UNIDAD 1 (II). DIAGRAMAS DE FLUJO

ÍNDICE DE CONTENIDO

DIAGRAMAS DE FLUJO

1. INTRODUCCIÓN

Como ya vimos en la unidad anterior, los **diagramas de flujo** (u ordinogramas) son una forma gráfica de **representar algoritmos**. Dicho de otro modo, **representa las instrucciones paso a paso que se ejecutarían en un programa de ordenador.**

Son muy útiles para practicar y aprender a pensar de forma algorítmica antes de programar utilizando lenguajes de programación reales como Java, Python, etc. Los ordinogramas pueden crearse directamente con papel y lapiz o utilizar algún software como DIA, yEd o draw.io.

Veamos uno a uno los diferentes elementos de un ordinograma y cómo utilizarlos.

2. INSTRUCCIONES DE INICIO Y FIN

Todos los programas tienen un punto inical y un punto final que marcan cuándo empieza y acaba un programa. En un ordinograma estos se representa mediante **INICIO** y **FIN**.

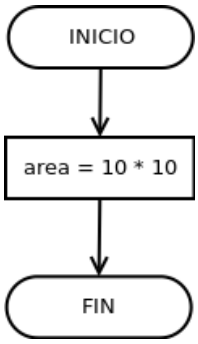
3. INSTRUCCIONES DE PROCESADO DE INFORMACIÓN

Todos los programas informáticos procesan información, es decir, realizan operaciones matemáticas para calcular, manipular o modificar información. Como ya vimos en la unidad anterior las operaciones pueden ser aritméticas (+ - * / %), relacionales (< <= > >= == <>) o lógicas (NOT, AND, OR). A su vez, se utilizan variables (A, B, Edad, Precio, etc.) para almacenar la información que estamos procesando.

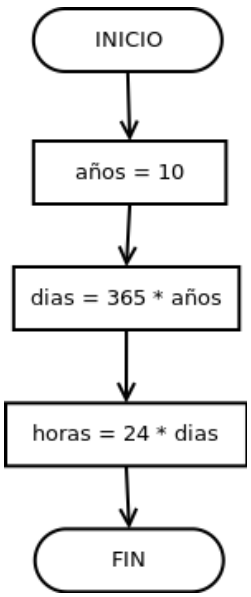
El encargado de realizar este tipo de operaciones es el procesador de un ordenador.

En un ordinograma se representan utilizando **rectángulos** que contienen las instrucciones.

Por ejemplo, el siguiente ordinograma calcula el área de un cuadrado de lado 10.

ORDINOGRAMA	EXPLICACIÓN
	Calcula 10 * 10 y guarda el resultado (100) en la variable **area**.

Podemos utilizar tantas instrucciones de procesado de información como necesitemos. Por ejemplo, el siguiente ordinograma calcula el número de horas que hay en 10 años.

ORDINOGRAMA	EXPLICACIÓN
	Guardamos 10 en la variable **años**. Calcula 365 * **años** (365 * 10) y guarda el resultado (3650) en la variable **dias**. Calcula 24 * **dias** (24 * 3650) y guarda el resultado (87600) en la variable **horas**.

Si se desea, es posible poner varias instrucciones dentro de un mismo rectángulo (para simplificar):

4. INSTRUCCIONES DE ENTRADA Y SALIDA DE INFORMACIÓN

Todos los programas informáticos utilizan algún tipo de entrada y salida de información. Es una parte muy importante de la programación ya que, entre otras cosas, es lo que permite a un usuario interactuar con un programa. Por ejemplo:

1. El usuario introduce información por teclado (entrada).
2. El programa procesa la información (hace algún cálculo).
3. El programa muestra el resultado por pantalla (salida).

Todos los programas de ordenador, apps de teléfono, páginas web, etc. siguen estos tres pasos. Tú apretas un botón (o haces click, tocas la pantalla...), luego el procesador lo procesa y por último sucede algo (se visita una página web, se escucha una canción, se envía un whatsapp, etc.).

No tendría sentido crear programas sin entrada ni salida ya que solo realizarían cálculos sin comunicarse con el exterior (como en los dos ejemplos del apartado anterior).

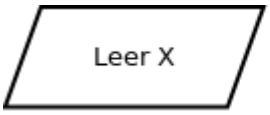
Algunos ejemplos de dispositivos utilizados para entrada y salida de información:

- Dispositivos de Entrada: teclado, ratón, micrófono, escáner, gps, wifi, etc.
- Dispositivos de Salida: pantalla, altavoces, impresora, wifi, etc.

Por ahora nos vamos a centrar en la entrada y salida más sencilla: el <u>teclado</u> y la <u>pantalla</u>.

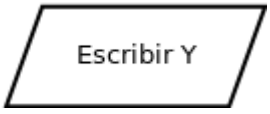
En los ordinogramas la entrada y salida de información se representa mediante un **paralelogramo**.

ENTRADA de información



El usuario introduce información por teclado y se guarda en la variable X.

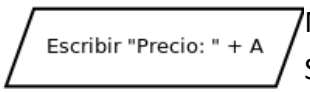
SALIDA de información



Muestra por pantalla el contenido de la variable Y.

La instrucción de SALIDA de información es muy versátil. También puede utilizarse para mostrar texto, además de combinaciones de texto y variables.

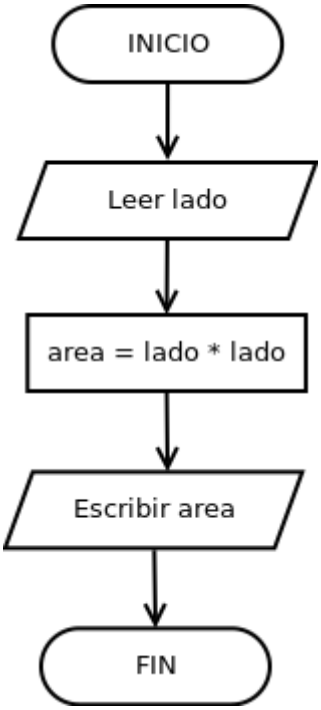
Muestra por pantalla el texto “Hola mundo!”



Muestra por pantalla el texto “Precio : ” seguido del valor de la variable A. Si por ejemplo A vale 15, mostrará el texto “Precio: 15”.

¿Recuerdas el primer ordinograma del apartado 3? Simplemente calculaba el área de un cuadrado de lado 5 y ni si quiera mostraba el resultado por pantalla…

Vamos a modificarlo añadiendo entrada y salida de modo que primero le pida al usuario que introduzca el lado del cuadrado, luego calcule el área, y por último la muestre.

ORDINOGRAMA	**EXPLICACIÓN**
	El usuario introduce un número por teclado (entrada) y se guarda en la variable **lado**. Calcula **lado * lado** y guarda el resultado en la variable **area**. Muestra por pantalla (salida) el contenido de la variable **area**.

Si creáramos un programa siguiendo este algoritmo mejorado (con entrada y salida) **nos permitiría utilizarlo para calcular el área de cualquier cuadrado**. Eso es mucho más útil y general que el programa original que solo calculaba el área un cuadrado de lado 5.

Por ello, **añadir entrada y salida a los programas es fundamental** para que sean de utilidad.

Ahora vamos a modificar el ordinograma que calculaba los días y horas para añadirle entrada y salida. Queremos que el programa le pida al usuario que introduzca por teclado los años, calcule los días y las horas, y lo muestre por pantalla.

ORDINOGRAMA	**EXPLICACIÓN**
	El usuario introduce un número por teclado (entrada) y se guarda en la variable **años**. Calcula 365 * **años** y lo guarda en la variables **dias**. Calcula 24 * **dias** y lo guarda en la variable **horas**. También se podría calcular directamente con *horas = años * 365 * 24* Muestra por pantalla (salida) el contenido de la variable **horas**.

De nuevo, si creáramos un programa que siguiera los pasos de este ordinograma, lo podríamos utilizar para calcular el número de horas de los años que quisiéramos, tantas veces como quisiéramos.

Ahora es un buen momento para hacer los **ejercicios del 1 al 7**.

5. ESTRUCTURAS DE CONTROL

Hasta ahora hemos visto algoritmos (representados como ordinogramas) en los que las instrucciones se ejecutan secuencialmente (una después de la otra). Pero a menudo es necesario diseñar algoritmos cuyas instrucciones no se ejecuten secuencialmente, para lo que es necesario utilizar estructuras de control.

Las estructuras de control son utilizadas para controlar la secuencia (el orden) en el que se ejecutan las instrucciones.

Existen dos tipos:

- **Estructuras alternativas**: Permiten alternan entre distintas instrucciones (ejecutar unas u otras) dependiendo de una condición. Pueden ser simples, dobles o múltiples.
- **Estructuras repetitivas**: Permiten repetir instrucciones (ejecutarlas varias veces).

En esta unidad nos vamos a centrar en las estructuras alternativas.

En la siguiente unidad veremos las estructuras repetitivas.

6. ESTRUCTURAS ALTERNATIVAS

Controlan la ejecución o la no ejecución de una o más instrucciones en función de que se cumpla una condición. Dicho de otra manera, **se utilizan para que sucedan cosas distintas dependiendo de una condición**.

Por ejemplo, al introducir una contraseña si esta es correcta se iniciará sesión, pero si es incorrecta mostrará un mensaje de error. Por poner otro ejemplo, cuando aprietas una letra o un número del teclado ésta se mostrará por pantalla, pero si es la tecla de intro entonces el cursor bajará a la siguiente línea.

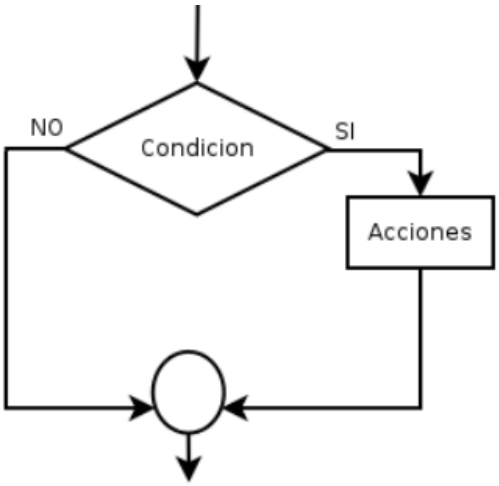
Puede parece obvio pero esto sucede así porque un programador ha utilizado una estructura alternativa para indicar exactamente qué debe suceder en cada caso. **Las estructuras alternativas son muy importantes para establecer distintos comportamientos a un programa.**

Existen tres tipos de estructuras alternativas: Simple, Doble y Múltiples.

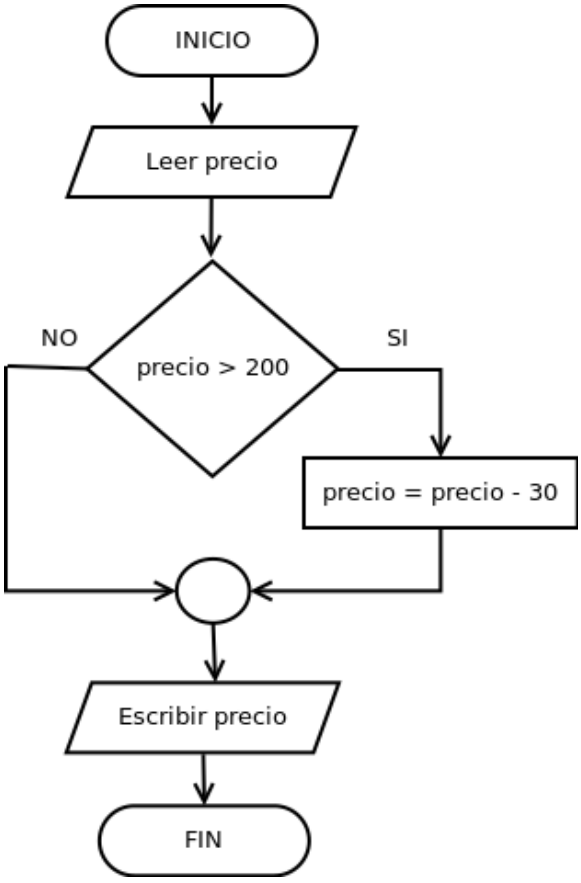
Todas ellas utilizan **condiciones**, como (*precio > 200)*, (*edad >= 18)*, (*contraseña == "1234")*, etc. En un ordinograma una condición se expresa mediante un **rombo**.

6.1 Estructura alternativa simple

La estructura alternativa simple es muy sencilla. Si la condición es verdadera se ejecutará una o varias instrucciones concretas, pero si es falsa éstas no se ejecutarán. Se representa así:

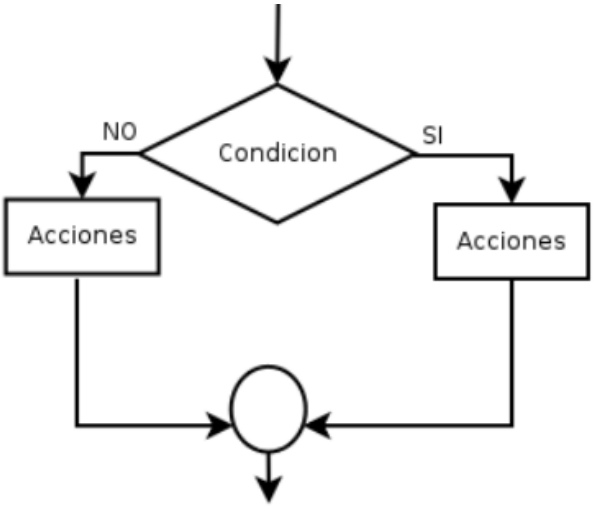


Veamos un ejemplo sencillo. Un programa que lee por teclado el precio inicial de un producto, si es superior a 200 € se le aplica un descuento de 30 €, y por último lo muestra por pantalla.

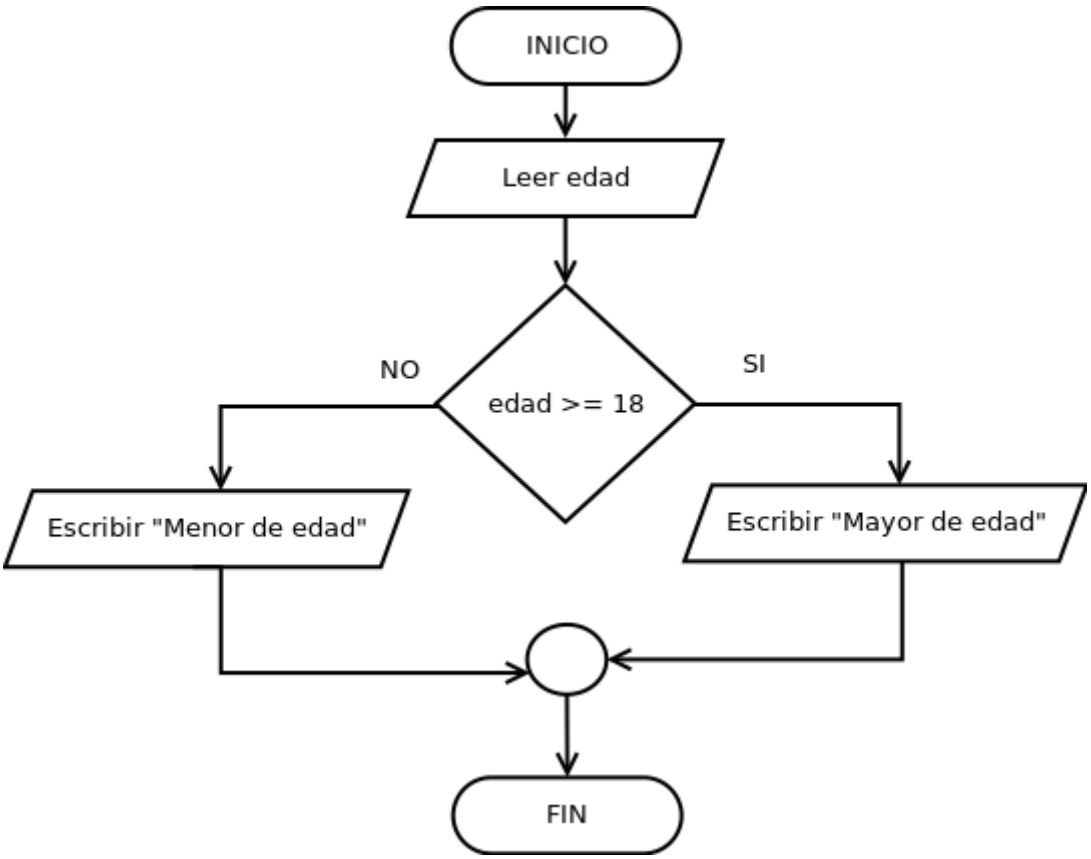


6.2 Estructura alternativa doble

La estructura alternativa doble es muy similar a la simple. La única diferencia es que si la condición es cierta se ejecutarán unas instrucciones y si es falsa se ejecutarán otras distintas.



Veamos un ejemplo. Un programa que pide la edad por teclado, si es mayor o igual a 18 mostrará por pantalla el texto “Mayor de edad”, en caso contrario mostrará “Menor de edad”.



6.3 Estructura de alternativa múltiple

En la estructura alternativa múltiple se pueden especificar distintas instrucciones dependiendo del valor de una expresión. ¡Ojo! de una expresión, no de una condición. La expresión dará un resultado concreto (por ejemplo 10, -5, 0.25, etc.) y se ejecutarán las instrucciones asociadas a cada valor.

Si el resultado de la expresión no es igual a ninguno de los valores indicados, se ejecutarán las instrucciones por defecto (si se ha indicado, esto es opcional).