TECNICAS DE PROGRAMACION ESTRUCTURADA

Introducción

Las técnicas de desarrollo y diseño de programas que se utilizan en la programación convencional tienen inconvenientes, sobre todo la hora de verificar y modificar algún programa. En la actualidad están adquiriendo gran importancia las técnicas de programación, cuyo objetivo principal es el de facilitar la comprensión del programa, y además permiten, de forma rápida, las ampliaciones y modificaciones que surjan en la fase de explotación del ciclo de vida de un programa o una ampliación informática.
Una forma de simplificar los programas, haciendo más sencilla su lectura y mantenimiento, es utilizar la técnica del diseño descendente de programas (TOP – DOWN).
En los últimos años la técnica mas utilizada que siguen las directrices TOP – DOWN es la *programación estructurada*.
La programación estructurada fue desarrollada en sus principios por Edsgar W. Dijkstra en sus **Notes on Structures Programming** y se basa en el denominado Teorema de la Estructura desarrollado en 1966 por Bohm y Jacopini, que se ratificó con los trabajos de Harlan D. Mills.
En la programación convencional se suele hacer un uso indiscriminado y sin control de las instrucciones de salto condicional e incondicional, lo cual produce cierta complejidad en la lectura y en las modificaciones del programa. Eliminar estas dificultades es uno de los propósitos de la programación estructurada y, por ello, en ocasiones, se ha definido como la técnica de la programación sin saltos condicionales e incondicionales. Esto no es rigurosamente cierto, y por lo tanto no lo tomaremos como definición, sino como una norma general.
Como consecuencia del párrafo anterior podemos indicar que todo programa estructurado puede ser leído de principio a fin sin interrupciones en la secuencia normal de lectura.
Al mismo tiempo que se obtiene una mayor clarificación del programa por medio de estas técnicas, la puesta a punto del mismo es mucho más rápida, así como la confección de su documentación.
Los programadores en la fase de diseño realizan cada tarea en módulos o bloques, los cuales pueden estandizar y así formar su propia biblioteca de programas para su utilización en sucesivas aplicaciones.
En los distintos departamentos de informática existentes no siempre se dispone de los mismos programadores con respecto al tiempo que se pretende que dure una aplicación, por lo cual es de suma importancia que un programa realizado por una persona sea fácil de modificar y mantener por otra. En este sentido, la programación estructurada ofrece muchas ventajas para lograr estos objetivos.
Un programa estructurado es:

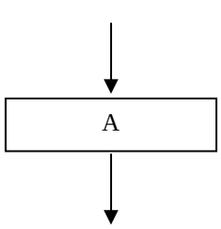
- Fácil de leer y comprender
- Fácil de codificar en una amplia gama de lenguajes y en diferentes sistemas.
- Fácil de mantener.
- Eficiente, aprovechando al máximo los recursos de la computadora.
- Modularizable.

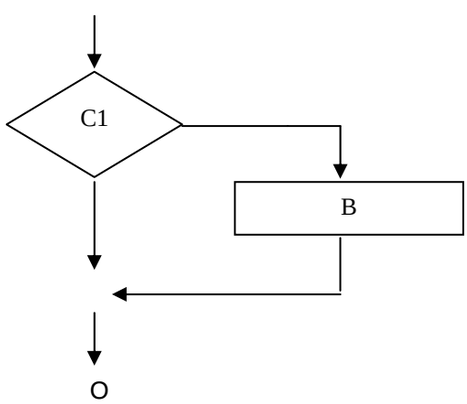
TEOREMA DE LA ESTRUCTURA

En la actualidad existen diversas definiciones de la programación estructurada, pero todas ellas giran en torno al teorema de la estructura que, como ya hemos dicho, se debe a Bohm y Jacopini.
Para un buen entendimiento del mismo realizamos la definición previa de diagrama propio, programa propio y equivalencia de programas que intervienen en su enunciado directa e indirectamente.

- **Diagrama propio:**

Es aquel que posee un solo punto de entrada y uno solo de salida.

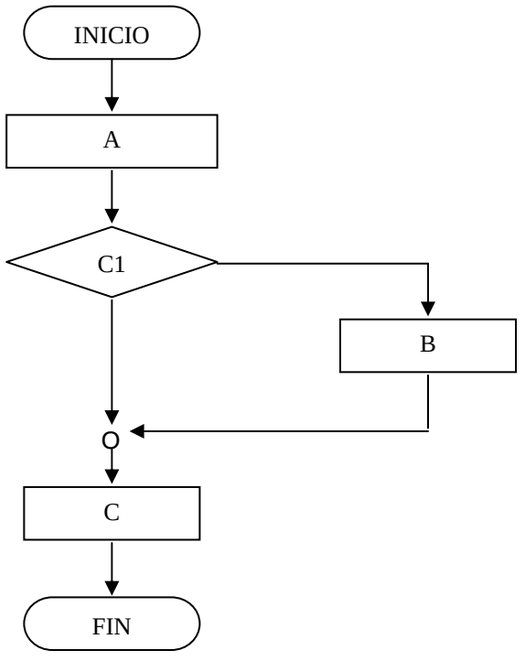




- **Programa propio:**

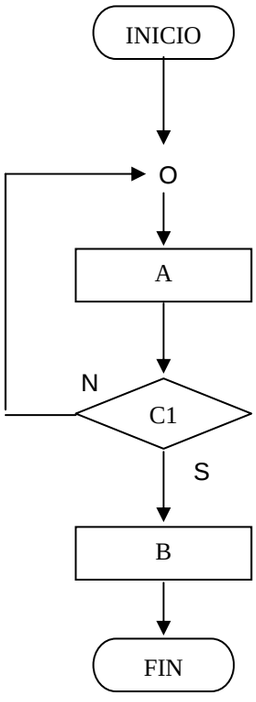
Es aquel programa que cumple las siguientes condiciones:

- Posee un solo inicio y un solo fin.
- Todo elemento del programa es accesible, es decir, existe al menos un camino desde el inicio al fin que pasa a través de él.
- No posee bucles infinitos.

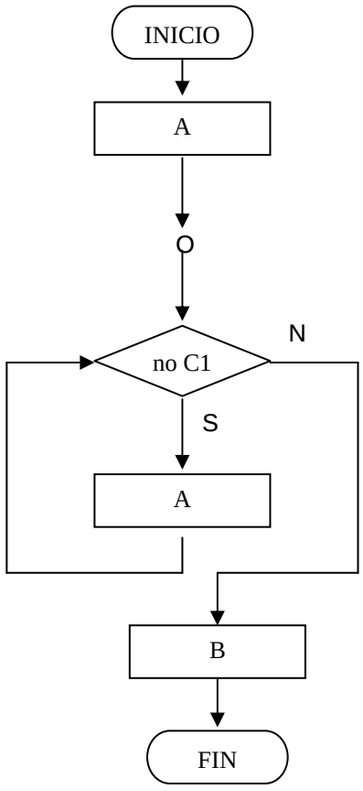


- **Equivalencia de programas:**

Dos programas son equivalentes si realizan, ante cualquier situación de datos, el mismo trabajo pero de distinta forma.



Es equivalente a



- *Teorema de la estructura:*

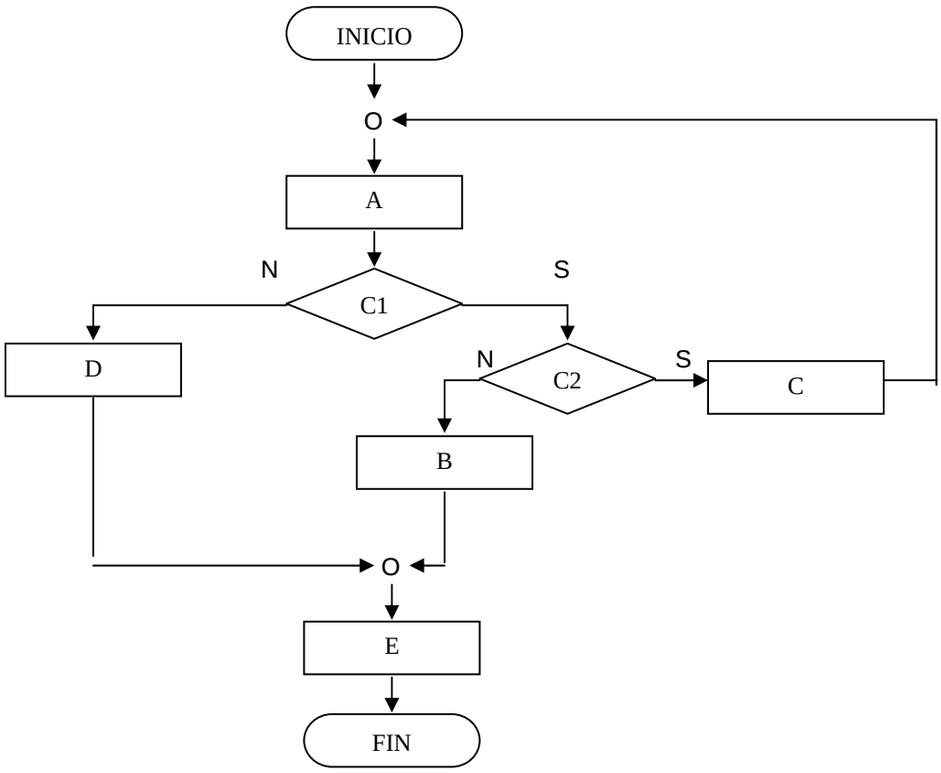
> Todo programa propio, realice el trabajo que realice, tiene siempre al menos un programa propio equivalente que solo utiliza las estructuras básicas de la programación, que son:
>
> - La secuencia
> - La selección
> - La repetición

En definitiva, el teorema nos viene a decir que, diseñando programas con sentencias primitivas (lectura, escritura y asignación) y estructuras básicas, no sólo podremos hacer cualquier trabajo sino que además conseguiremos mejorar la creación, lectura, comprensión y mantenimiento de los programas.

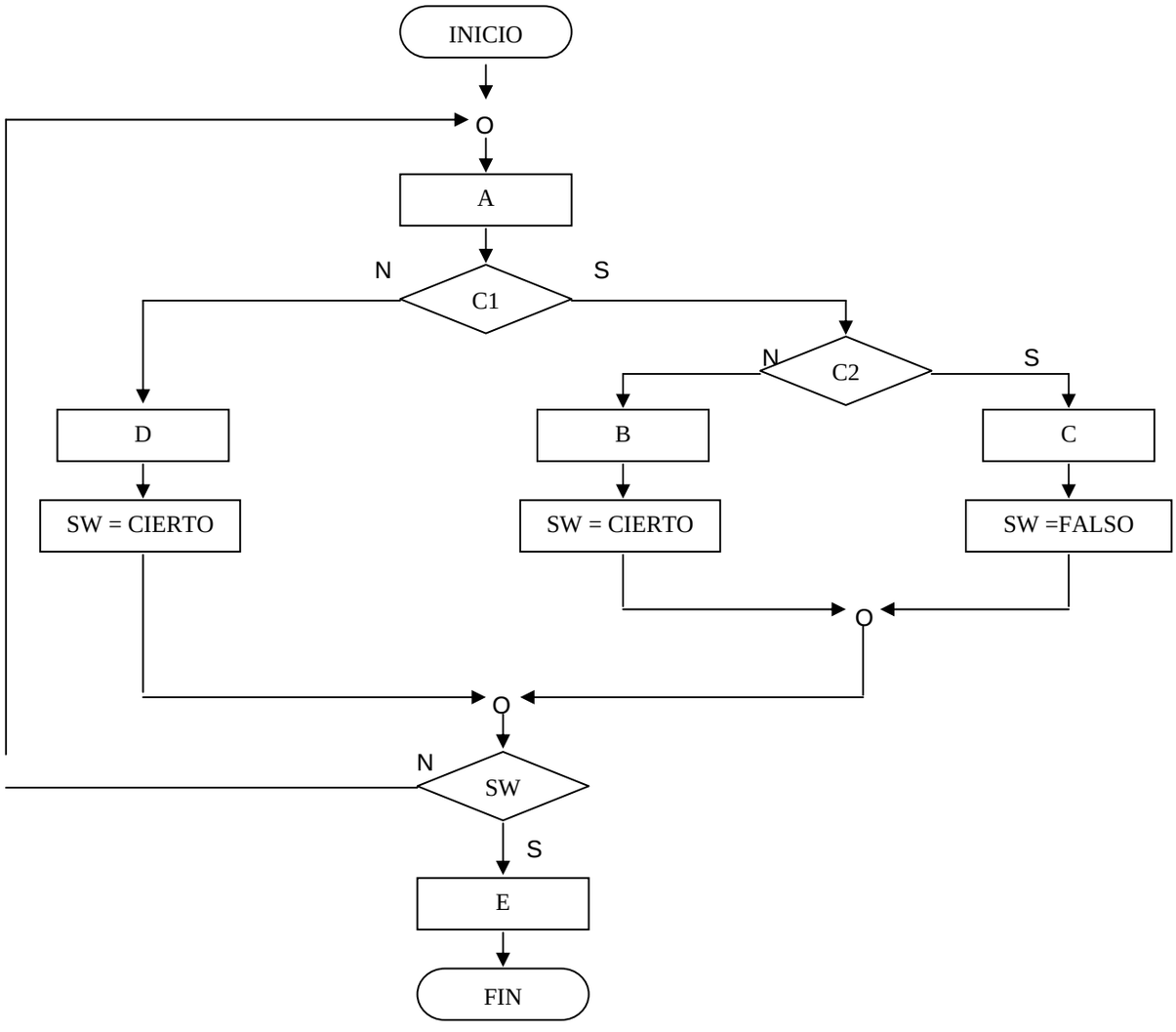
Ejemplo: Aplicación del teorema de la estructura a un algoritmo.

El siguiente programa propio no utiliza sólo estructuras básicas, como puede verse.

Encontrar un programa propio equivalente que sólo utilice dichas estructuras.



El equivalente estructurado, entre otros, puede ser:



HERRAMIENTAS DE LA PROGRAMACION ESTRUCTURADA

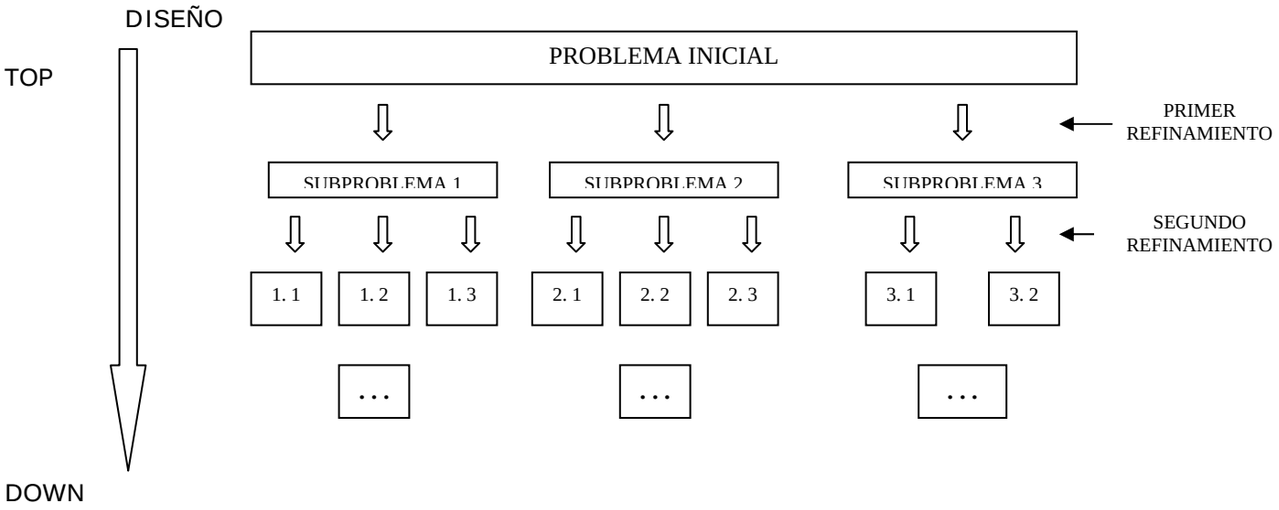
Además de elementos comunes en otros métodos de programación (objetos, variables auxiliares, operadores, etc.), la programación estructurada utiliza:

- Diseño descendente (TOP – DOWN)
- Recursos abstractos

- Estructuras básicas

♦ Diseño TOP – DOWN

Los programas se diseñan de lo general a lo particular por medio de sucesivos refinamientos o descomposiciones que nos van acercando a las instrucciones finales del programa.



♦ Utilización de recursos abstractos

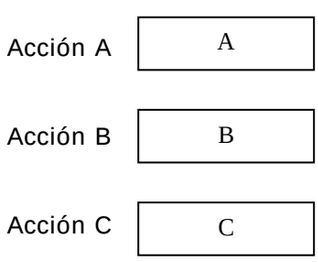
Es el complemento perfecto para el diseño TOP – DOWN donde se utiliza el concepto de abstracción: es decir, en cada descomposición se supone que todas las partes resultantes están resueltas, dejando su realización para el siguiente refinamiento y considerando que todas ellas pueden llegar a estar definidas en instrucciones y estructuras disponibles en los lenguajes de programación.

♦ Estructuras básicas

Como se indico anteriormente, el teorema de la estructura dice que toda acción se puede realizar utilizando tres estructuras básicas de control, la estructura secuencial, alternativa y repetitiva. Esta afirmación es cierta y demostrable.
Para la representación gráfica de las estructuras utilizaremos el concepto de acción cuyo significado es totalmente general.
Una acción puede representar:

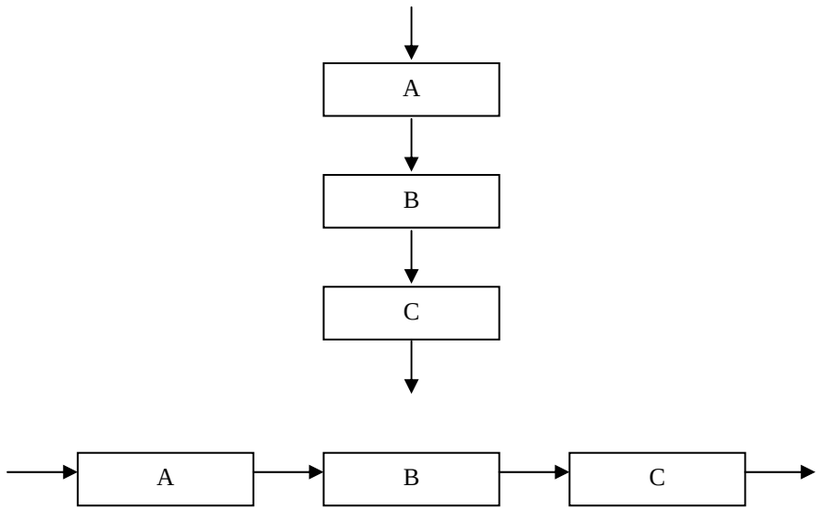
- Ninguna operación.
- Una operación sencilla; por ejemplo, el movimiento de un valor de un campo a otro, una operación de salida, etc.
- Un proceso de cualquier tipo; por ejemplo, una ordenación de datos.

Con ello interpretaremos que una determinada acción representada en una de las tres estructuras puede estar compuesta por una o más estructuras en su interior.
La notación utilizada para representar dicha acciones constara de rectángulos horizontales en cuyo interior pondremos letras mayúsculas.



♦ Estructura secuencial

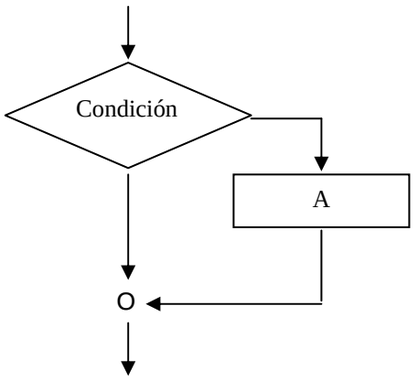
Es una estructura con una entrada y una salida en la cual figuran una serie de acciones cuya ejecución es lineal y en el orden en que aparecen. A su vez, todas las acciones atienen una única entrada y una única salida.



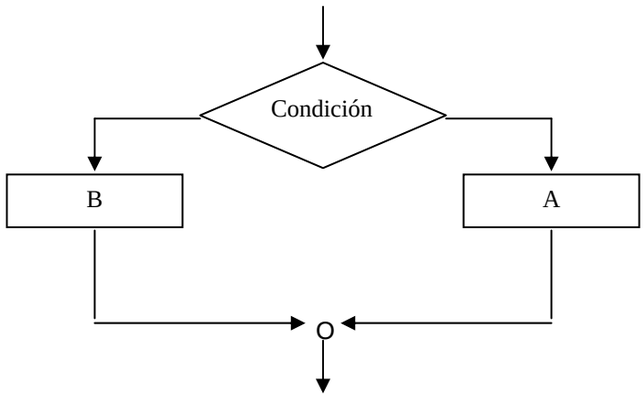
- **Estructura alternativa**

Es una estructura con una sola entrada y una sola salida en la cual realiza una acción de entre varias, según una condición, o se realiza una acción según el cumplimiento o no de una determinada condición. Esta condición puede ser simple o compuesta.
Las estructuras alternativas pueden ser:

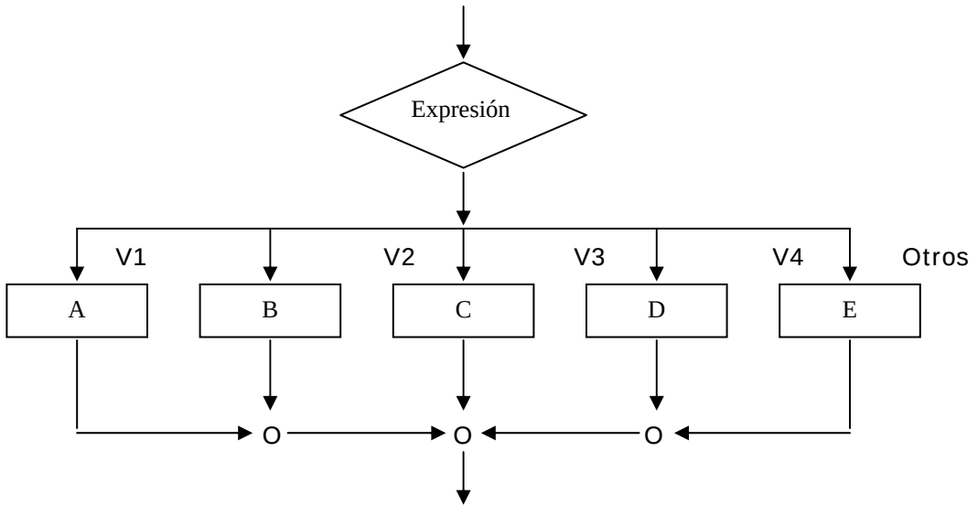
- De dos salidas, en la que una de ellas puede ser la acción nula.
- De tres o más salidas, que también se llama múltiple.



Alternativa Simple



Alternativa doble



Alternativa Múltiple

- **Estructura repetitiva**

Es una estructura con una entrada y una salida en la cual se repite una acción un numero determinado o indeterminado de veces, dependiendo en este caso del cumplimiento de una condición.
Las estructuras repetitivas pueden ser:

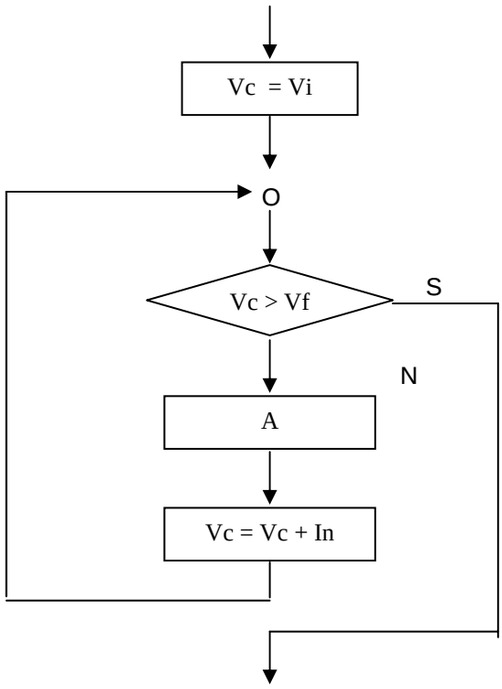
- Estructura para (FOR)
- Estructura mientras (WHILE)
- Estructura hasta (UNTIL)
- Estructura iterar (LOOP)

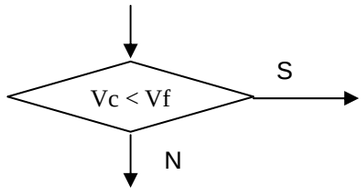
♦ Estructura PARA (FOR)

En esta estructura se repite una acción un numero fijo de veces representado normalmente por *N*. Es necesario para el control de la repetición utilizar una variable de control *Vc* y los valores que asignaremos a la misma inicialmente *Vi* y su correspondiente valor final *Vf*. El incremento de la variable de control *Vc* es normalmente 1, pero puede tomar otros valores positivos y negativos, en cuyos casos es necesario indicarlo por medio de *In*.
El numero de repeticiones *N* esta dado por la formula

$$N = \text{parte entera}\left[\frac{Vf - Vi}{In}\right] + 1$$

Si N = O se obtiene un valor negativo, se dice que el bucle es inactivo y no se repite ninguna vez.



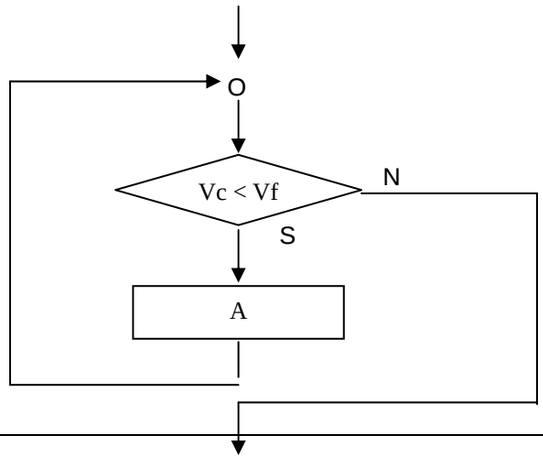


En este tipo de estructuras existen una serie de normas de obligado cumplimiento, como son:

- El *In* no puede ser 0 (bucle infinito).
- *Vc* no puede modificarse en el rango del bucle (acción A).

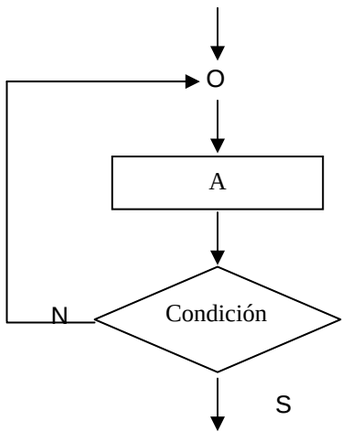
♦ Estructura MIENTRAS (WHILE)

En esta estructura se repite una acción mientras se cumpla la condición que controla el bucle. La característica principal de esta estructura es la de que la condición es evaluada siempre antes de cada repetición.
El numero de repeticiones oscila entre 0 e infinito, dependiendo de la evaluación de la condición, cuyos argumentos en los casos de repetición, al menos una vez, deberían modificarse dentro del bucle, pues de no ser así el numero de repeticiones seria infinito y nos encontraremos en un bucle sin salida.



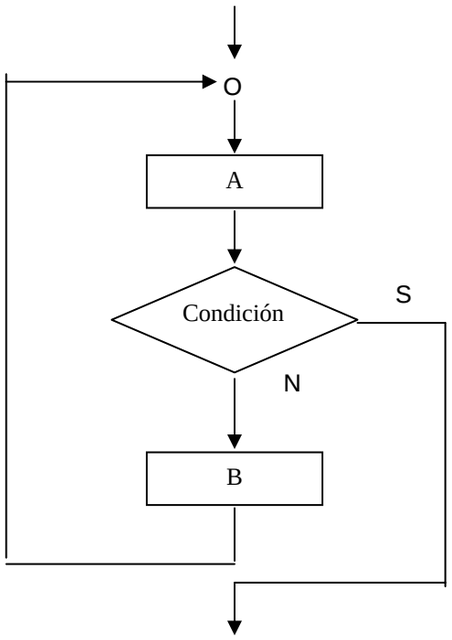
- **Estructura HASTA (UNTIL)**

En esta estructura se repite una acción hasta que se cumpla la condición que controla el bucle, la cual se evalúa después de cada ejecución del mismo.
El numero de repeticiones oscila entre 1 e infinito, dependiendo de la evaluación de la condición, cuyos argumentos en los casos de repetición, al menos dos veces, deberán modificarse dentro del bucle, pues de no ser así el numero de repeticiones será infinito y nos encontraremos en un bucle sin salida.



- **Estructura ITERAR (LOOP)**

En esta estructura se repiten alternativamente dos acciones, evaluando la condición de salida entre ambas.
El numero de repeticiones oscila, por la acción A, entre 1 e infinito, y para la acción B, entre 0 e infinito, cumpliéndose que siempre se repite A una vez más que B.
Los bucles anteriores son casos particulares de éste.



Ejemplo de ordinograma estructurado: