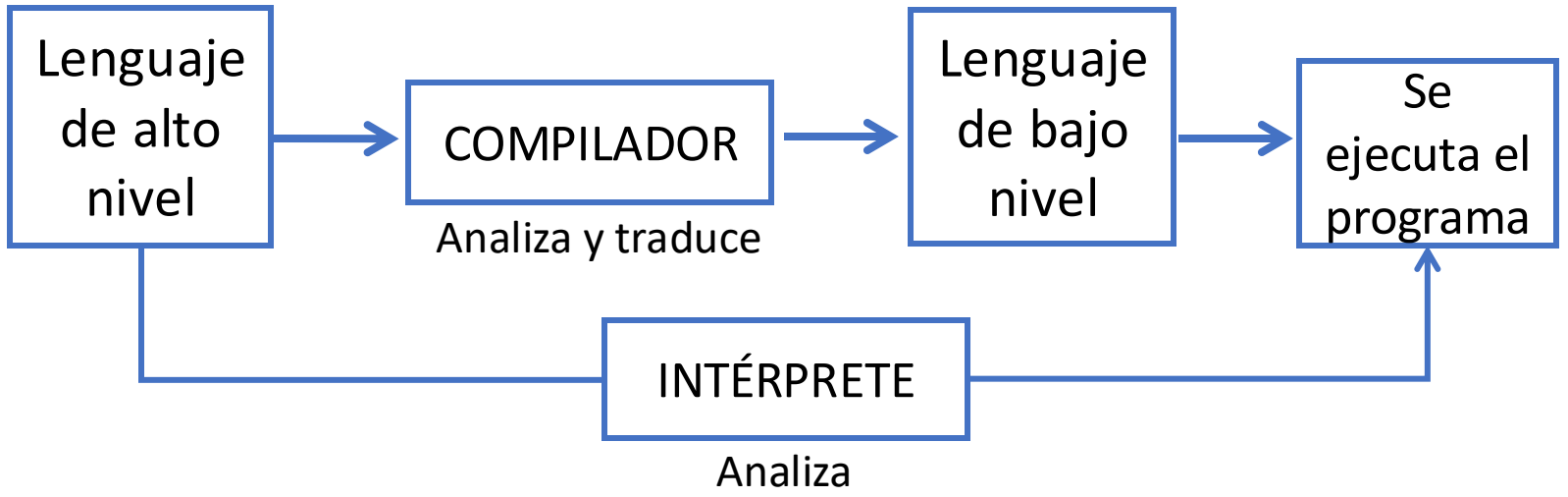
Lenguaje máquina: es el lenguaje empleado por los equipos para comunicarse entre sí. Consiste en un código binario (ceros y unos) → **Lenguaje de bajo nivel.**

Lenguaje de alto nivel → lenguaje escrito por una persona para crear un programa. Este lenguaje tiene que ser transformado en código máquina para que el ordenador pueda interpretarlo. Ej.: Java, PHP, Python, MATLAB, BASIC, etc.

Compilador: analiza el programa y lo traduce a código máquina. Ej.: C++ Builder, BlueJ, etc.
Intérprete: analiza el programa y lo ejecuta, sin traducirlo a código máquina. Ej.: Motor Zend, Ruby MRI, etc.

1. Lenguajes de programación.





ALGORITMOS

- **Un algoritmo** es una secuencia de pasos lógicos necesarios para llevar a cabo una tarea especifica, como la solución de un problema.

Las **características fundamentales** que debe cumplir todo algoritmo son:

- Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.

- Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.

- Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea debe de tener un número finito de pasos.

- La definición de un algoritmo debe describir tres partes: Entrada, Proceso y Salida.

Algoritmo para abrir una puerta

1. Verificar si la puerta está abierta.
    1. Si no está abierta:
        1. Pararse al frente
        2. Girar la manilla
        3. Si la puerta tiene llave
            1. Buscar la llave
            2. Colocar la llave
            3. Volver al paso 2
    2. Si está abierta
        1. Entrar
    3. Fin

Algoritmo para tomar una ducha

1. Abrir agua caliente
2. Probar la temperatura con la mano
3. Si el agua está muy caliente entonces
    1. Abrir el agua helada
    2. Se vuelve a ver si el agua está caliente (punto 3)
4. Si el agua está muy fría
    1. Cerrar el agua fría
    2. Se vuelve a ver si el agua está fría (punto 4)
5. Si el agua tiene una temperatura agradable
    1. Ingresar a la ducha
6. Caso contrario, se vuelve a comprobar todo de nuevo (punto 2)
7. FIN

Forma General de un Algoritmo

```
Proceso MiProceso
    Instrucción 1;
    Instrucción 2;
    .
    .
    Instrucción N;
FinProceso
```

Algoritmo 1 para cruzar la calle

1. Me detengo en la esquina
2. Miro al semáforo
3. SI (semáforo en verde)
    1. Cruzo
4. Caso contrario
    1. Me detengo
    2. Vuelvo al paso 2
5. FIN

Algoritmo 2 para cruzar la calle

1. Me detengo en la esquina
2. Miro al semáforo
3. SI (semáforo en verde) **y (no vienen autos)**
    1. Cruzo
4. Caso contrario
    1. Me detengo
    2. Vuelvo al paso 2
5. FIN

¿Qué algoritmo es más preciso?

- Las dos herramientas mas utilizadas comunmente para describir algoritmos son:
- **Diagramas de Flujo:** son representaciones gráficas de secuencias de pasos a realizar. Cada operacion se representa mediante un símbolo normalizado el Instituto Norteamericano de Normalizacion (ANSI - American National Standars Institute). Las líneas de flujo indican el orden de ejecución.
- Los diagramas de flujo suelen ser usados solo para representar algoritmos pequeños, ya que abarcan mucho espacio.

•**Pseudocódigos:** describen un algoritmo de forma similar a un lenguaje de programacióon pero sin su rigidez, de forma más parecida al lenguaje natural. Presentan la ventaja de ser más compactos que los diagramas de flujo, más fáciles de escribir para las instrucciones complejas y más fáciles de transferir a un lenguaje de programación. El pseudocódigo no está regido por ningún estándar.

•Algunas palabras usadas son LEER/IMPRIMIR para representar las acciones de lectura de datos y salida de datos.

•*Calcular una 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**

Diagrama de Flujo

- Un algoritmo describe una secuencia de pasos escritos para realizar un tarea.
- El **Diagrama de Flujo** es su representación esquemática. **Los diagramas de flujo representan la secuencia lógica o los pasos que tenemos que dar para realizar una tarea mediante unos símbolos y dentro de ellos se describen los pasos ha realizar.**

Reglas Básicas Para la Construcción de un Diagrama de Flujo

1. Todos los símbolos han de estar conectados.

2. A un símbolo de proceso pueden llegarle varias líneas.

3. A un símbolo de decisión pueden llegarle varias líneas, pero sólo saldrán dos (Si o No, Verdadero o Falso).

4. A un símbolo de inicio nunca le llegan líneas.

5. De un símbolo de fin no parte ninguna línea.

PASOS PARA PLANTEAR LA SOLUCIÓN A UN PROBLEMA (METODOLOGÍA)

1.- Análisis del problema.

2.- Identificar las entradas, procesos y salidas del problema. Declaración de variables.

3.- Diseño del Algoritmo: Describe la secuencia ordenada de los pasos, sin ambigüedad, es decir, siendo preciso y veraz en la búsqueda de la solución al problema.

3.1. Creación de Diagrama de Flujo

4.- Codificación del Algoritmo: Es la expresión en un lenguaje de programación de los pasos definidos en el algoritmo.

5.- Ejecución y validación del programa por el computador.

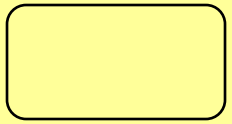
6.- Documentación.

7.- Mantenimiento.

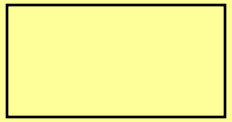
2. Algoritmos y diagramas de flujo.

Algoritmo: Es una serie de instrucciones o pasos ordenados que nos llevan a resolver un problema o hacer una actividad.

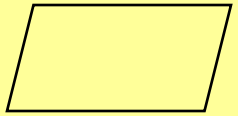
Estas instrucciones se pueden representar mediante **diagramas de flujo.**



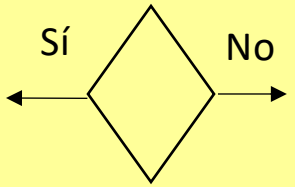
Principio o fin de un proceso



Proceso

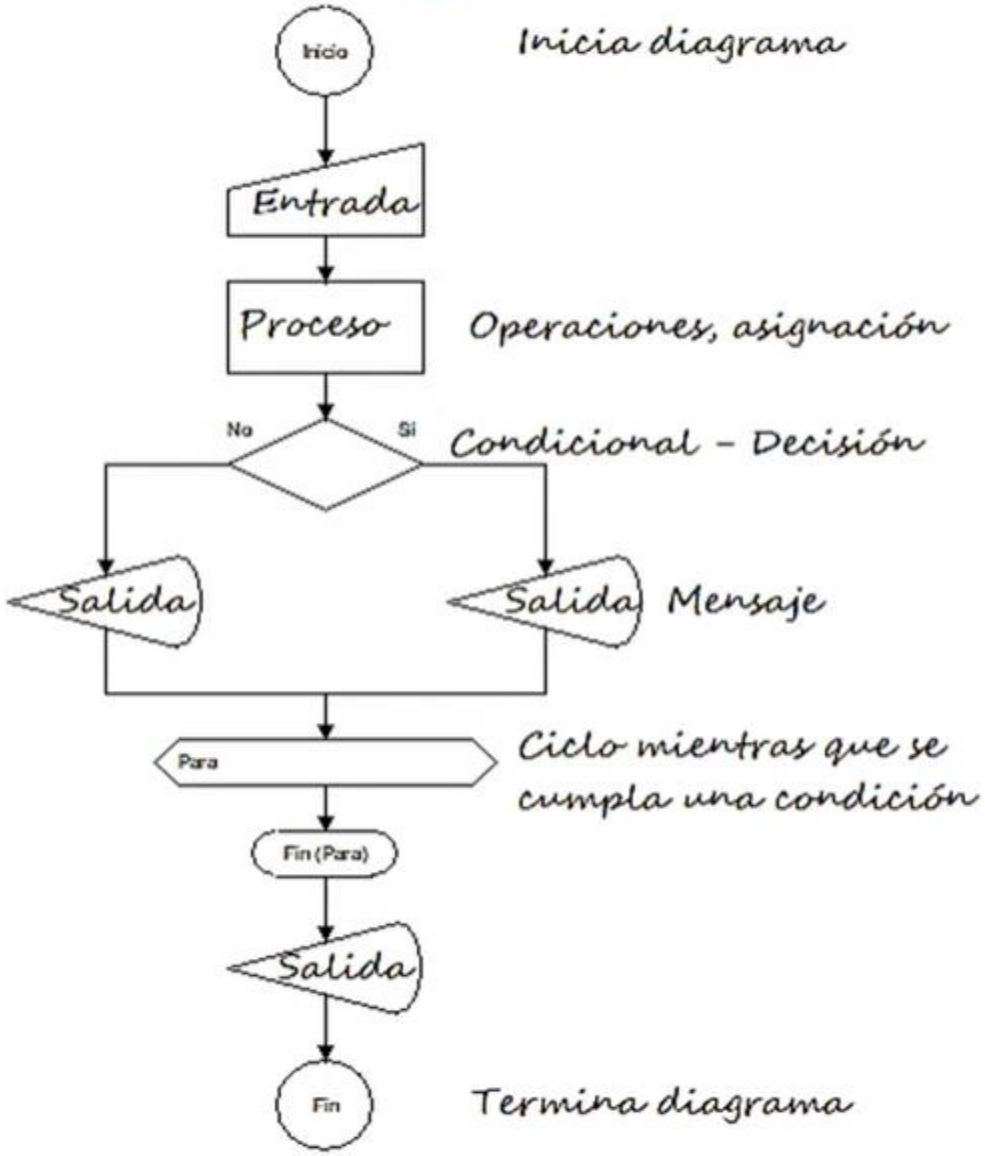


Entrada o salida de datos

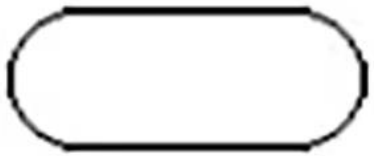


Toma de decisiones

Simbología DFD 1.1

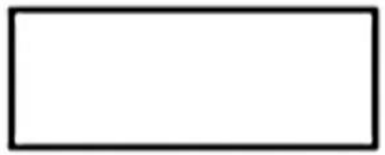


Simbología de los Diagramas de Flujo



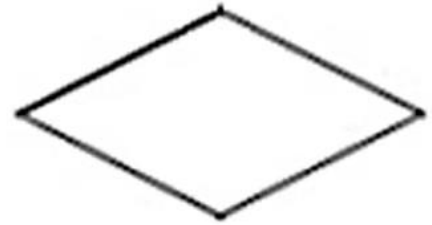
Inicio/Final
Se utiliza para indicar el inicio y el final de un diagrama; del Inicio sólo puede salir una línea de flujo y al Final sólo debe llegar una línea.

Entrada General
Entrada/Salida de datos en General (en esta guía, solo la usaremos para la Entrada).

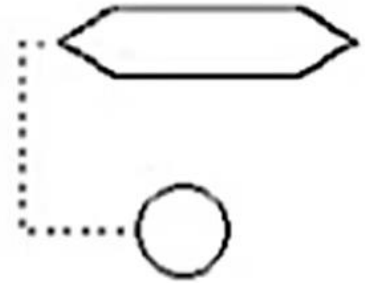


Acción/Proceso General
Indica una acción o instrucción general que debe realizar el computador (cambios de valores de variables, asignaciones, operaciones aritméticas, etc).

Conector.
Indica el flujo que segurirá el diagrama de Flujo



Decisión
Indica la comparación de dos datos y dependiendo del resultado lógico (falso o verdadero) se toma la decisión de seguir un camino del diagrama u otro.



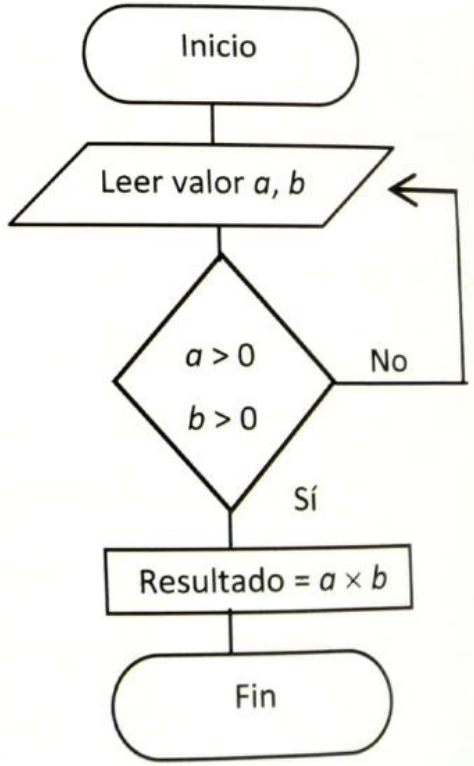
Iteración
Indica que una instrucción o grupo de instrucciones deben ejecutarse varias veces.

Salida en Pantalla
Instrucción de presentación de mensajes o resultados en pantalla.

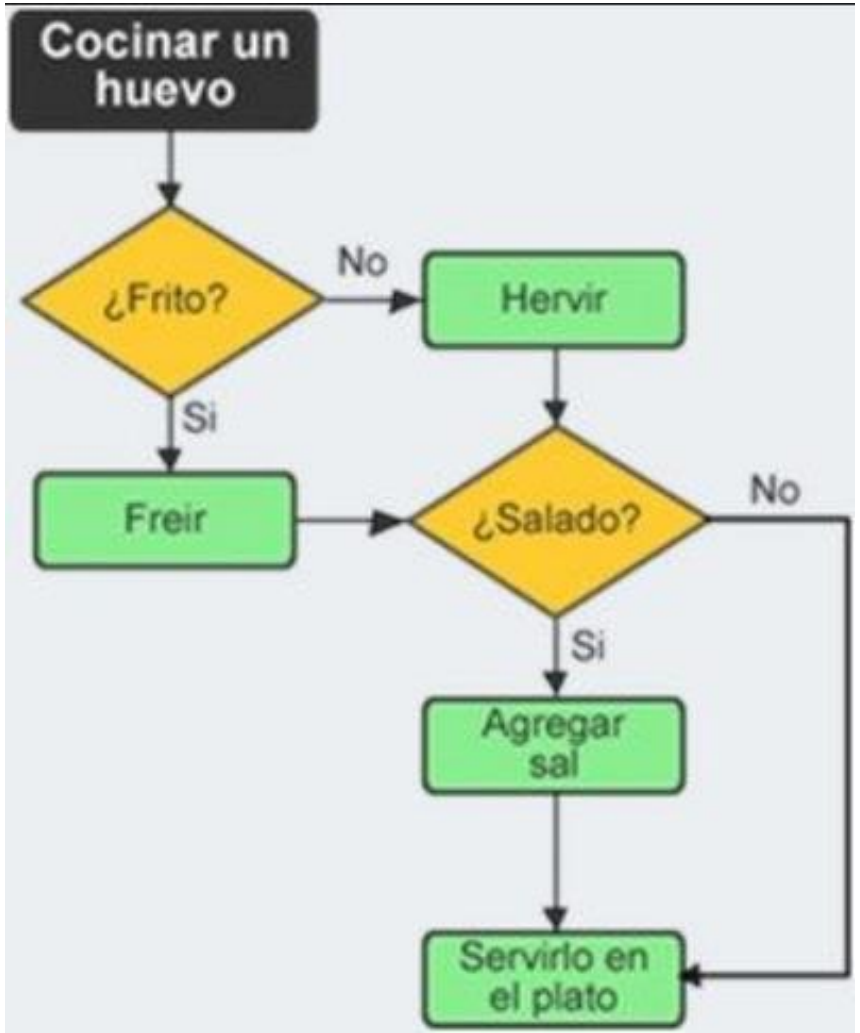
2. Algoritmos y diagramas de flujo.

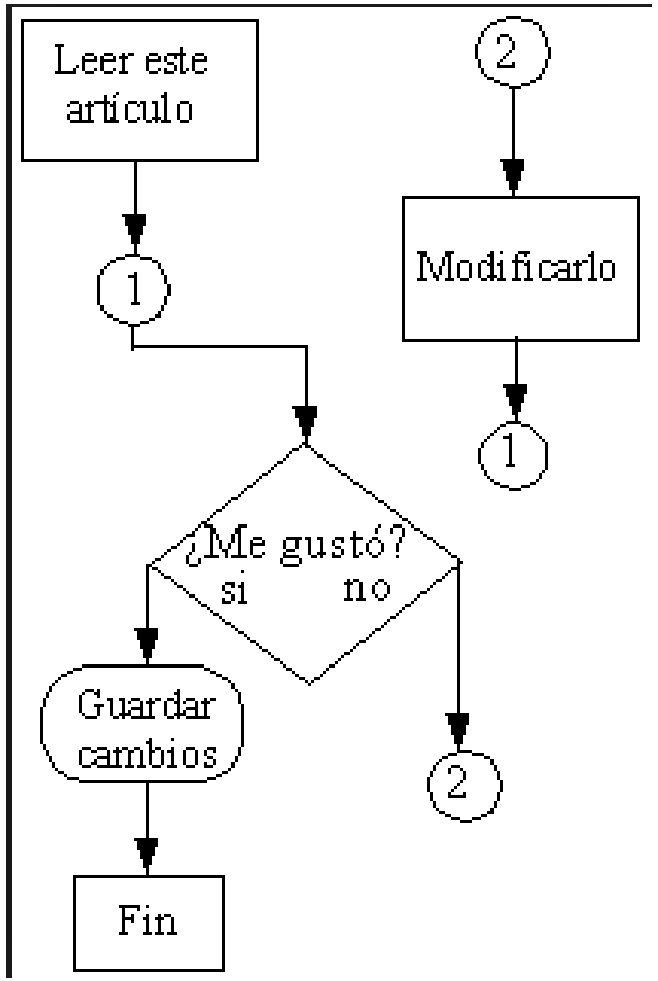
Por ejemplo, queremos multiplicar dos números *a* y *b* solamente si ambos son mayores que 0.

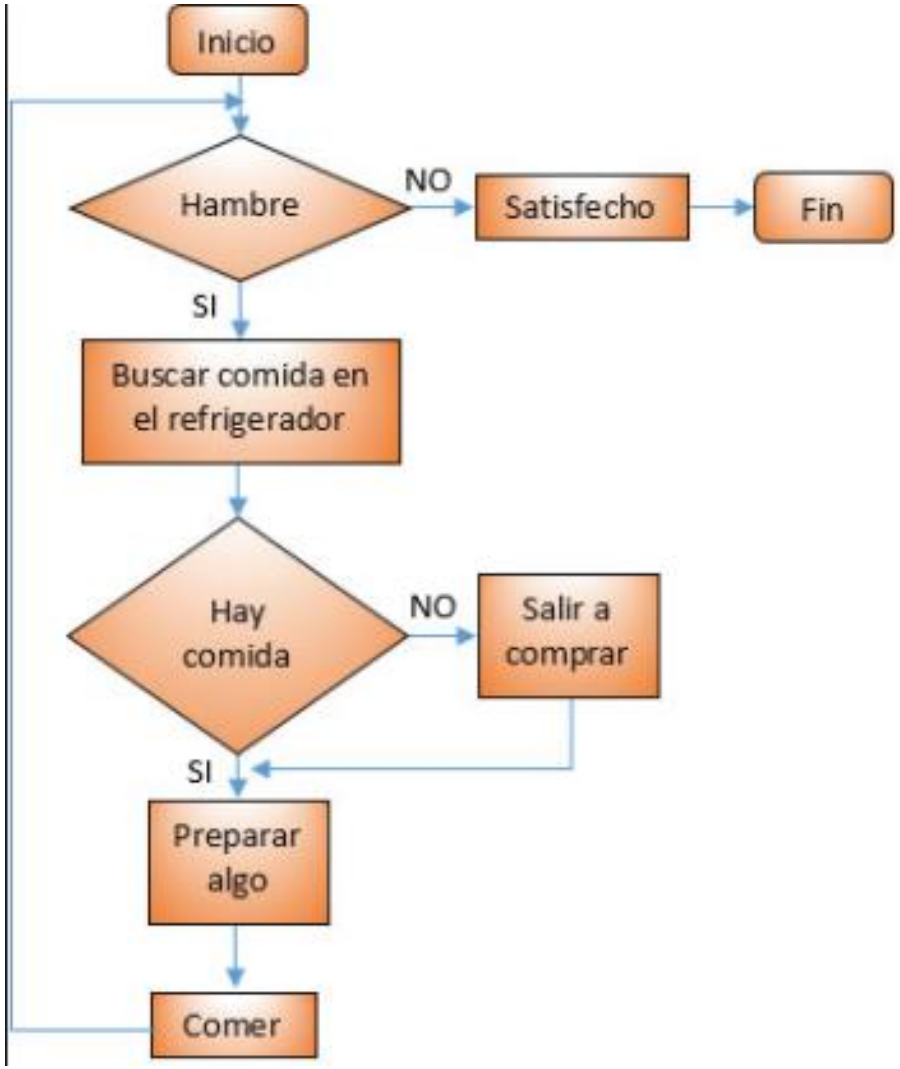
1. Iniciamos el programa.
2. Introducimos el valor de *a* y *b*.
3. Comprobamos que ambos son mayores que 0.
4. Si no lo son, volvemos a empezar.
5. Si lo son, realizamos la operación.
6. Finalizamos el programa.



2. Algoritmos y diagramas de flujo.

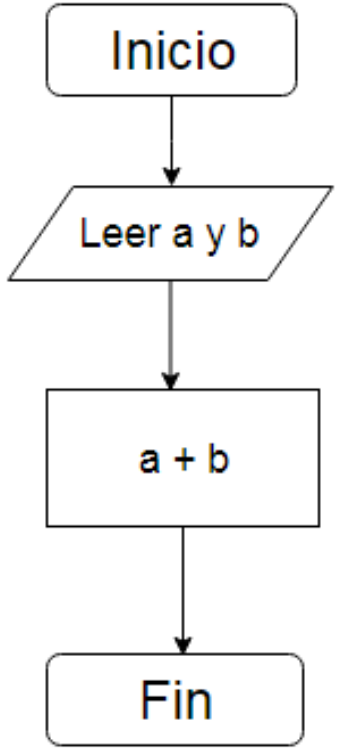






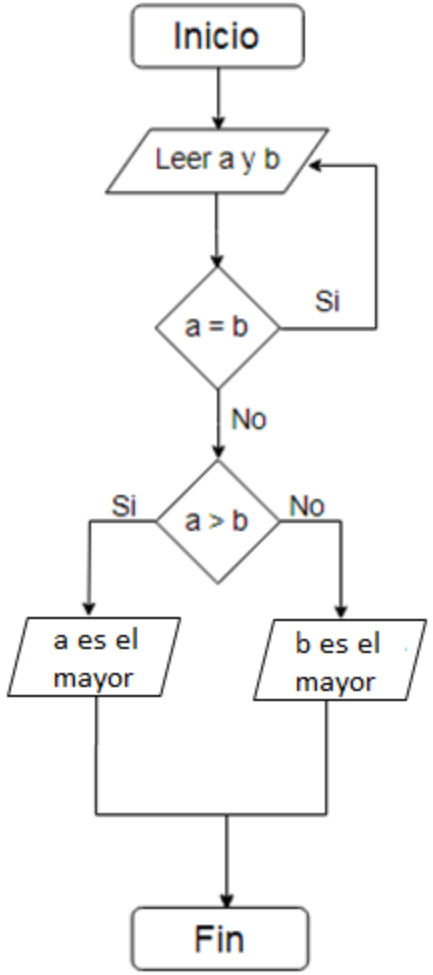
2. Algoritmos y diagramas de flujo.

1. Hacer el diagrama de flujo para sumar dos números leídos por teclado y escribir el resultado.



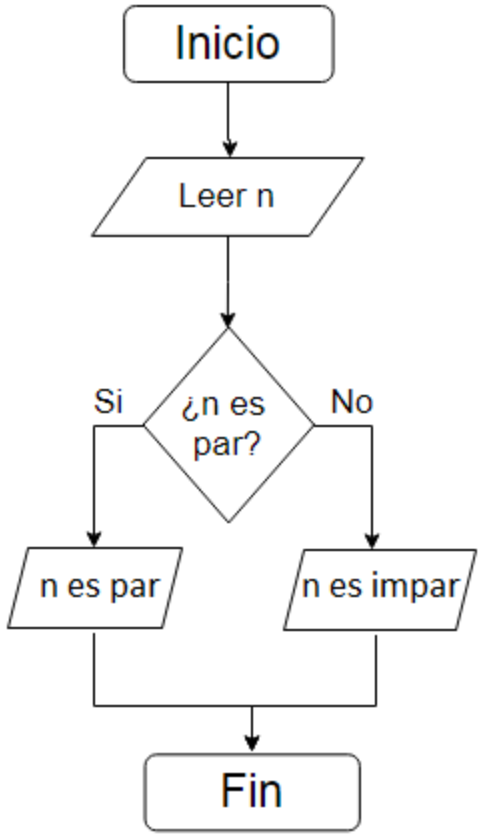
2. Algoritmos y diagramas de flujo.

2. Hacer un diagrama de flujo que permita leer 2 números diferentes y nos diga cual es el mayor de los 2 números.



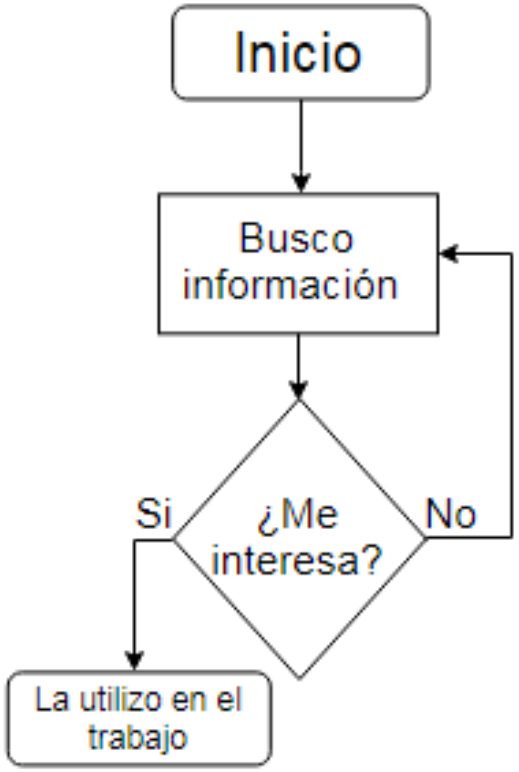
2. Algoritmos y diagramas de flujo.

3) Hacer un algoritmo que permita leer un número y decir si es par o impar.



2. Algoritmos y diagramas de flujo.

4) Hacer un diagrama de flujo de la siguiente situación: “Tengo que hacer un trabajo para clase, decido buscar información por internet, si encuentro información que no me sirve sigo buscando. Si encuentro información que me sirve entonces la utilizo en mi trabajo”.



2. Algoritmos y diagramas de flujo.

Práctica II: Diagramas de flujo.

1. Hacer un diagrama para dividir dos números introducidos por teclado y escribir el resultado.
2. Queremos multiplicar dos números, si *a* es mayor que *b,* sino es así que salga un mensaje: “prueba de nuevo”.
3. Queremos restar dos números si *a* es mayor que *b*, y sumarlos si b es mayor que *a*.
4. Hacer un diagrama de flujo para dividir dos números siempre y cuando los dos sean pares (múltiplos de 2).
5. Representa un diagrama de flujo de la siguiente situación: “He salido a montar en bici y de repente pincho. Miro si tengo parches para poner en el pinchazo, si es así lo reparo, sino tendré que ir a comprar una cámara nueva a la tienda”.

2. Algoritmos y diagramas de flujo.

6. Representa en un diagrama de flujo las situaciones descritas a continuación:

a) Voy a encender la lámpara y no funciona. Tengo que comprobar si está desenchufada, si la bombilla está rota o si pasa alguna otra cosa. Si esta desenchufada, la enchufo. Si la bombilla está rota, la cambio. Si no ocurre nada de eso, la llevo a arreglar.

2. Algoritmos y diagramas de flujo.

6. Representa en un diagrama de flujo las situaciones descritas a continuación:

b) Quiero sacar buena nota en programación. Voy a probar a atender en clase y hacer todos los deberes. Si saco buena nota en el primer examen, sigo igual. Si no lo consigo, tengo que estudiar más horas y preguntar las dudas a mi profesor.

Pseudocódigo

Es empleado para representar la solución de un algoritmo empleando lenguaje natural escrito estableciendo la secuencia de pasos sin imprecisiones y de manera clara.

Ejemplo:

```
Proceso
        Leer lista_de_variables;
        variable<-expresion;
        Escribir lista_de_expresiones;
FinProceso
```

Uso del Diagrama de flujo, pseudocódigo y prueba de escritorio para los tipos de estructuras

Secuenciales

Implica escribir un paso tras de otro, donde el primero que se haya escrito es el primero que se ejecutará.

Inicio
Acción1
Acción2
.
.
AcciónN
Fin

Secuenciales

Ejemplo:

Pseudocódigo

Inicio

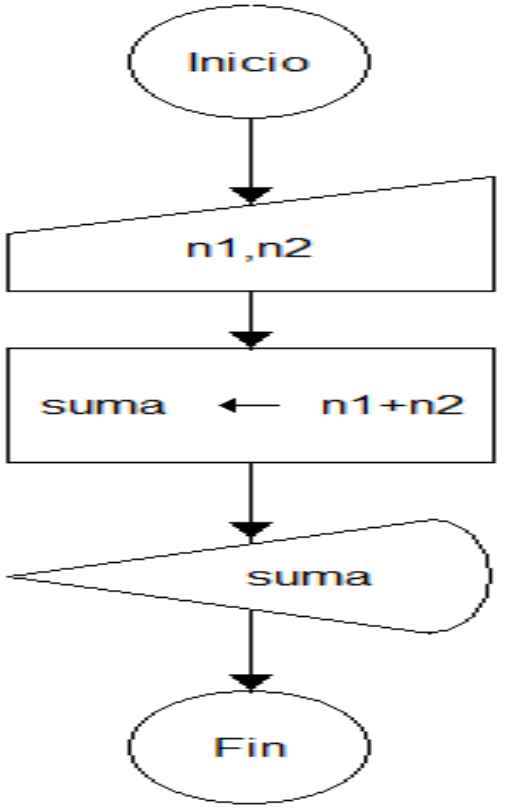
Leer N1, N2

SUMA=N1+N2

Escribir SUMA

Fin

DFD



Selectivas: Se utilizan para TOMAR DECISIONES.

- ✓ **Simples**

Lo que se hace es EVALUAR la condición, si la condición es verdadera realiza la acción, en caso contrario termina el programa.

```
Si <condición> entonces
    Acción(es)
Fin-si
```

2) Selectivas Simples

Ejemplo:

Pseudocódigo

Inicio

Leer COMPRA

Si COMPRA>1000 entonces

DESCUENTO=COMPRA*0.10

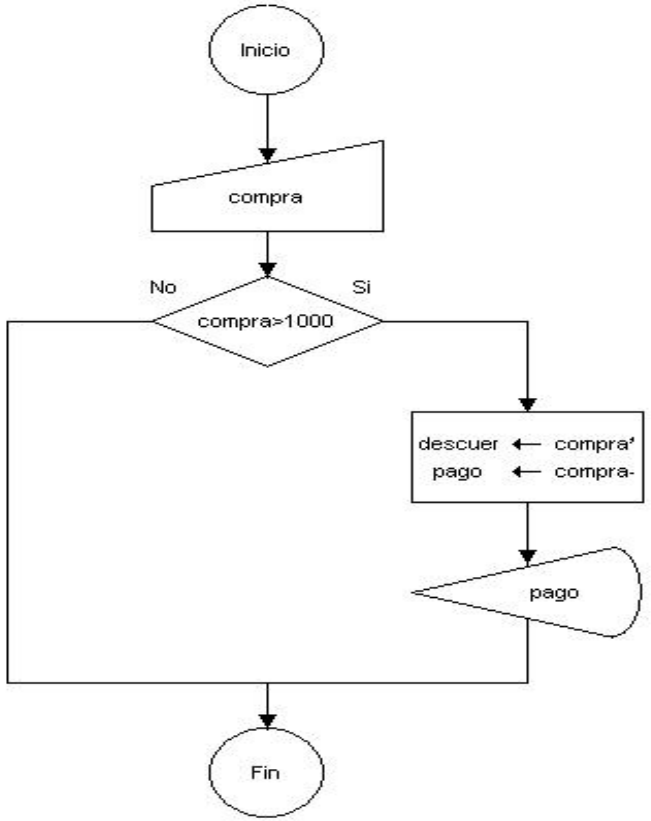
PAGO=COMPRA–DESCUENTO

Escribir PAGO

Finsi

Fin

DFD



2) **Selectivas**

- ✓ **Doble**

Luego de evaluar una condición si esta se cumple, es decir si es verdadera realiza una serie de acciones, y si esta es falsa se realiza otra serie de acciones distinta a la primera.

```
Si <condición> entonces
        Acción(es)
Sino
        Acción(es)
Finsi
```

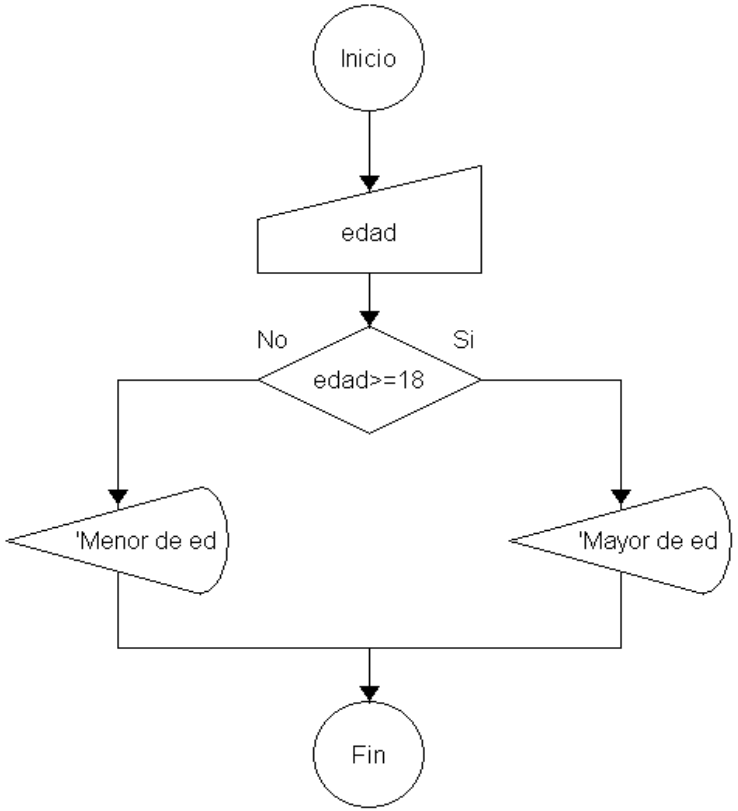
2) Selectivas Doble

Ejemplo:

Pseudocódigo

Inicio
    Leer EDAD
    Si EDAD>=18 entonces
        Escribir “Mayor de edad”
    Sino
        Escribir “Menor de edad”
    Finsi
Fin

DFD



2) Selectivas

✓ Múltiple

Se realiza a partir de anidar estructuras simples y/o dobles, de manera tal que se realicen diferentes acciones con base a varias comparaciones, así habrá tantas opciones como se requieran.

```
Si <condición> entonces
    Acción(es)
Sino
    Si <condición> entonces
            Acción(es)
    Sino
            .
            .  Varias condiciones
            .
    Finsi
Finsi
```

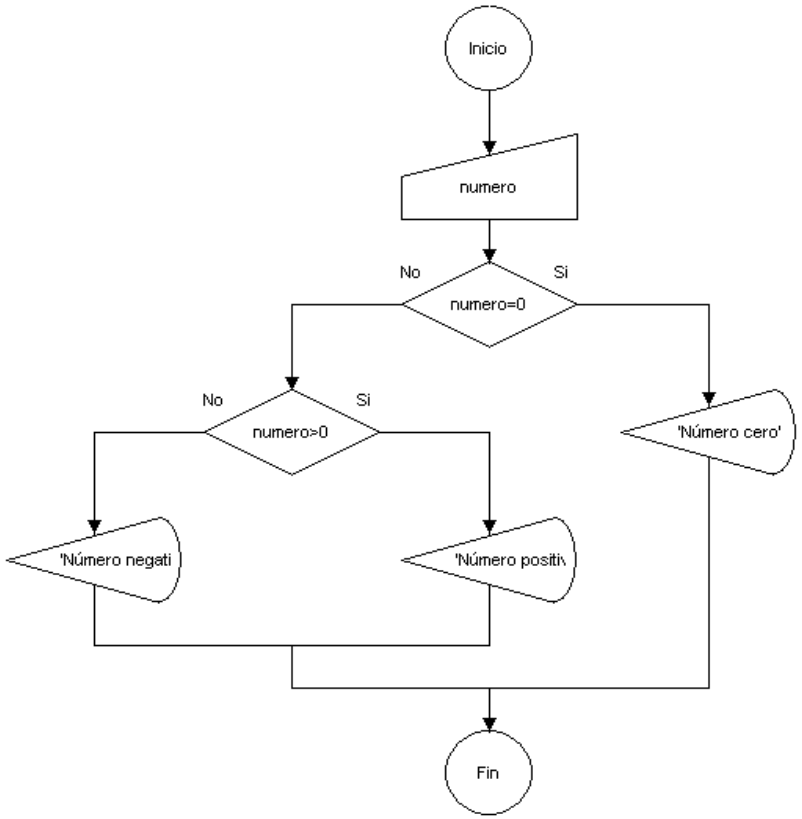
2) Selectivas Múltiple

Ejemplo:

Pseudocódigo

```
Inicio
    Leer NUMERO
    Si NUMERO=0 entonces
       Escribir “Número cero”
    Sino
       Si NUMERO>0
          Escribir “Número positivo”
       Sino
          Escribir “Número negativo”
       Finsi
    Finsi
Fin
```

DFD



Repetitivas: Este tipo de estructura se utilizan para ejecutar acciones repetidamente, esto se hace posible mediante una secuencia de instrucciones que se repiten una y otra vez y así evitamos escribir múltiples veces las mismas instrucciones.

3) Repetitiva

- ✓ **Para**

Esta estructura ejecuta los pasos de la solución del algoritmo un número definido de veces y de modo automático controla el número de iteraciones o pasos a través del cuerpo del ciclo. Para el control se utiliza un contador en el cual se va acumulando el número de veces que se ha repetido las instrucciones.

```
Hacer para V.C = LI a L.S
        Acción1
        Acción2
        .
        .
        AcciónN
Fin para
```

V.C Variable de control de ciclo
L.I Límite inferior
L.S Límite superior

3) Repetitiva Para

Ejemplo:

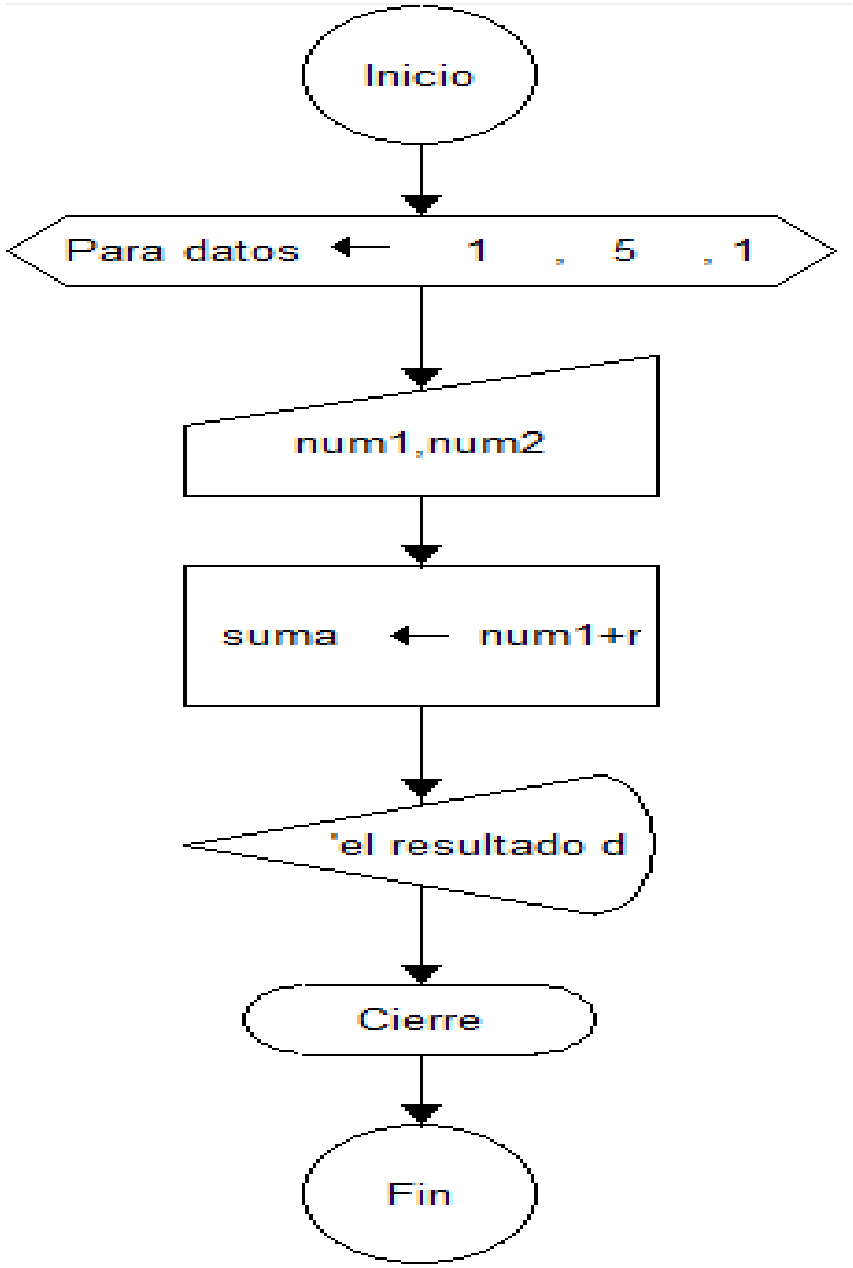
Pseudocódigo

```
Proceso sin_titulo
        Para DATOS<-1 Hasta 5 Con Paso 1 Hacer
                Leer NUM1,NUM2;
                SUMA<-NUM1+NUM2;
                Escribir "el resultado de sumar ",NUM1," + ",NUM2," = ",SUMA;
        FinPara
FinProceso
```

3) Repetitiva Para

Ejemplo:

DFD



3) Repetitiva

- ✓ **Mientras**

Este se utiliza cuando NO sabemos el número de veces que se ha de repetir un ciclo, los ciclos se determinan por una condición que se evalúa al inicio del ciclo, es decir, antes de ejecutarse todas los pasos.

```
Hacer  mientras <condición>
        Accion1
        Accion2
        .
        .
        AccionN
Fin-mientras
```

3) Repetitiva Mientras

Ejemplo

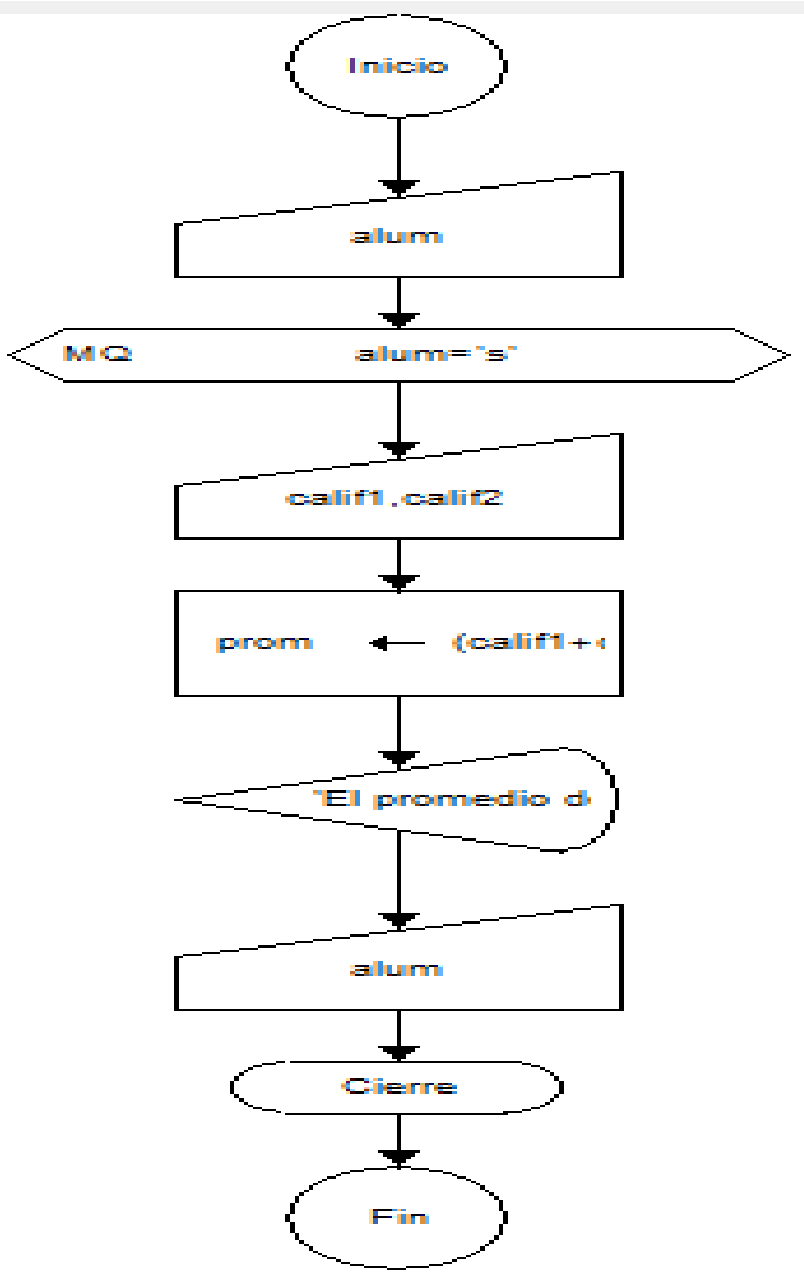
Pseudocódigo

```
Proceso sin_titulo
        Escribir "Hay alumno";
        Leer ALUM;
        Mientras ALUM="s" Hacer
                Leer CALIF1,CALIF2;
                PROM<-(CALIF1+CALIF2)/2;
                Escribir "El promedio del alumno es ",PROM;
                Escribir "Hay alumno";
                Leer ALUM;
        FinMientras
FinProceso
```

3) Repetitiva Mientras

Ejemplo

DFD



3) Repetitiva

- ✓ **Hacer – Mientras ó Repetir**

En esta estructura el ciclo se va a repetir hasta que la condición se cumpla, a diferencia de las estructuras anteriores la condición se escribe al finalizar la estructura.

```
Repetir
        Accion1
        Accion2
        .
        .
        AccionN
Hasta <condicion>
```

3) Repetitiva Hacer – Mientras ó Repetir

Ejemplo

Pseudocódigo

```
Proceso sin_titulo
        Repetir
                Leer SALARIO;
                SAL_FIN<-SALARIO*1.15;
                Escribir "El salario con aumento es",SAL_FIN;
                Escribir "hay otro empleado";
                Leer EMPLEA;
        Hasta Que EMPLEA="n"
FinProceso
```

3) Repetitiva Hacer – Mientras ó Repetir

Ejemplo

Convertido a diagrama de flujo desde PseInt