

Algoritmos

1. COMPETENCIAS EXTENDIDAS

- Conoce que es un algoritmo, sus características y estructura.
- Aplica la metodología de los algoritmos para resolver problemas.
- Aprende las diferentes alternativas para representar un algoritmo.
- Identifica las estructuras de los algoritmos y las aplica convenientemente.
- Diseña algoritmos con aplicación a su vida diaria.

1. Objetivos de aprendizaje:

El alumno comprende y aplica la metodología de los algoritmos para la solución de problemas comunes mediante el análisis y razonamiento lógico matemático que le permita elegir la mejor solución.

1. Algoritmos

1.1 Concepto e importancia

Es un conjunto de pasos lógicos y estructurados que nos permiten dar solución a un problema.

La importancia de un algoritmo radica en desarrollar un razonamiento lógico matemático a través de la comprensión y aplicación de metodologías para la resolución de problemáticas, éstas problemáticas bien pueden ser de la propia asignatura o de otras disciplinas como matemáticas, química y física que implican el seguimiento de algoritmos, apoyando así al razonamiento crítico deductivo e inductivo.

1.3 Estructura de un Algoritmo

Todo algoritmo consta de tres secciones principales:



Entrada: Es la introducción de datos para ser transformados.

Proceso: Es el conjunto de operaciones a realizar para dar solución al problema.

Salida: Son los resultados obtenidos a través del proceso.

2. Metodología para la descomposición de un algoritmo.

2.1 Conceptos

2.1.1 Definición del problema ¹

En esta etapa se deben establecer los resultados y objetivos que se desea para poder saber si los datos que se tienen son suficientes para lograr los fines propuestos.

2.1.2 Análisis ¹

Una vez definido el problema se deberán organizar los datos de tal manera que sean susceptibles de usar en los cálculos siguientes.

2.1.3 Diseño ¹

En esta etapa se proponen soluciones a los problemas a resolver, por lo que se realiza una toma de decisiones aplicando los conocimientos adquiridos y utilizando los datos existentes.

2.1.4 Verificación o prueba de escritorio ¹

Se consideran resultados previstos para datos conocidos a fin de que al probar cada una de sus partes podamos ir comprobando que el algoritmo sirve o requiere modificarse.

2.2 Análisis del Problema

2.2.1 Identificadores

Un identificador es el nombre que se le asigna a los datos de un programa (constantes, variables, tipos de datos), y que nos permiten el acceso a su contenido.

Ejemplo:

Calf1
Valor_1
Num_hrs

2.2.2 Tipos de datos

Es el valor que puede tomar una constante o variable . Por ejemplo, para representar los datos de un alumno como: Nombre, Num_cta, calf1, calf2, etc.

Los tipos de datos más utilizados son:

a) Numéricos: Representan un valor entero y real.

Ejemplo:

Entero: 250, -5

Real: 3.1416, -27.5

2. Metodología para la descomposición de un algoritmo.

2.2.2 Tipos de datos

b) Lógicos: Solo pueden tener dos valores (verdadero o falso), y son el resultado de una comparación.

c) Alfanuméricos: Son una serie de caracteres que sirven para representar y manejar datos como nombres de personas, artículos, productos, direcciones, etc.

2.2.3 Variables

Permite almacenar de forma temporal un valor y el cual puede cambiar durante la ejecución del algoritmo ó programa.

Toda variable tiene un nombre que sirve para identificarla.

Ejemplo:

$$\text{prom} = (\text{calf1} + \text{calf2} + \text{calf3}) / 3$$

Las variables son: prom, calf1, calf2, calf3.

2.2.4 Constantes

Son datos numéricos o alfanuméricos que contienen un valor y que no cambia durante la ejecución del algoritmo ó programa.

Ejemplos:

$\text{prom} = (\text{calf1} + \text{calf2} + \text{calf3}) / 3$

$\text{PI} = 3.1416$

Las constantes son: 3, PI.

2.2.5 Operadores y Expresiones

Expresiones: Es un conjunto de constantes, variables, operadores con lo que se realizan las operaciones y permite obtener un resultado.

Ejemplo:

resultado $\leftarrow a*(2*b+5)/c$

Cal_final $\leftarrow (cali1+cali2)/2$

Operadores: Es un símbolo que permite manipular los valores de variables y/o constantes.

2.2.5.1 Operadores matemáticos

- 1) \wedge $**$
- 2) $*$ $/$ div mod
- 3) $+$ $-$

Los operadores con igual nivel de prioridad se evalúan de izquierda a derecha.

2.2.5.2 Operador de asignación

- 1) $=$ \acute{o} \leftarrow

Sirve para recuperar o guardar los valores obtenidos al realizarse o ejecutarse una expresión.

2.2.5.3 Operadores de relación

- | | |
|--------------------|--------|
| 1) Mayor que | > |
| 2) Menor que | < |
| 3) Mayor igual que | >= |
| 4) Menor igual que | <= |
| 5) Igual | = |
| 6) Diferencia | < > != |

- Son empleados para comparar dos ó más valores.
- Su resultado produce valores como verdadero y falso.
- Tienen el mismo nivel de prioridad.

2.2.5.4 Operadores Lógicos o booleanos

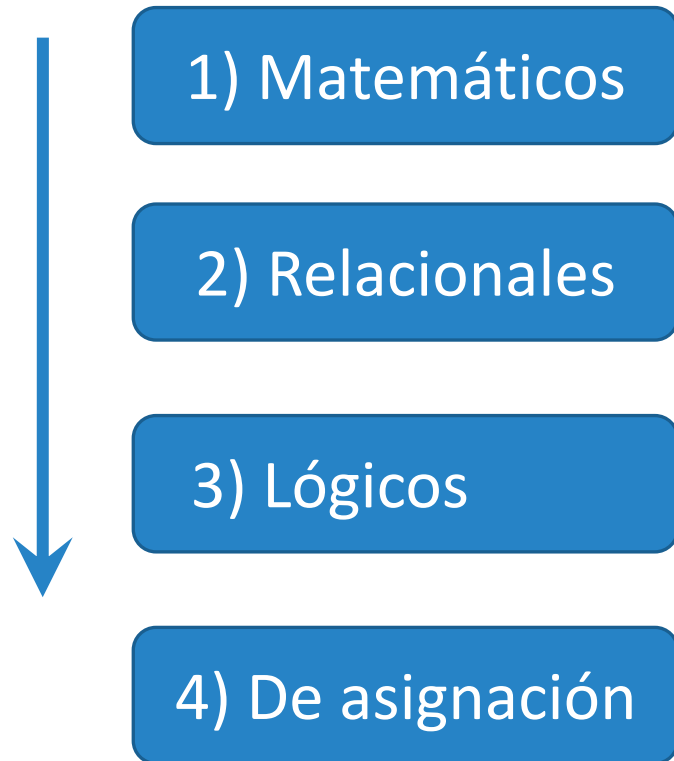
AND		
VAL1	VAL2	RESUL
Cierto	Cierto	Cierto
Cierto	Falso	Falso
Falso	Cierto	Falso
Falso	Falso	Falso

OR		
VAL1	VAL2	RESUL
Cierto	Cierto	Cierto
Cierto	Falso	Cierto
Falso	Cierto	Cierto
Falso	Falso	Falso

NOT	
VAL1	RESUL
Cierto	Falso
Falso	Cierto

- Son empleados para comparar dos valores (verdadero y falso)
- Su resultado produce valores como verdadero y falso.
- Los tres tienen el mismo nivel de prioridad.

Prioridad entre los Operadores



Siempre se ejecutan de izquierda a derecha en caso de haber dos ó más operadores con el mismo nivel de prioridad.

2.3. Diseño de algoritmos

2.3.1. Alternativas de solución

Es la forma de representar la secuencia lógica de ejecución de instrucciones.


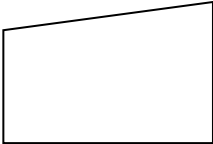
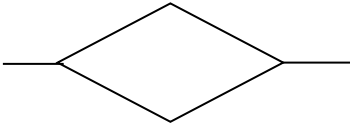
Esta puede ser a través de:

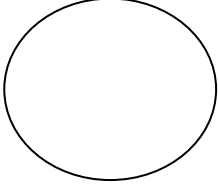
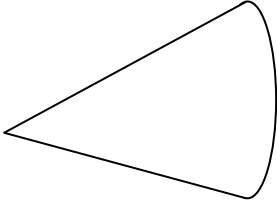

- 1) Diagramas de flujo
- 2) Pseudocódigo


2.3.1.1 Diagrama de flujo

Es empleado para representar la solución de un algoritmo empleando figuras geométricas, donde cada una de ellas representa en particular una tarea específica que realizar.

Las más comunes son:

SIMBOLO	UTILIDAD
	<p>El rectángulo se utiliza para identificar las acciones a realizar, es decir, este símbolo indica el <i>proceso</i> a realizar</p>
	<p>El trapezoide, indica la <i>entrada</i> o <i>lectura</i> de los datos</p>
	<p>El rombo, es la <i>caja de decisiones</i>, representa las alternativas con solo dos posibles opciones <i>SI y NO</i></p>

SIMBOLO	UTILIDAD
	Los círculos, son utilizados para indicar el inicio y el final del algoritmo.
	El cono se utiliza para indicar una salida en pantalla .
	La flecha, indica la secuencia de acciones a realizar , es decir, es quien marca la continuidad y orden de ejecución de las acciones propias del problema a resolver.

SIMBOLO	UTILIDAD
	Representa la repetición de pasos a través de los ciclos

2.3.1.2 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

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

2.3.2.1 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

2.3.2.1 Secuenciales

Ejemplo:

Pseudocódigo

Inicio

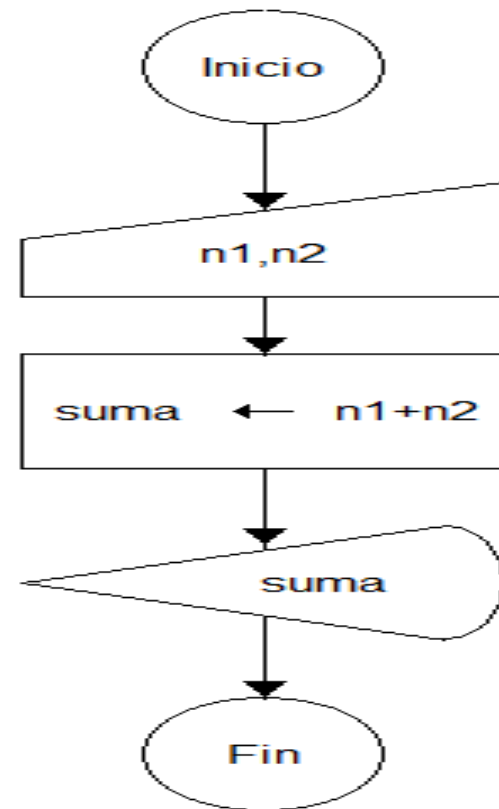
Leer N1, N2

SUMA=N1+N2

Escribir SUMA

Fin

DFD



2.3.2.2 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

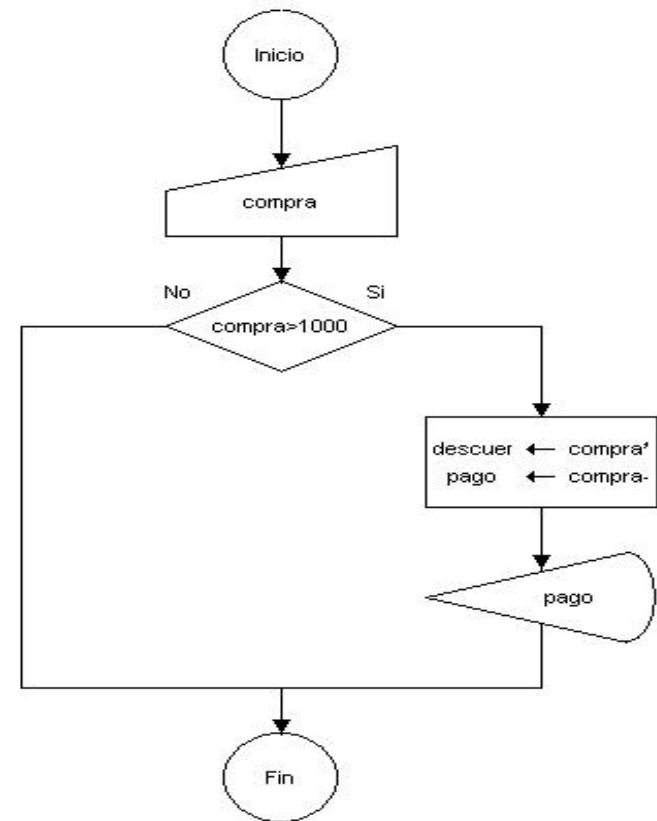
PAGO = COMPRA - DESCUENTO

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 $EDAD \geq 18$ entonces

 Escribir “Mayor de edad”

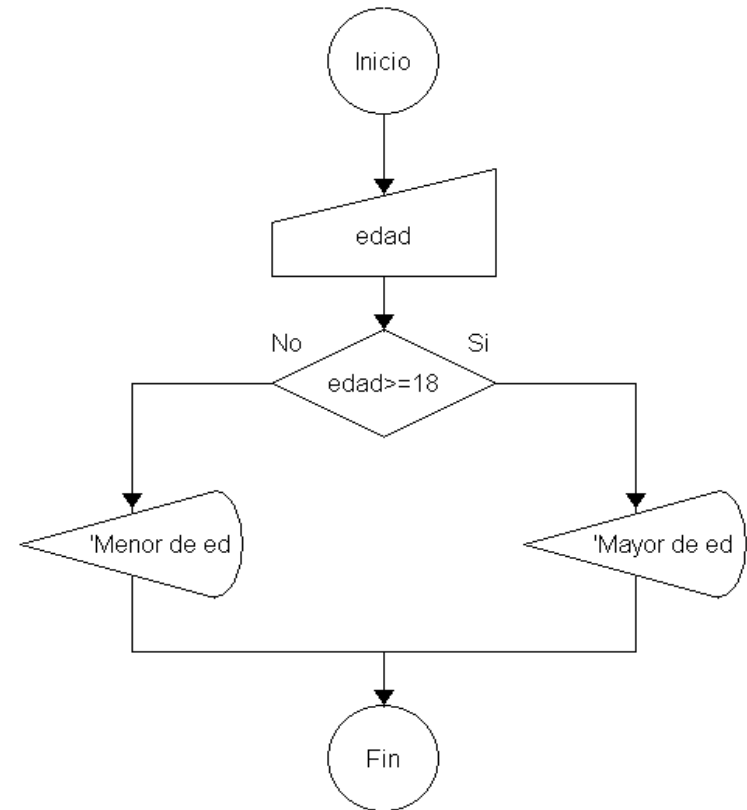
Sino

 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

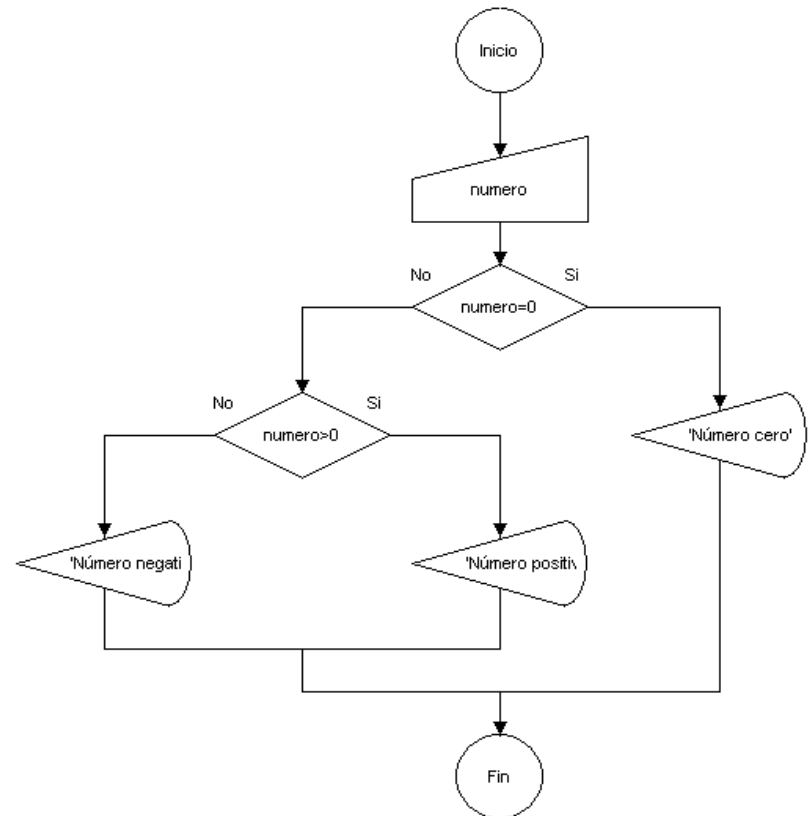
 Escribir “Número negativo”

 Finsi

Finsi

Fin

DFD



2.3.2.3 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;

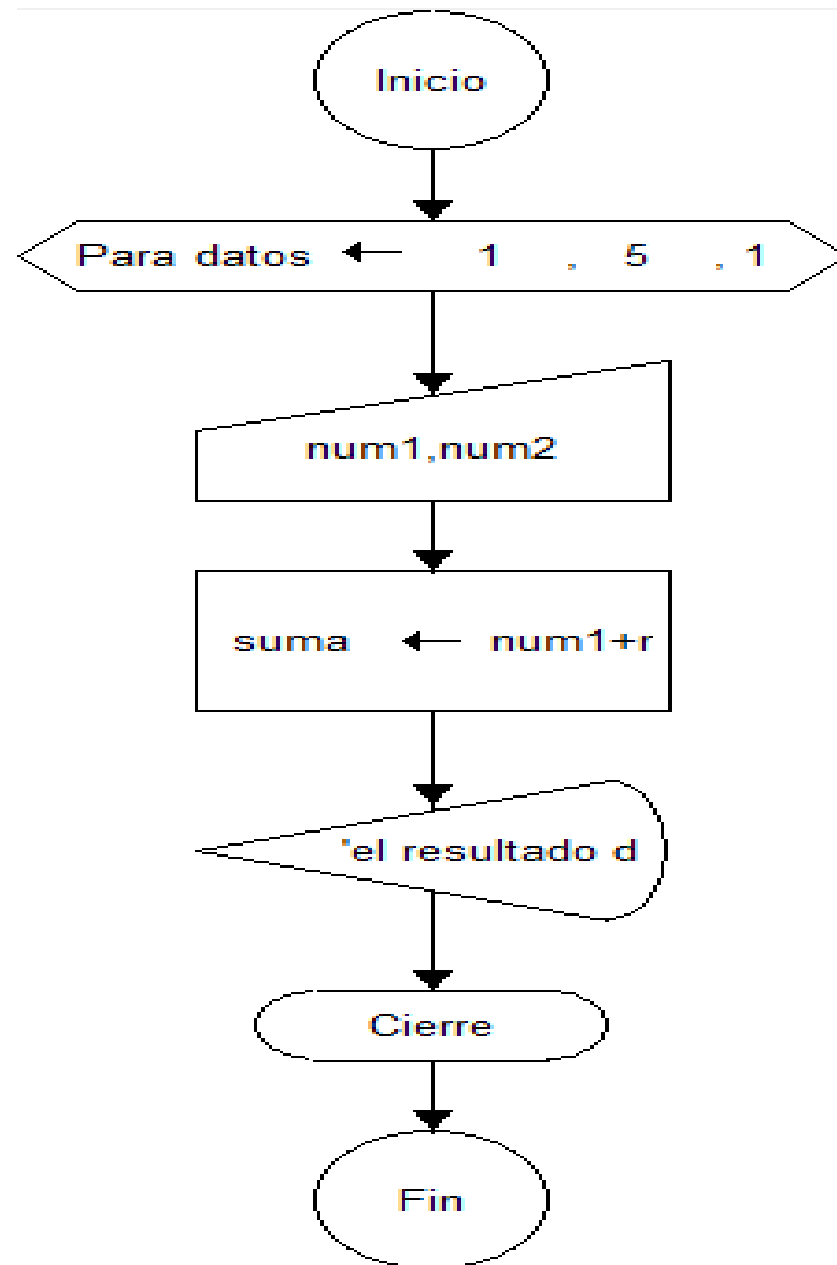
FinPara

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;

$PROM \leftarrow (CALIF1 + CALIF2) / 2$;

 Escribir "El promedio del alumno es ",PROM;

 Escribir "Hay alumno";

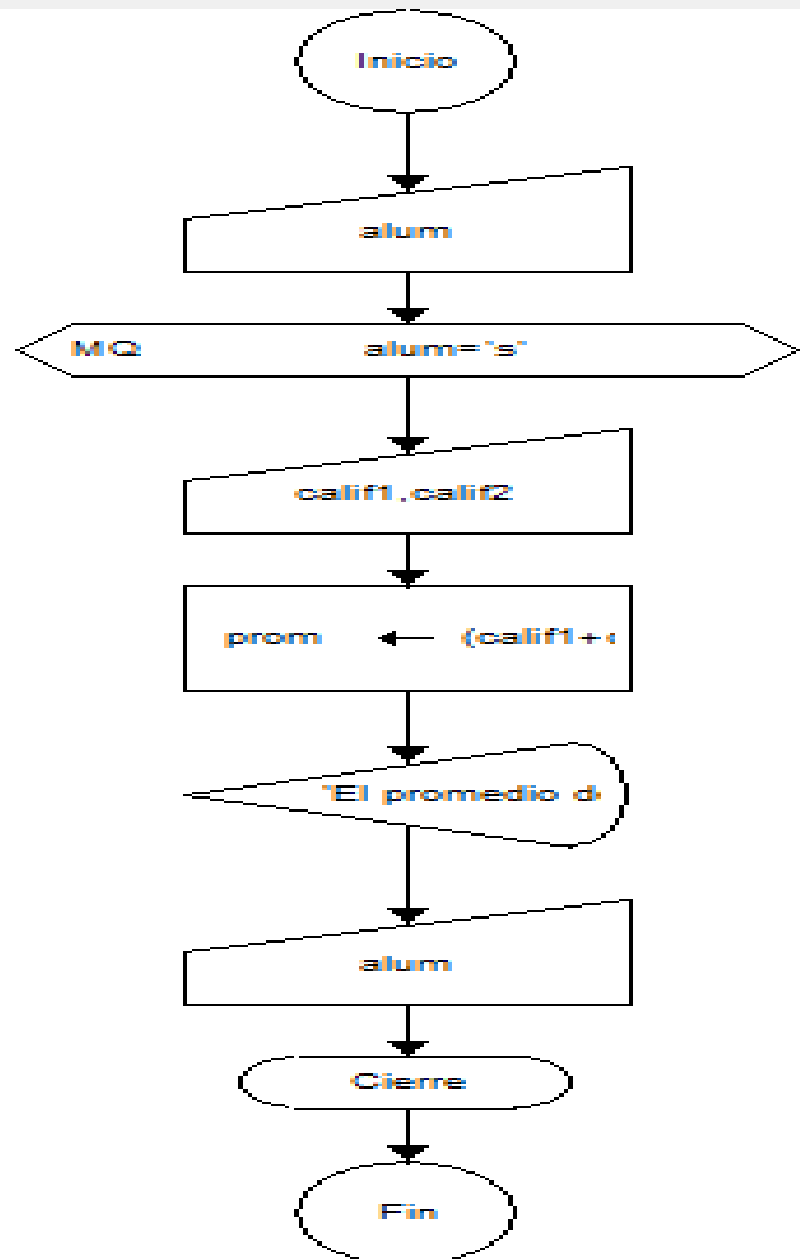
 Leer ALUM;

 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;

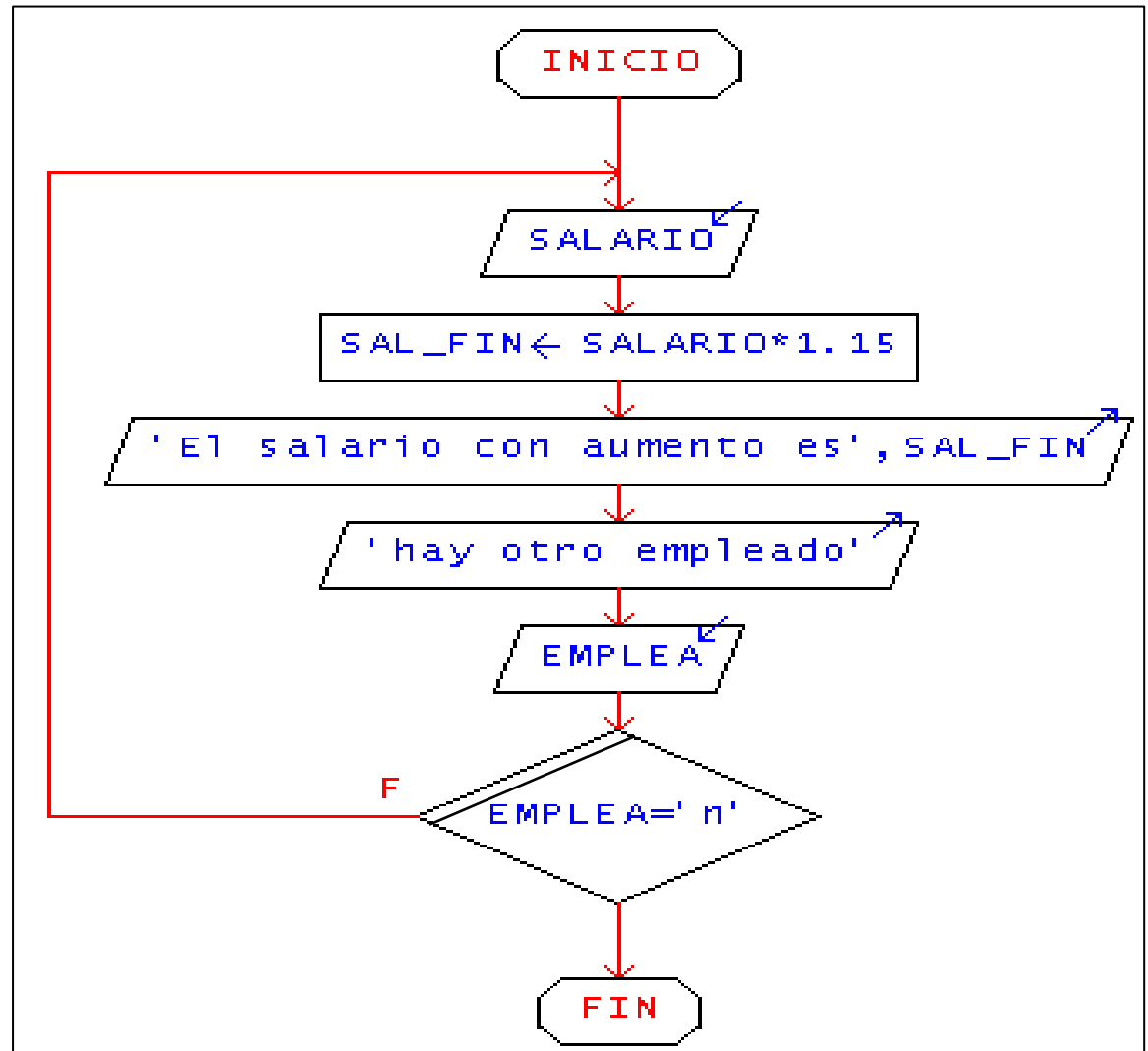
Hasta Que EMPLEA="n"

FinProceso

3) Repetitiva Hacer – Mientras ó Repetir

Ejemplo

Convertido a
diagrama de flujo
desde PseInt



BIBLIOGRAFÍA

1. Samperio Monroy Theira Irasema. Antología “Programación Estructurada”. Diciembre 2006
2. Cairó Olvaldo, Metodología de la programación (algoritmos, diagramas de flujo y programas), Editorial Alfaomega, Segunda edición.
3. Joyanes Aguilar Luís, Fundamentos de programación (Algoritmos, estructuras de datos y objetos), Editorial McGraw Hill, Tercera Edición.
4. Ferreyra Cortés Gonzalo, Informática para cursos de bachillerato, Editorial Alfaomega, Segunda Edición
5. Imágenes obtenidas del Software DFD y Pseint