Algoritmo

Un algoritmo es una secuencia finita de instrucciones realizables, no ambiguas, cuya ejecución conduce a una resolución de un problema.

Otra definición de algoritmo es la siguiente: Un algoritmo es una metodología para resolver unos problemas mediante una serie de fases o etapas precisas, definidas y finitas.

Así pues, si queremos que un ordenador efectúe una tarea, primero debemos descubrir un algoritmo para llevarla a cabo; programar el algoritmo en la máquina consiste en representar ese algoritmo de modo que se pueda comunicar a una máquina. En otras palabras, debemos transformar el algoritmo conceptual en un conjunto de instrucciones y representar estas últimas en un lenguaje sin ambigüedad.

ESTRUCTURA SUPERFICIAL DE LOS ALGORITMOS:

Los datos pueden ser de varios tipos:

- De entrada.
- De salida.
- Internos o de proceso (ofrecen resultados).

Hay al igual que los datos acciones de..

- Entrada: recogen los datos para el trabajo.
- Proceso: ejecutan los cálculos.
- Salida: ofrecen resultados.

- DISEÑO DEL ALGORITMO

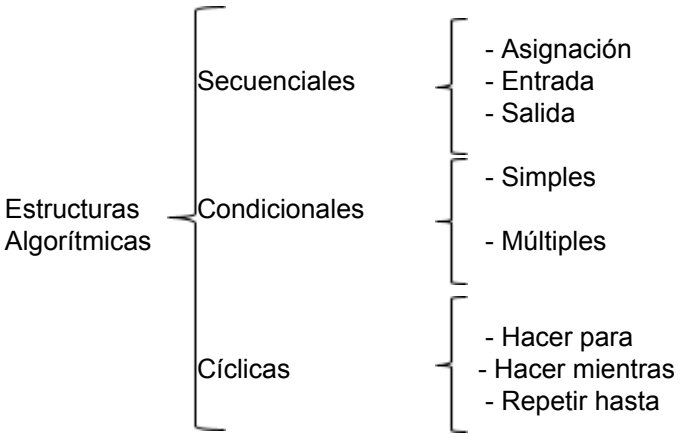
  Las características de un buen algoritmo son:
  - Debe tener un punto particular de inicio.
  - Debe ser definido, no debe permitir dobles interpretaciones.
  - Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
  - Debe ser finito en tamaño y tiempo de ejecución.

Prueba de escritorio o Depuración

Se denomina prueba de escritorio a la comprobación que se hace de un algoritmo para saber si está bien hecho. Esta prueba consiste en tomar datos específicos como entrada y seguir la secuencia indicada en el algoritmo hasta obtener un resultado, el análisis de estos resultados indicará si el algoritmo está correcto o si por el contrario hay necesidad de corregirlo o hacerle ajustes.

ESTRUCTURAS ALGORITMICAS

Las estructuras de operación de programas son un grupo de formas de trabajo, que permiten, mediante la manipulación de variables, realizar ciertos procesos específicos que nos lleven a la solución de problemas. Estas estructuras se clasifican de acuerdo con su complejidad en:



Una de las formas de representar y expresar un algoritmo son los diagramas de flujo.

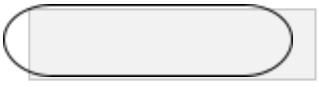
Diagramas de flujo.

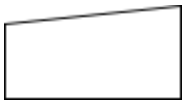
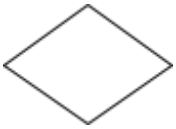
Los diagramas de flujo son una manera de representar visualmente el flujo de datos a través de sistemas de tratamiento de información. Los diagramas de flujo describen que operaciones y en que secuencia se requieren para solucionar un problema dado.

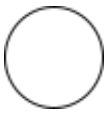
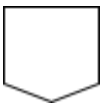
Un diagrama de flujo u organigrama es una representación diagramática que ilustra la secuencia de las operaciones que se realizarán para conseguir la solución de un problema. Los diagramas de flujo se dibujan generalmente antes de comenzar a programar el código frente a la computadora. Los diagramas de flujo facilitan la comunicación entre los programadores y los usuarios finales si el diseño de dichos programas viene de las necesidades de un tercero. Estos diagramas de flujo desempeñan un papel vital en la programación de un problema y facilitan la comprensión de problemas **complicados** y sobre todo **muy largos,** a demás de esto solo a través de un diagrama de flujo es posible la rápida localización de los errores de lógica los cuales no son fáciles de detectar aun cuando se logare una codificación que corra en el computador. Una vez que se dibuja el diagrama de flujo, llega a ser fácil escribir el programa en cualquier lenguaje de alto nivel. Además vemos a menudo cómo los diagramas de flujo nos dan ventaja al momento de explicar el programa a otros. Por lo tanto, está correcto decir que un diagrama de flujo es una necesidad para la mejor documentación de un programa complejo.

Reglas para dibujar un diagrama de flujo.

Los Diagramas de flujo se dibujan generalmente usando algunos símbolos estándares; sin embargo, algunos símbolos especiales pueden también ser desarrollados cuando sean requeridos. Los símbolos estándares, que se requieren con frecuencia para diagramar programas de computadora se muestran a continuación:

Simbolo	Descripción
	Inicio / Fin. Este símbolo se utiliza para señalar el comienzo así como el final de un diagrama. Tradicionalmente se colocan las palabras “INICIO” ó “FIN” dentro de la figura para hacerlo más explícito.

	Es el único símbolo que solamente tiene una conexión (flecha) ya sea de salida, en el de inicio, o de entrada, para el de fin.
	Entrada por teclado de datos. En este símbolo se indican los valores iniciales que deberá recibir el proceso. Esto se hace asignándoles letras o nombres de variables para cada uno de los valores y anotando estas letras en el interior de la figura. algunos desarrolladores consideran que el símbolo representa una salida por pantalla además de una entrada por teclado.
	Operaciones de entrada y salida. Este símbolo representa cualquier operación de entrada y/o salida por lo que normalmente se indica esto con el uso de las palabras leer y/o escribir respectivamente.
	Proceso de datos. Este símbolo lo utilizaremos para señalar operaciones matemáticas, aritméticas o procesos específicos que se realicen con nuestros datos. La manera de anotar dichos procesos, puede ser mediante una descripción breve de la operación o mediante una asignación de dicha operación hacia una variable como por ejemplo: R ← A + B Este símbolo siempre deberá tener al menos una conexión de entrada y una de salida. Este símbolo es usado también para resumir una serie de pasos ya conocidos en un alguritmo.
	Decisión. Este símbolo nos representa una disyuntiva lógica o decisión. En su interior se anota una instrucción o pregunta que pueda ser evaluada como cierta o falsa y que determine el flujo del programa. Este símbolo es el único que puede contener dos salidas y en cada una de las salidas se suele poner un rótulo de “si/no” o “cierto/falso” indicando con esto cual de ellas se tomará según el resultado de la evaluación de la función. Es una buena práctica de diagramación utilizar siempre el mismo lado para los positivos siempre que esto sea posible.
	Salida por pantalla. Este símbolo se utiliza para mostrar un resultado, o para representar cualquier información que debe ver el usuario del programa.
	Salida por impresora. Dentro de el se coloca el mensaje y datos que queremos mandar a la impresora.

	**Conector de entrada o salida.** Este símbolo se utiliza para indicar un salto dentro del diagrama. Se utiliza con el propósito de facilitar la disposición plana de un diagrama y evitar el cruce excesivo de líneas a través del mismo. Este conector va asociado a un conector “gemelo” y junto con él, representa una puerta de entrada y de salida para el flujo del diagrama, es decir que cuando una flecha termina en un conector marcado con la letra “A”, se continuará el diagrama a partir de otro conector marcado con la misma letra tal como si se tratara de una línea continua in interrumpida.
	**Conector de fin página.** Este conector es idéntico en funcionamiento que el anterior, pero su forma pentagonal lo distingue y nos indica que debemos buscar el “gemelo” en una página distinta de la actual. Este conector lleva asociado una especie de salto entre páginas.
	Se utiliza para representar los subprogramas.

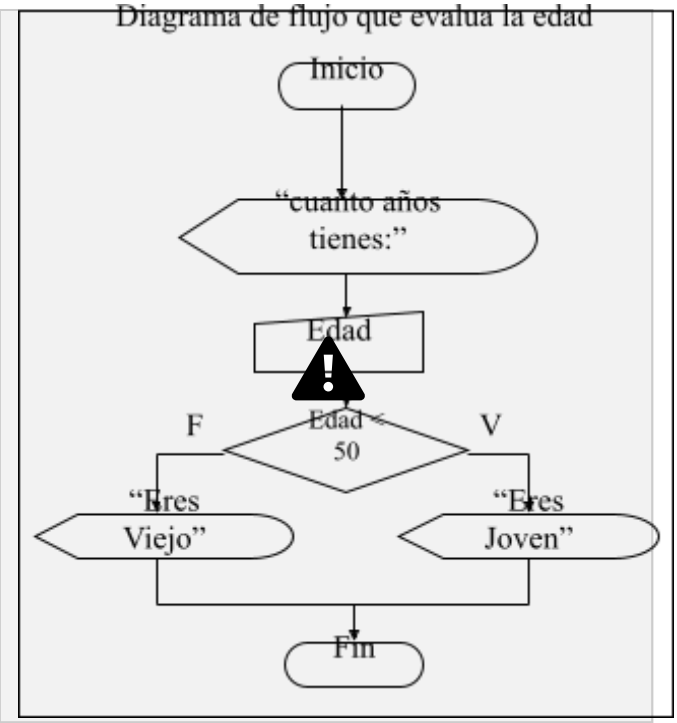
Observación: Para obtener la correcta elaboración de los símbolos, existen plantillas. Las puedes conseguir en Papelerías.

Reglas para la creación de Diagramas

1. Los Diagramas de flujo deben escribirse de arriba hacia abajo, y/o de izquierda a derecha.
2. Los símbolos se unen con líneas, las cuales tienen en la punta una flecha que indica la dirección que fluye la información procesos, se deben de utilizar solamente líneas de flujo horizontal o verticales (nunca diagonales).
3. Se debe evitar el cruce de líneas, para lo cual se quisiera separar el flujo del diagrama a un sitio distinto, se pudiera realizar utilizando los conectores. Se debe tener en cuenta que solo se vana utilizar conectores cuando sea estrictamente necesario.
4. No deben quedar líneas de flujo sin conectar
5. Todo texto escrito dentro de un símbolo debe ser legible, preciso, evitando el uso de muchas palabras.
6. Todos los símbolos pueden tener más de una línea de entrada, a excepción del símbolo final.
7. Solo los símbolos de decisión pueden y deben tener mas de una línea de flujo de salida.

Ejemplos de diagramas de flujo

Diagrama de flujo que encuentra la suma de los primeros 50 números naturales



Para la representación de un algoritmo así como para acercarnos más al código final de un programa existe un paso intermedio entre el diagrama y el código este es el denominado pseudocódigo

El pseudocódigo o pseudolenguaje, son una serie de instrucciones en nuestro lenguaje natural (español, ingles, etc.) y expresiones que representan cada uno de los pasos que resuelven un problema especifico (algoritmo).

Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado. El pseudocódigo utiliza palabras que indican el proceso a realizar, por todo lo anterior es una técnica **NO GRÁFICA**.

Se considera un primer borrador, dado que el Pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. Cabe señalar que el pseudocódigo no puede ser ejecutado por una computadora.

Para la escritura de un Pseudocódigo se han convenido las siguientes palabras:

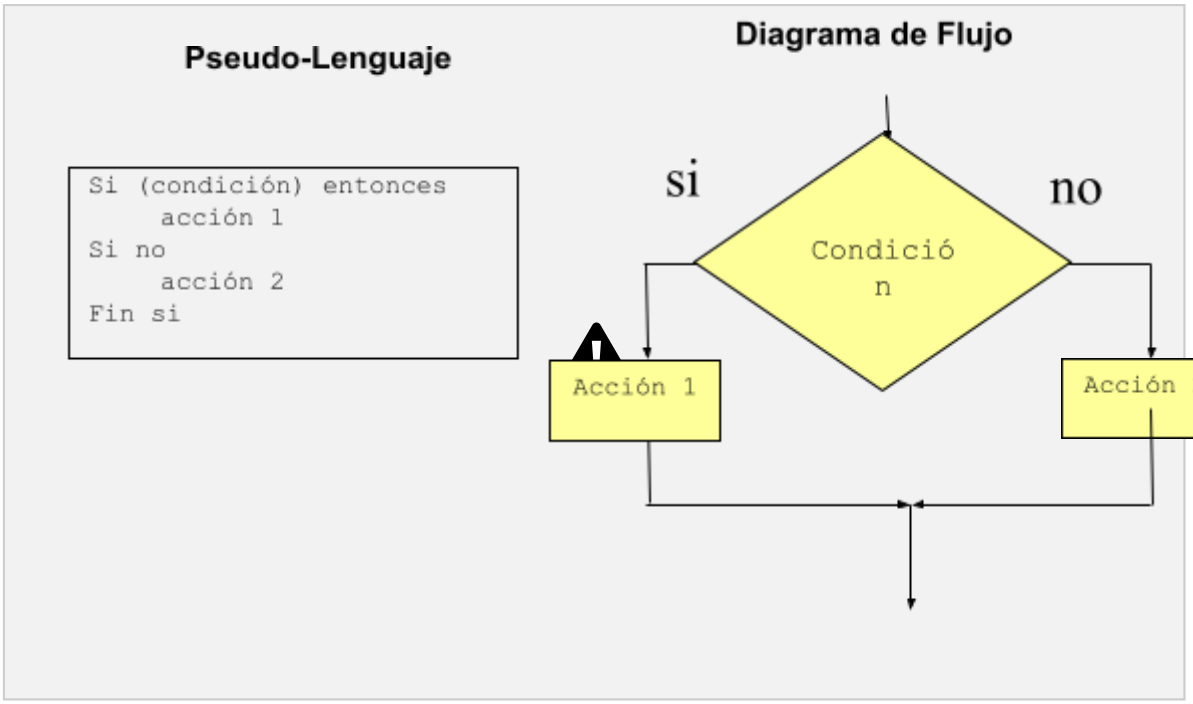
- **Inicio, Fin**. Indica el comienzo y término del algoritmo.
- **Escribir.** Muestra mensajes e información en el monitor.
- **Imprimir.** Datos y mensaje que son enviados a la impresora.
- **Leer.** Almacena un dato que es capturado desde el teclado en una variable.

- **Llamar a...** Indica que se debe de ejecutar a la función o módulo que se esta invocando.

- **Si ... entonces**. Es una pregunta para una estructura de decisión o selección, donde si la respuesta es verdad se realizan unas tareas especificas y cuando es falso se pueden realizar otras.

    - ☞ **Si no**. Indica el comienzo de las instrucciones a realizar cuando la respuesta a la pregunta ***si...entonces*** es falsa.
    - ☞ **Fin si**. Indica el término de la estructura condicional ***si...entonces***.

Veamos el diagrama comparativo de la dedición en diagrama y en seudo código



- **Seleccionar Caso** ... / **Fin selección**. Indica las acciones a realizar cuando una variable puede tener uno de varios posibles valores.

- **Hacer mientras...** / **fin mientras**. Estructura cíclica la cual indica un conjunto de instrucciones que se deben de repetir mientras que la respuesta a la pregunta ***hacer mientras...*** sea verdadera.

- **Repetir** / **hasta**.. Estructura cíclica la cual indica un conjunto de instrucciones que se deben de repetir mientras que la respuesta a la pregunta ***hasta...*** sea falsa.

- **Hacer para... hasta ...** / **fin para**. Estructura cíclica la cual indica el número exacto de veces que un conjunto de instrucciones que se deben de repetir.

La forma en que se escribe un pseudocódigo es la siguiente:

El Encabezado:

1. Declaración de **variables** se declaran los diferentes tipos de datos que posee el algoritmo. En Pseudocódigo solo se consideran los siguientes tipos entero, real y alfanumérico.

2. Declaración de **constantes** si existen.

3. Declaración de **valores iniciales** si existen contadores, acumuladores o banderas.

El Cuerpo del Pseudocódigo:

4. Son todas las instrucciones que componen la lógica del algoritmo

5. Se colocan en orden las instrucciones y expresiones a ejecutar, respetar lo siguiente:

   - ➢ La primera instrucción es la palabra **inicio** o **Programa** (nombre).
   - ➢ Cada línea representa una sola sentencia o instrucción.
   - ➢ Debe de respetarse una sangría que permita la colocación de etiquetas a las líneas que el algoritmo requiera.
   - ➢ La última instrucción es la palabra **fin**.

A continuación tenemos el ejemplo de un pseudocódigo, el cual no realiza nada específico, pero se muestra la estructura que debe de tener.

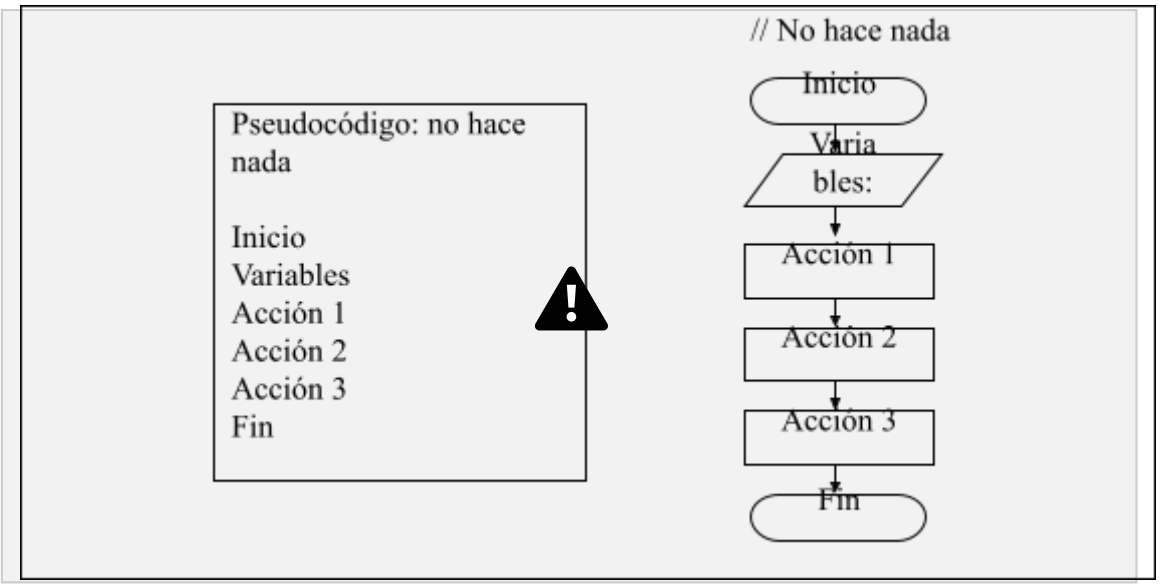
```
Inicio
Entero var3
Escribir “dame un número”
Leer var3
Si var3 == 10 entonces
        Escribir “hola”
si no
        Escribir “adiós”
fin si
escribir “gracias”
fin
```

Estructuras Secuénciales

Los algoritmos más sencillos de realizar son los que no toman decisiones, tan solo se dedican a realizar o ejecutar instrucción tras instrucción en el orden determinado.

Estos algoritmos están representados por las estructuras secuénciales, en las que una acción (instrucción) sigue a otra en **secuencia**. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

De manera general un algoritmo con una estructura secuencial se representa de la siguiente forma en las tres diferentes técnicas algorítmicas (el siguiente ejemplo no realiza nada en específico, solo es de carácter ilustrativo):



Estructuras Condicionales o de selección

Las estructuras condicionales comparan una variable contra otro(s) valor(es), para que en base al resultado de esta comparación, se siga un curso de acción dentro del programa.

Estas estructuras son las que nos dan la capacidad de crear sistemas inteligentes, es decir, que toman decisiones.

Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Existen dos tipos básicos, las simples y las múltiples.

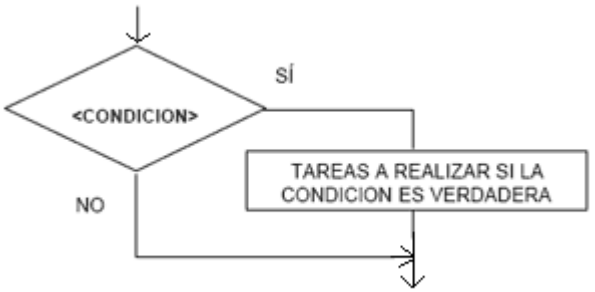
Condiciones Simples. Son aquellas en que solamente se puede escoger uno de dos caminos posibles y al seleccionar se ejecutarán las instrucciones que se encuentren dentro de este. Esto es similar a la situación que sufrimos cuando nos encontramos en la punta de una cuchilla, solamente se puede ir por un camino ya que es imposible cruzar por ambos a la vez.

Simples:

Las estructuras condicionales simples se les conocen como "Tomas de decisión". Estas tomas de decisión tienen la siguiente forma:

Pseudocódigo: **Diagrama de flujo:**

Si <condición> entonces
Instrucción (es)
Fin-Si

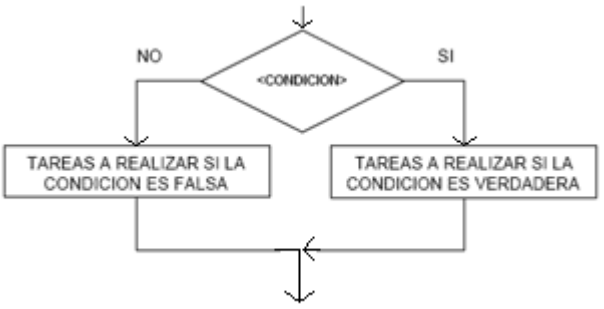


Dobles:

Las estructuras condicionales dobles permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición. Se representa de la siguiente forma:

Pseudocódigo: **Diagrama de flujo:**

Si <condición> entonces
Instrucción (es)
Si no
Instrucción (es)
Fin-Si



Donde:
Si: Indica el comando de comparación
Condición: Indica la condición a evaluar
Entonces: Precede a las acciones a realizar cuando se cumple la condición
Instrucción(es): Son las acciones a realizar cuando se cumple o no la condición
si no: Precede a las acciones a realizar cuando no se cumple la condición
Dependiendo de si la comparación es cierta o falsa, se pueden realizar una o más acciones.
Fin si: indica que las acciones alternativas si y sino ya se han culminado

Condiciones Múltiples. Son aquellas en que solamente se puede escoger uno de **n** caminos posibles, y al seleccionar se ejecutarán las instrucciones que se encuentren dentro de este. Esto

es similar a la situación que sufrimos cuando nos encontramos en un cruce de caminos, solamente se puede ir por un camino ya que es imposible cruzar por todos a la vez.

Pseudocódigo:

```
Si <condición> entonces
    Instrucción(es)
  Si no
    Si <condición> entonces
        Instrucción(es)
      Si no
          .
          .   Varias condiciones
          .
```

Diagrama de flujo:

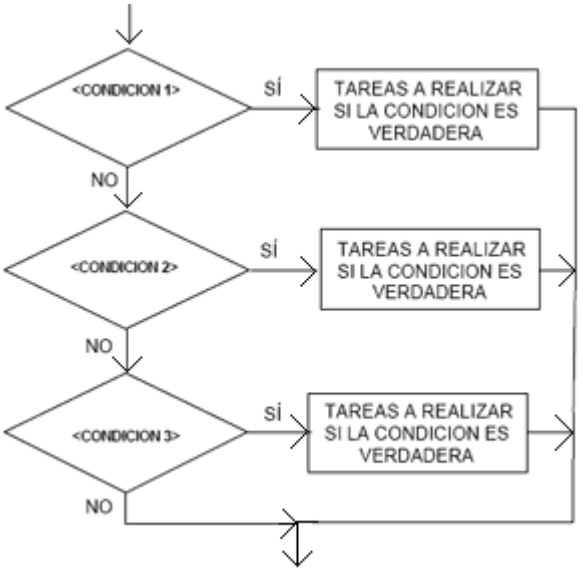
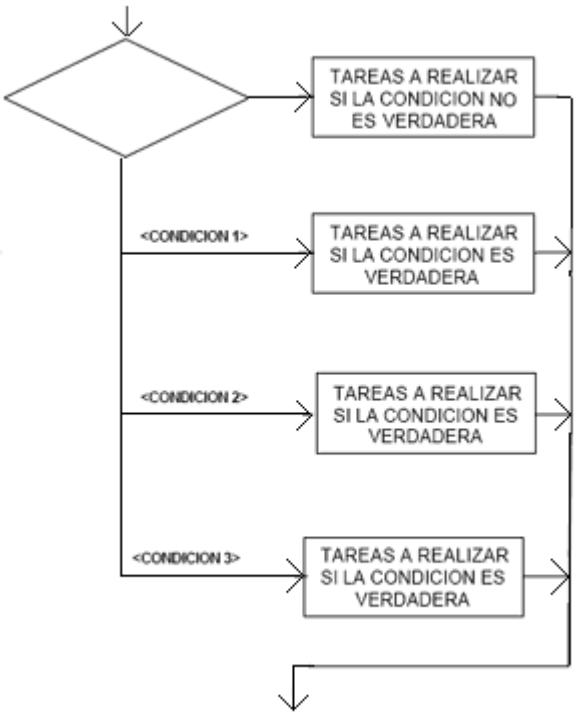


Diagrama de flujo para condiciones de selección múltiple:



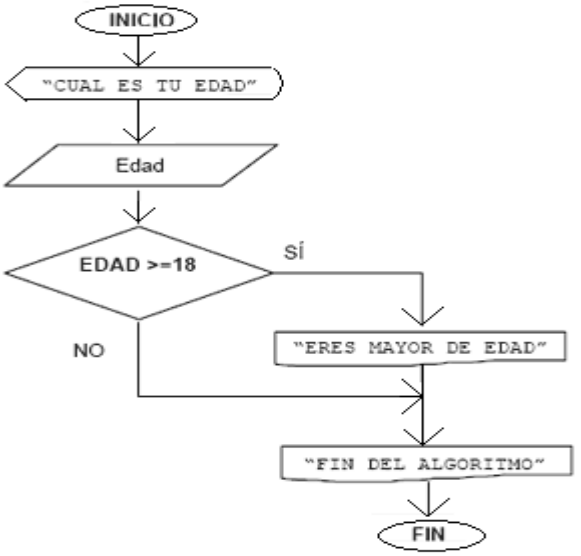
Veamos algunos ejemplos donde se aplique todo lo anterior:

Realizar un algoritmo en donde se pide la edad del usuario; si es mayor de edad debe aparecer un mensaje indicándolo. Expresarlo en Pseudocódigo y Diagrama de flujos.

Pseudocódigo:

```
INICIO
   Entero Edad
   ESCRIBA "Cual es tu edad?"
   LEA Edad
   Si Edad >=18 entonces
      Escriba "Eres mayor de edad"
   Fin Si
   ESCRIBA "Fin del algoritmo"
FIN
```

Diagrama de flujo:

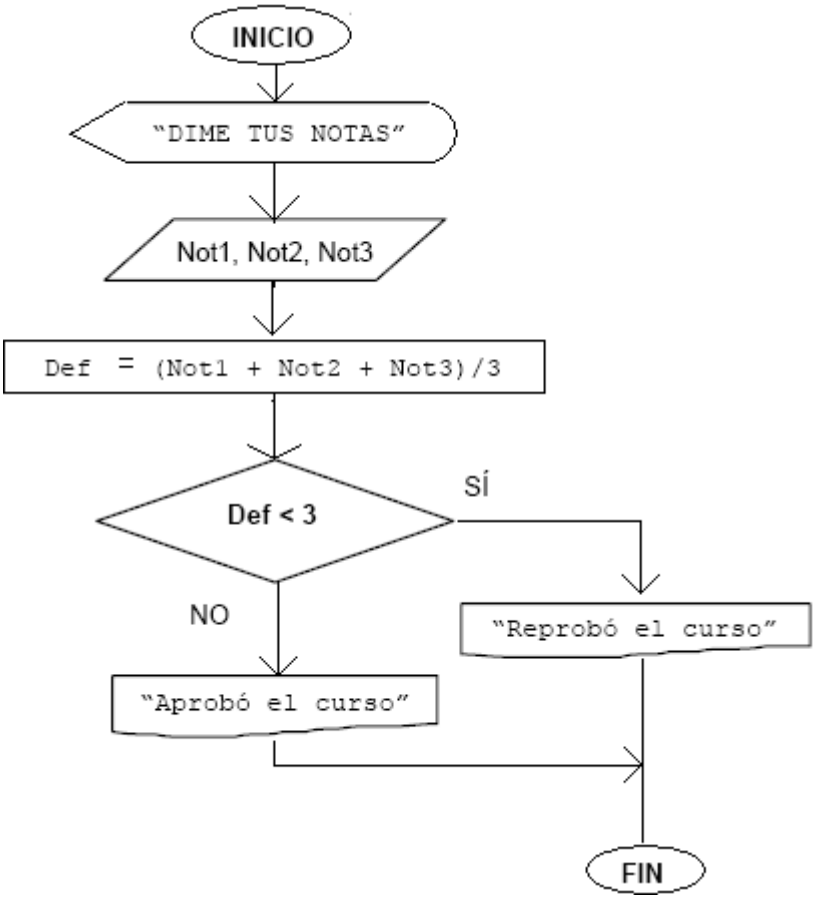


Se pide leer tres notas del alumno, calcular su definitiva en un rango de 0-5 y enviar un mensaje donde diga si el alumno aprobó o reprobó el curso. Exprese el algoritmo usando Pseudocódigo y diagrama de flujos.

Pseudocódigo:

```
INICIO
   REAL Not1, Not2, Not 3
   LEA Nota1, Nota2, Nota3
   Def ß (Not1 + Not2 + Not3) /3
   Si Def < 3 entonces
      Escriba "Reprobó el curso"
   Sino
      Escriba "Aprobó el curso"
   Fin-Si
FIN
```

Diagrama de flujo:



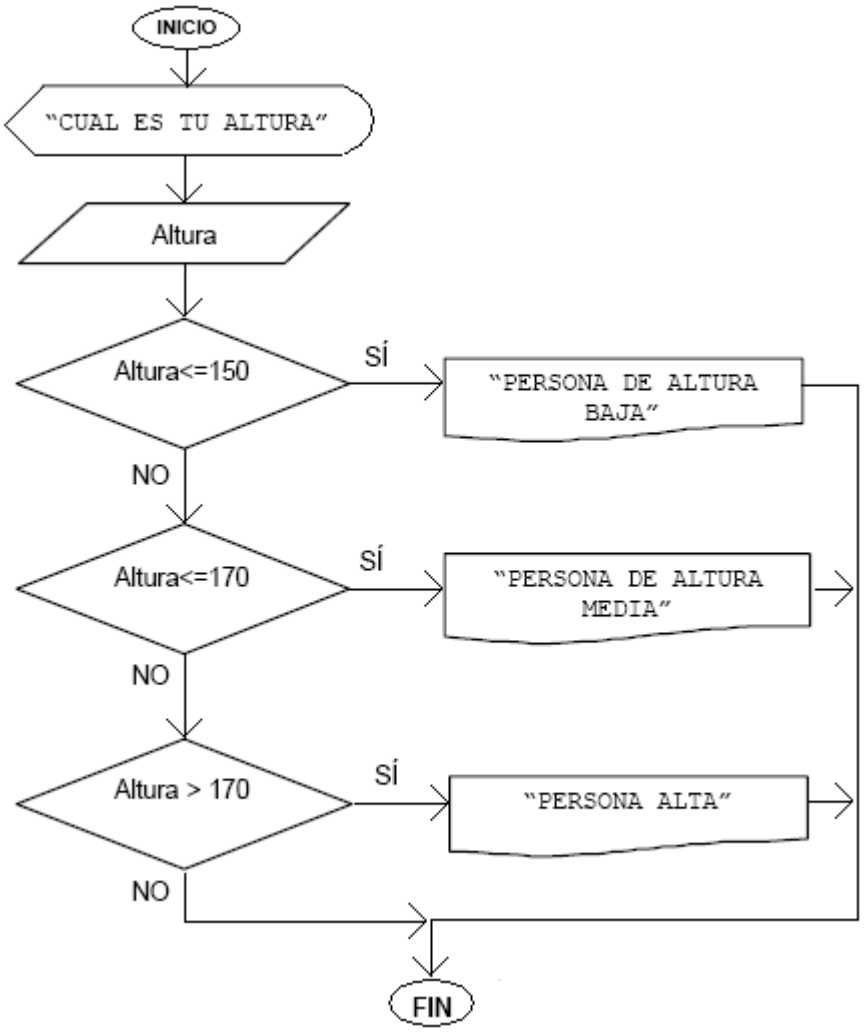
Se desea escribir un algoritmo que pida la altura de una persona, si la altura es menor o igual a 150 cm envíe el mensaje: "Persona de altura baja"; si la altura está entre 151 y 170 escriba el mensaje: "Persona de altura media" y si la altura es mayor al 171 escriba el mensaje: "Persona alta". Exprese el algoritmo usando Pseudocódigo y diagrama de flujos.

Pseudocódigo:

```
INICIO
  ENTERO Altura
  ESCRIBA "Cuál es tu altura?"
  LEA Altura
  Si Altura <=150 entonces
    ESCRIBA "persona de altura baja"
  Sino
    Si Altura <=170 entonces
      ESCRIBA "persona de altura media"
    Sino
      Si Altura>170 ENTONCES
        ESCRIBA "persona alta"
      Fin-Si
    Fin-Si
  Fin-Si
FIN
```

¡Es importante ser ordenado en el código que se escribe!

Diagrama de flujo:



Ejercicios propuestos

1. Hacer un diagrama para calcular el área de un triangulo.
2. Hacer un diagrama para convertir de grados centígrados a grados Fahrenheit.
3. Hacer un diagrama para imprimir la suma de los números del 1 al 100.
4. Hacer un diagrama que te pida un número y te diga si es par, es non y/o es primo.
5. Hacer un diagrama que pida 10 números y muestre el promedio.
6. Hacer un diagrama que pida 3 números y diga cual es el mayor.
7. Hacer un diagrama que pida la edad y despliegue si es menor de edad (<18), mayor (>=18) o si pertenece a la 3ª edad.(>=60)
8. Hacer un diagrama que te pida un número y te diga si es par, es none y/o es primo.
9. Hacer un diagrama para calcular el factorial de un número.
10. Hacer un diagrama que calcule e imprima N números primos.
11. Hacer un diagrama que solicite 4 calificaciones y diga si está reprobado o no, según las reglas la universidad.

12. Hacer un diagrama que despliegue la tabla de multiplicar de un número X.
13. Hacer un diagrama que pida 3 números y calcule el común denominador.