



UNIDAD 2.- ESTRUCTURA ELEMENTAL DE DATOS. DISEÑO DE ALGORITMOS.

OBJETIVOS:

- Que el alumno comience a formularse y resolver problemas, diseñando las estrategias correspondientes de manera clara, sistémica y por sobre todo sencilla, mediante el diseño de algoritmos.
- Que el alumno logre aumentar la capacidad de reflexión reforzando las conductas logradas mediante la Unidad 1.

TEMAS:

1. Concepto y definición de algoritmo.
2. Su representación gráfica: el diagrama de flujo lógico.
3. Símbolos utilizados.
4. Ventajas de la diagramación.
5. Prueba de escritorio.
6. Pautas básicas para el diseño general de un algoritmo.
7. El diseño descendente.
8. El teorema fundamental de la programación estructurada.
9. Estructuras: secuencial, de selección.
10. Estructuras de Repetición.
11. Complejidad Computacional. Orden de Complejidad.



1. CONCEPTO Y DEFINICIÓN DE ALGORITMO

CONCEPTO: Proceso de metodización del proceso heurístico.

DEFINICIONES:

- Método o procedimiento en que establece la secuencia de pautas a cumplir para realizar un determinado trabajo.
- Conjunto de instrucciones que permiten obtener una salida (output) específica partiendo de una entrada (input) conocida.

Según Donald Knuth (científico de computación), para que sea considerado un algoritmo informático debe cumplir las siguientes condiciones:

- 1) **SECUENCIAL:** *"Los pasos deben tener un orden perfectamente definido".*
- 2) **DEFINIDO:** *"Cada paso de un algoritmo debe estar precisamente definido; las operaciones a llevar a cabo deben ser especificadas de manera rigurosa y no ambigua para cada caso".*
- 3) **GENERAL:** *"Al encararse la solución de un problema, ésta debe desarrollarse para la generalidad de los casos que pueden presentarse dentro del ámbito de validez de esa solución".*
- 4) **FINITO:** *"Un algoritmo siempre debe terminar después de un número finito de pasos".*

Ejemplo 1:

Realice el algoritmo para el siguiente enunciado:

Ingresar tres valores enteros, calcule y muestre el promedio.

1. Inicio
2. Ingresar A, B, C
3. $S = A + B + C$
4. $P = S/3$
5. Mostrar P
6. Fin

Ejemplo 2:

Indique si el siguiente algoritmo es o no un Algoritmo Informático:

Ingresar dos valores enteros, calcule y muestre el cociente entre ambos valores.

1. Inicio
2. Ingresar A, B
3. $C = A/B$
4. Mostrar C
5. Fin



El algoritmo anterior no es un algoritmo informático porque no cumple las condiciones de ser:

- **General:** no cumple para la generalidad del universo de datos ($B=0$).
- **Definido:** cuando B toma el valor cero, el resultado no está definido.

2. SU REPRESENTACIÓN GRÁFICA: EL DIAGRAMA DE FLUJO LÓGICO

Es un esquema para representar gráficamente un algoritmo.

Se basan en la utilización de diversos símbolos para representar operaciones específicas.

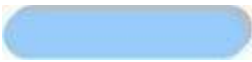
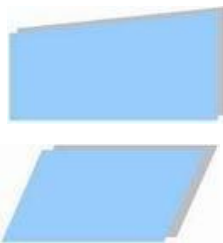

Se denomina diagrama de flujo porque los símbolos utilizados se conectan por medio de flechas para indicar la secuencia de operación.

Para hacer comprensibles los diagramas, los símbolos se someten a una normalización; es decir.




3. SÍMBOLOS UTILIZADOS

Existen diferentes símbolos utilizados en la diagramación, de los cuales podemos destacar cinco símbolos básicos con los cuales se puede representar prácticamente cualquier algoritmo informático por muy complejo que éste sea.



Estos símbolos son los siguientes:

Símbolo	Descripción
	Inicio / Terminación. 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 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. Este símbolo siempre deberá tener al menos una conexión entrante (generalmente del inicio) y una de salida.
	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.



	<p>Decisión. Este símbolo nos representa una 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.</p> <p>Este símbolo es el único que puede contener dos salidas y en cada una de las salidas se suele poner un rótulo de “sí/no” indicando con esto cual de ellas se tomará según el resultado de la evaluación de la decisión.</p> <p>Es una buena práctica de diagramación utilizar siempre el mismo lado para los positivos siempre que esto sea posible.</p>
 	<p>Desplegado de información. Este símbolo se utiliza para mostrar un resultado, el cual puede representar la solución al problema que se pretende resolver y que fue conseguida a través del resto del diagrama.</p> <p>Dentro de su interior se anotará la variable con el resultado final o el mensaje que represente el resultado del algoritmo.</p> <p>Este símbolo siempre deberá tener al menos una conexión de entrada y una de salida.</p>

En la diagramación, también contamos con una serie de símbolos auxiliares que no intervienen en el proceso del algoritmo, pero que pueden ser útiles para ayudarnos a dar claridad a nuestros diagramas, algunos de ellos son los siguientes:

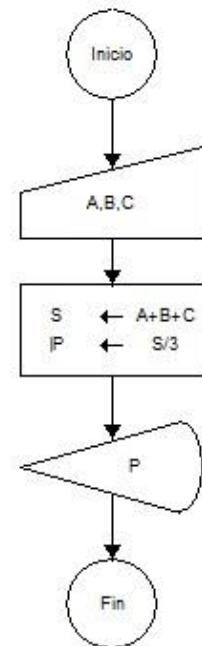
Símbolo	Descripción
	<p>Conector. Este símbolo se utiliza para indicar un salto dentro del diagrama. Se utiliza con el propósito de facilitar la disposición plana de un diagrama y evitar el cruce excesivo de líneas a través de este.</p> <p>Este conector va asociado a un conector “gemelo” y junto con 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 ininterrumpida.</p>
	<p>Conector de 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.</p>



Ejemplo:

Realice el diagrama de flujo para el siguiente enunciado:

Ingresar tres valores enteros, calcule y muestre el promedio.



4. VENTAJAS DE LA DIAGRAMACIÓN

Favorecen la comprensión del proceso a través de mostrarlo como un dibujo. El cerebro humano reconoce fácilmente los dibujos.

- **De uso:** Facilita su empleo.
- **De destino:** Permite la correcta identificación de actividades.
- **De comprensión e interpretación:** Simplifica su comprensión.
- **De simbología:** Disminuye la complejidad y accesibilidad.
- **De diagramación:** Se elabora con rapidez y no requiere de recursos sofisticados.

5. PRUEBA DE ESCRITORIO

La prueba de escritorio es una herramienta útil para entender que hace un determinado algoritmo, o para verificar que un algoritmo cumple con la especificación sin necesidad de ejecutarlo.

Básicamente, una prueba de escritorio es una ejecución 'a mano' del algoritmo, por lo tanto, se debe llevar registro de los valores que va tomando cada una de las variables involucradas en el mismo.

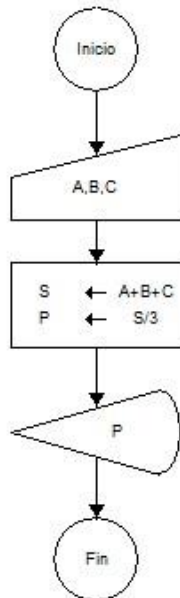
La prueba de escritorio es una tabla en la cual en la primera fila se colocan todos los identificadores de las variables y constantes y a continuación se completa con los valores que toma cada una de ellas.



Ejemplo:

Realice la prueba de escritorio para el diagrama de flujo que responde al siguiente enunciado:

Ingresar tres valores enteros, calcule y muestre el promedio.



PRUEBA DE ESCRITORIO

A	B	C	S	P
3	4	5	12	4

6. PAUTAS BÁSICAS PARA EL DISEÑO GENERAL DE UN ALGORITMO

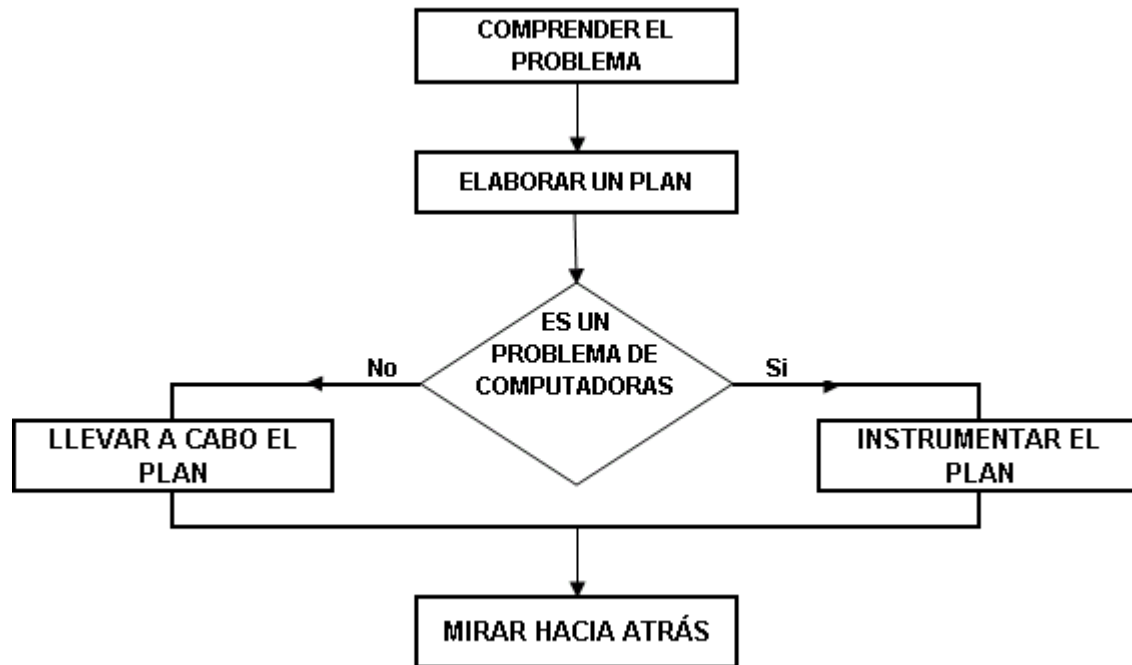
Los principios de George Polya, nos llevan al conocimiento de una técnica ordenada para el conocimiento y comprensión de los problemas.

No existe un método para la resolución de problemas en forma metódica o mecánica, sin embargo, los principios de este matemático brindan una serie de pautas para conducir nuestro pensamiento durante el proceso de solución de un problema, disponiendo así de un método heurístico para la resolución de problemas.

ENUNCIADO GENERAL DE LOS PRINCIPIOS DE GEORGE POLYA

1. Comprender el problema.
2. Elaborar un plan.
3. ¿Es un problema de computadora?
 - Afirmativo: Instrumentar el plan.
 - Negativo: Llevar a cabo el plan.
4. Mirar hacia atrás.

Gráficamente:



1. Comprender el problema.

- 1.1. ¿Cuáles son los inputs?
- 1.2. ¿Cuáles son los outputs deseados?
- 1.3. ¿Cuál o cuáles serían las condiciones vinculantes?
- 1.4. Dibujar una figura representativa esquematizando el enunciado del problema.
- 1.5. Introducir una notación adecuada, a usar durante el desarrollo resolutivo del problema.
- 1.6. Separar clara y adecuadamente las distintas partes del problema.

2. Elaborar un Plan

- 2.1. Encontrar conexiones entre los inputs y los outputs
- 2.2. ¿Puede derivarse algo del uso de determinados inputs?
- 2.3. Tal vez sea útil introducir algún problema auxiliar.
- 2.4. Tratar de hacer una buena suposición.
- 2.5. ¿Vio antes este problema?
- 2.6. ¿Conoce Usted algún problema similar?
- 2.7. Mirar lo desconocido.
- 2.8. Tal vez convenga simplificar el problema tratando de acercarnos a la solución.



3. ¿Es un problema de computadora?

Revisar si cumple con las condiciones para que el algoritmo sea considerado informático: Secuencial – Definido – Finito – General.

- o Afirmativo: Instrumentar el plan, elaborando el algoritmo correspondiente.
- o Negativo: Llevar a cabo el plan.

4. Mirar hacia atrás

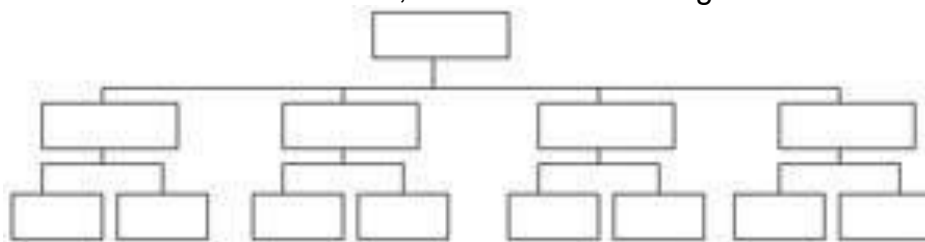
- 4.1. Revisar lo hecho y preguntarnos si podemos mejorar el plan.
- 4.2. ¿Podríamos usar esta solución en otras soluciones?
- 4.3. ¿Puede usted controlar los resultados?
- 4.4. ¿En base a los estudios anteriores se podrían derivar los resultados en forma diferente?

7. EL DISEÑO DESCENDENTE

También denominado diseño Top-Down, o de arriba hacia abajo, ya que se refiere a ver una gran imagen del sistema y luego de explotarla en partes o subsistemas pequeños, tal como, se muestra en la siguiente figura.

El diseño descendente permite que el analista de sistemas logre primero los objetivos organizacionales generales. Luego, el analista se mueve para dividir el sistema en subsistemas y sus requerimientos.

El diseño descendente es compatible con la manera de pensar sobre los sistemas en general. Cuando el analista de sistemas emplea un enfoque descendente, esta pensando acerca de las interdependencias de los subsistemas, tal como es en la organización existente.



Dentro de las ventajas de la utilización de un enfoque descendente en el diseño de sistemas, se encuentra:

- Evitar el caos originado al tratar de diseñar el sistema "en un solo paso". Como la planeación y la implementación de sistemas de información es increíblemente compleja.
- Posibilidad de contar con grupos de analistas de sistemas trabajando por separado, pero simultáneamente en subsistemas independientes, pero necesarios. Esto puede ahorrar una gran cantidad de tiempo. El trabajo de grupos integrados para el diseño de subsistemas es particularmente conveniente para la búsqueda del aseguramiento de la calidad total.



Existen ciertos inconvenientes en el diseño descendente que el analista de sistemas debe mantener en mente.

- Riesgo de que el sistema se divida en subsistemas "incorrectos". Se debe prestar atención a la necesidad de la superposición y la distribución de los recursos, de tal forma que una participación de subsistemas tenga sentido en el esquema global del sistema. Además, es importante que cada subsistema se integre de manera correcta al sistema.
- Una vez que se realizan las divisiones en subsistemas, sus interfaces pueden descuidarse o simplemente ignorarse. La responsabilidad para lograr la adecuada interrelación debe quedar bien detallada.

El enfoque descendente proporciona al grupo de sistemas una clara división establecida de los usuarios en fuerzas de tareas para los subsistemas.

Los grupos asignados a las tareas establecidos por esta manera pueden tener una función doble, tal y como los círculos de control de calidad de los sistemas informáticos para la administración.

8. EL TEOREMA FUNDAMENTAL DE LA PROGRAMACIÓN ESTRUCTURADA

La técnica de la programación estructurada tiende a elaborar sus programas en forma modular, lo cual posee dos ventajas:

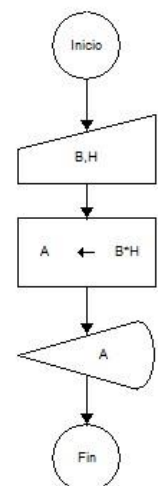
- Parcializar el estudio de la solución del problema, resolviendo el tema que plantea un determinado módulo.
- Trabajar en equipo.

PROGRAMA

Es un conjunto ordenado de instrucciones que permiten transformar los datos de entrada en datos de salida.

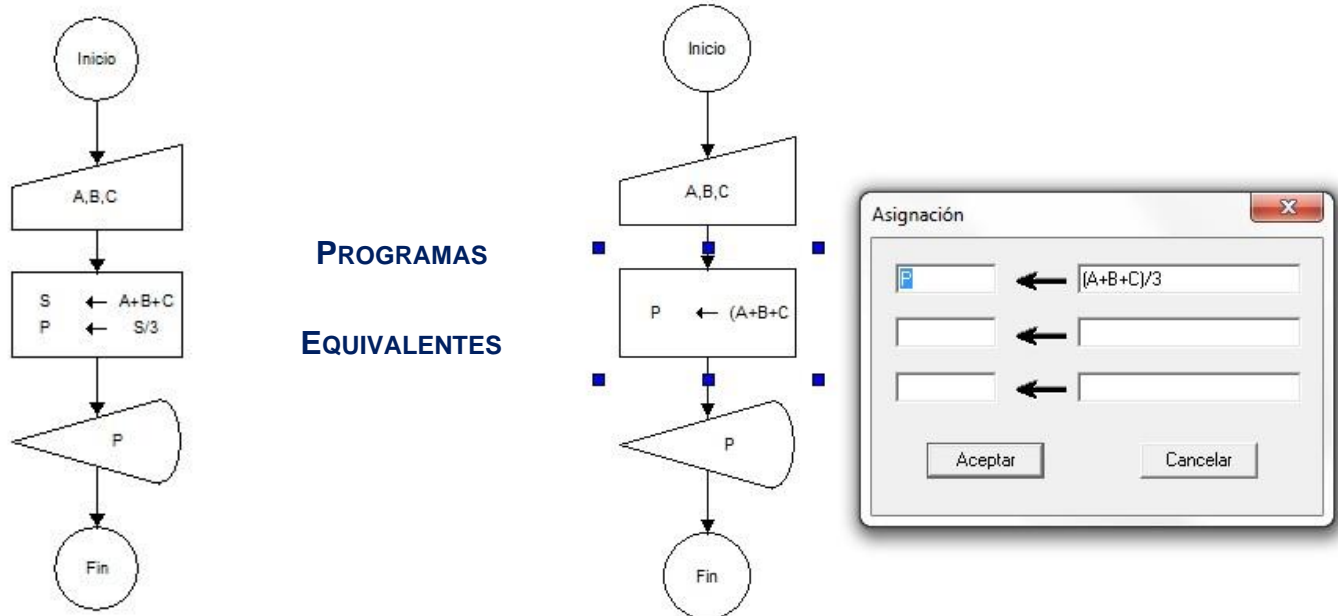
PROGRAMA PROPIO

Es aquel que posee un solo punto de entrada y uno de salida para controlar el programa.



PROGRAMA EQUIVALENTE

Son aquellos programas que producen iguales transformaciones de datos, es decir que para las mismas entradas producen idénticas salidas.



TEOREMA FUNDAMENTAL DE LA PROGRAMACIÓN ESTRUCTURADA

“Todo programa propio puede ser sustituido por un programa equivalente que tenga únicamente como estructuras lógicas las siguientes estructuras básicas:

1. *Secuencial,*
2. *Condicional, de Decisión o Selección,*
3. *Repetición o Iteración”*

La Filosofía de la Programación Estructurada se basa en 3 pautas fundamentales:

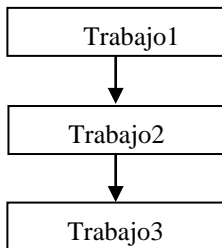
1. Estructuras Básicas.
2. Recursos abstractos o razonamiento.
3. Razonamiento Top-Down o Diseño Descendente.



9. ESTRUCTURAS: SECUENCIAL, DE SELECCIÓN Y REPETICIÓN

ESTRUCTURA SECUENCIAL

Representa una secuencia simple de problemas de evaluación, se representa esquemáticamente usando bloques convencionales.



Ejemplo 1:

Introduzca la altura y la base de un triángulo, encuentre y muestre el área del mismo.

Datos

B: identificador que indica la base del triángulo.

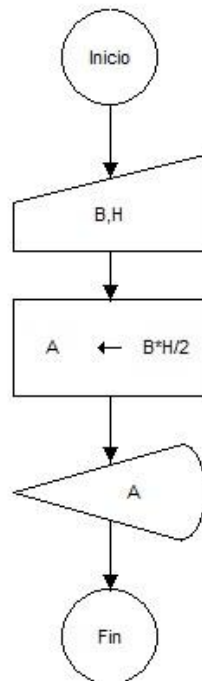
H: identificador que indica la altura del triángulo.

Resultado

A: identificador que indica el valor del área del triángulo.

Condiciones Vinculantes

Fórmula del área: $A = B * H / 2$



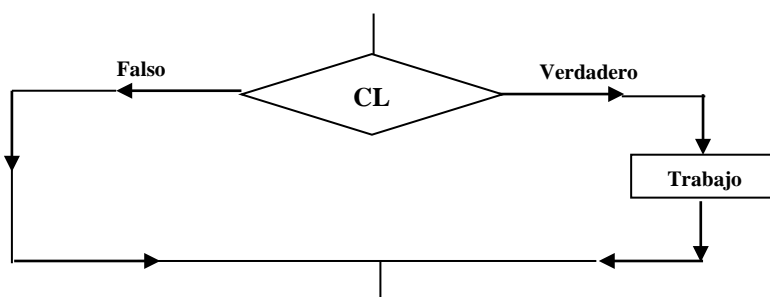
ESTRUCTURA DE SELECCIÓN

Su uso es evidente en los casos en los cuales se necesita bifurcar la solución por dos caminos alternativos de acuerdo a alguna condición lógica.

Las estructuras de selección pueden ser:

1. *Selección Simple*
2. *Selección Doble*
3. *Selección Múltiple*

Estructura de Selección Simple



Esta estructura presenta trabajo por una sola rama, generalmente por la rama cuyo resultado de la condición lógica es verdadero.

Ejemplo 1:

Introduzca un número entero e indique si el mismo es positivo.

Datos

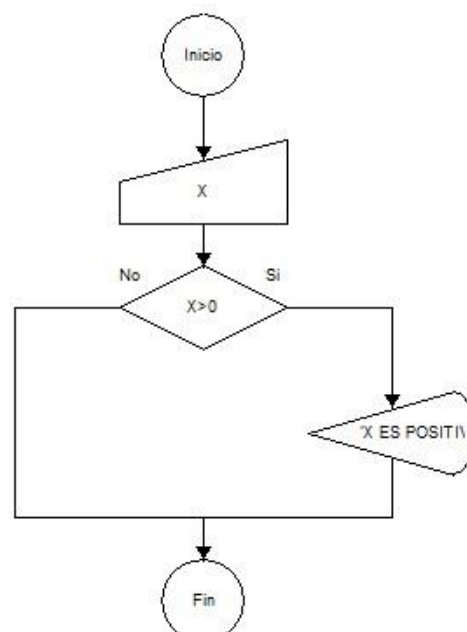
X: identificador que indica el valor del número entero ingresado.

Resultado

Mensaje que indica "X es positivo"

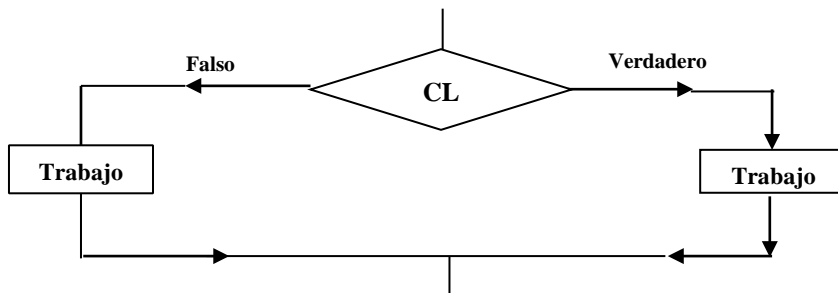
Condiciones Vinculantes

Si $(X > 0)$ entonces "X es positivo"





Estructura de Selección Doble



Esta estructura presenta trabajo por ambas ramas.

Ejemplo 1:

Introduzca un número entero e indique si el mismo es positivo o no lo es.

Datos

X: identificador que indica el valor del número entero ingresado.

Resultado

Mensajes:

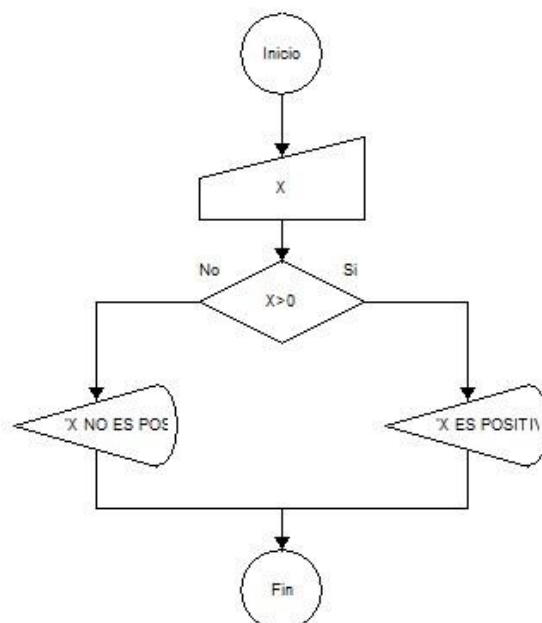
“X es positivo”

“X no es positivo”

Condiciones Vinculantes

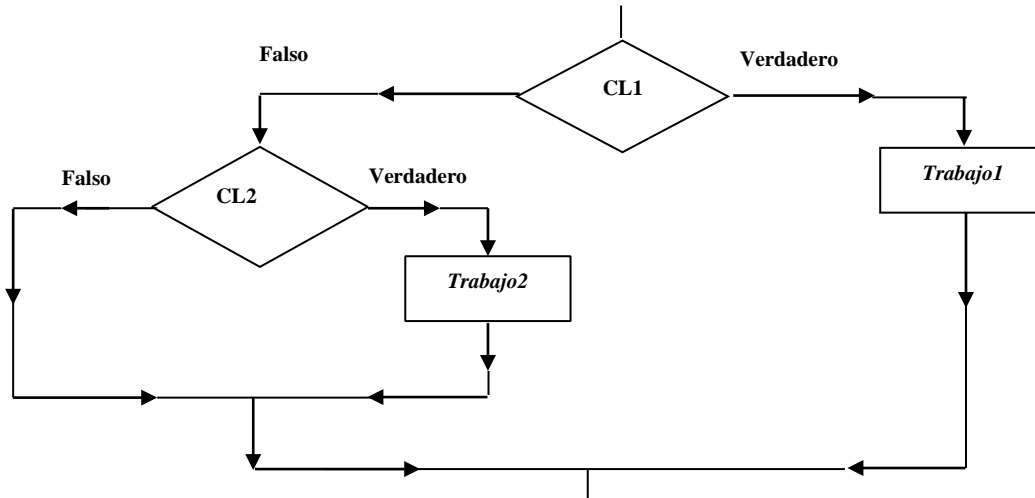
Si $(X > 0)$ entonces “X es positivo”

caso contrario “X no es positivo”





Estructura de Selección Múltiple



La estructura de selección múltiple presenta condiciones lógicas anidadas, es decir que por una o por varias ramas existen también otras condiciones lógicas. Este es un ejemplo de una estructura de selección múltiple.

Ejemplo 1:

Introduzca las coordenadas de un punto en el plano, e indique a qué cuadrante pertenece dicho punto.

Datos

X: identificador que indica el valor de la abscisa del número ingresado.

Y: identificador que indica el valor de las ordenadas del número ingresado.

Resultados

Mensajes:

"I cuadrante"

"II cuadrante"

"III cuadrante"

"IV cuadrante"

Condiciones Vinculantes

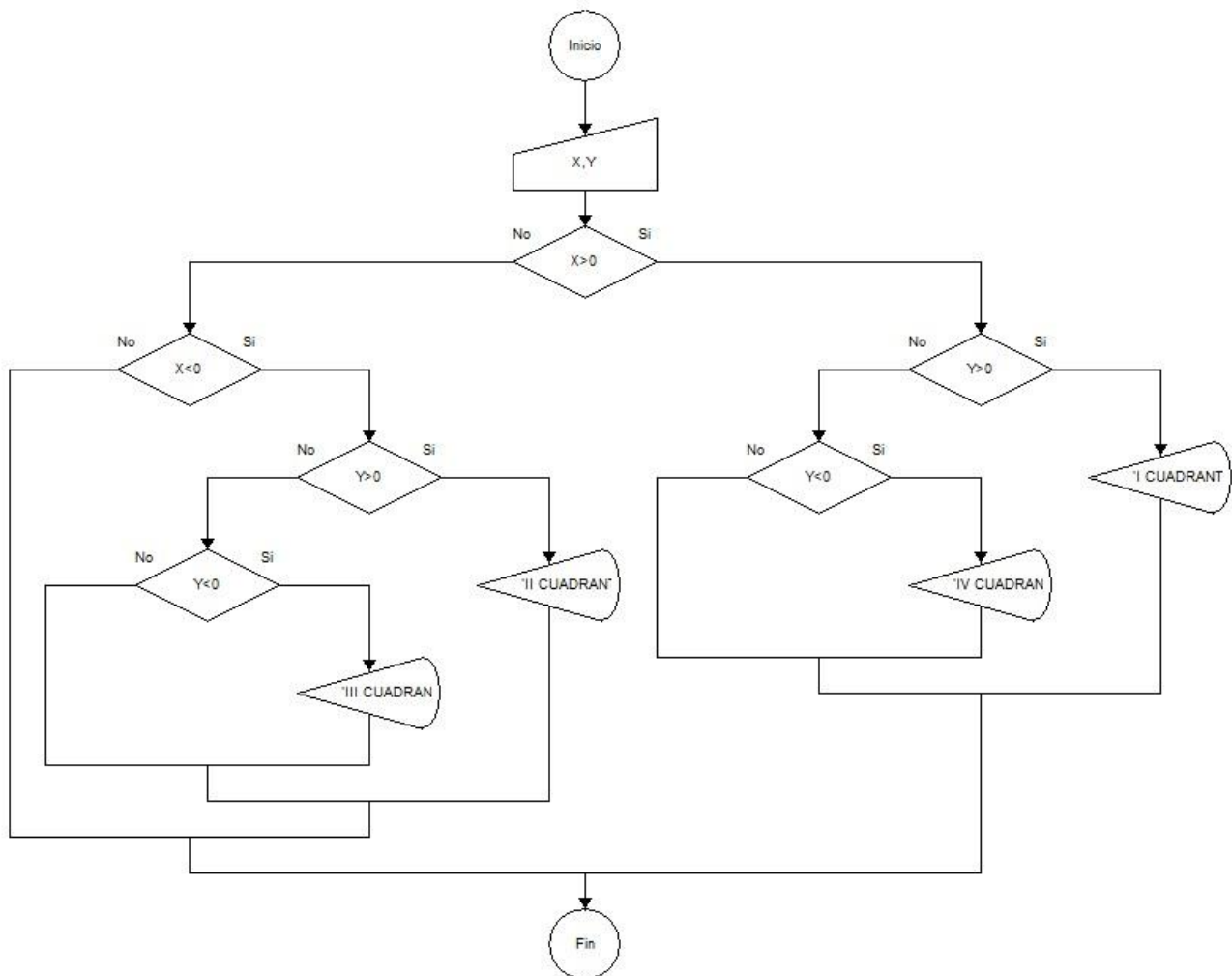
Condiciones para determinar a qué cuadrante pertenece el punto

$(X > 0) \text{ and } (Y > 0) \Rightarrow \text{"I cuadrante"}$

$(X < 0) \text{ and } (Y > 0) \Rightarrow \text{"II cuadrante"}$

$(X < 0) \text{ and } (Y < 0) \Rightarrow \text{"III cuadrante"}$

$(X > 0) \text{ and } (Y < 0) \Rightarrow \text{"IV cuadrante"}$



Toda estructura de selección múltiple que se abre a lo ancho puede ser representada por una estructura más simple pero cuyas condiciones lógicas serán expresiones compuestas.

