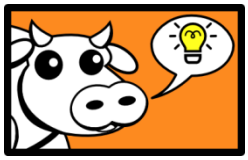
2 – Algoritmos en Pseudocódigo

2.1 – Estructura Básica de un Algoritmo Escrito en Pseudocódigo

Como se dijo en el capítulo anterior, para que sea considerado como escrito en pseudocódigo, un algoritmo debe cumplir con ciertos requisitos en su sintaxis. Estos requisitos tienen que ver con una cierta ***formalidad sintáctica*** que debe cumplirse para que el algoritmo sea relacionable, más adelante, con los lenguajes de programación en los que se pretenda implementar el mismo.

El pseudocódigo nos permite hacer el paso de las series de instrucciones en lenguaje cotidiano, que suelen contener ambigüedades y contradicciones, a una secuencia bien formada y validable de pasos únicos y ordenados. De esta manera es posible pasar del pseudocódigo directamente al lenguaje de programación, ya que no solo manejan la misma secuencia lógica, sino que también comparten, en gran medida, la misma estructura.

Para entender esta estructura, vamos a ver cómo se organiza la misma mediante algunos ejemplos muy sencillos que iremos complejizando de a poco.



NOTA: Todos estos ejemplos pueden realizarse en papel, con un editor de texto cualquiera, aunque lo ideal es realizarlo en el editor de ***PSeInt.***

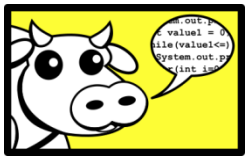
1. Inicio y Fin

Todo algoritmo escrito en pseudocódigo tiene bien definidos su inicio y su final. Esto también es cierto en los algoritmos definidos en lenguaje cotidiano, pero en este caso usaremos una sintaxis específica para expresarlo:

- Todo algoritmo comienza con la palabra **`Algoritmo`**, seguida del nombre del algoritmo.
  - El nombre del algoritmo no puede contener espacios, en su lugar pueden usarse guiones bajos entre las palabras (Ej.: varias_palabras_juntas) o escritura ***camelback***[2] (Ej.: variasPalabrasJuntas).
- Todo algoritmo finaliza con la orden **`FinAlgoritmo.`**

Por ejemplo, comencemos un algoritmo muy sencillo, el más sencillo, quizás, que puede diseñarse: el **holaMundo**, que es un algoritmo que lo único que hace es mostrar en pantalla la frase "Hola Mundo"**.**

Siguiendo las pautas establecidas más arriba, nuestro algoritmo se comienza a declarar de esta manera:



```
Algoritmo holaMundo

FinAlgoritmo
```

Entre las líneas que indican el comienzo y el fin del algoritmo colocaremos las instrucciones que componen nuestro algoritmo.

2. Sentencias

Para que un algoritmo tenga sentido, debe estar formado por instrucciones. En el contexto del pseudocódigo, y de la programación en general, a una instrucción concreta le llamamos ***sentencia***, y, por lo general, se declara y finaliza en **una línea única del algoritmo**.

[2] Se le dice escritura *camel case* o *camelback* a un estilo de escritura en el que no se utilizan espacios entre las palabras, sino que la separación entre las mismas se diferencia mediante el uso de mayúsculas en la primera letra, dejando el resto en minúsculas. El nombre proviene del hecho de que la "silueta" de la frase escrita queda con "jorobas" debido a las mayúsculas, recordando a la espalda de un camello dromedario. *Camel back* significa "espalda de camello" en inglés.

Por este motivo, la unidad más utilizada para medir la longitud y comenzar el análisis de la complejidad de un algoritmo es la cantidad de líneas que conforman dicho algoritmo.

A los efectos de nuestro curso, usaremos esta sintaxis para expresar sentencias:

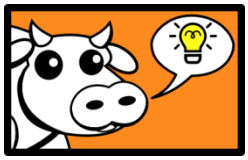
- Toda sentencia comienza con una orden o comando específicos, seguida de los parámetros y/o datos requeridos para su ejecución.
- Toda sentencia indica su finalización mediante el uso del símbolo ***punto y coma*** (;).

Con estas pautas le agregaremos a nuestro algoritmo una sentencia: la orden **Escribir,** que permite mostrar en pantalla el contenido de un texto o de una variable. En este caso le indicaremos que debe mostrar en pantalla la frase "Hola Mundo".

```
Algoritmo holaMundo

      Escribir "Hola Mundo";

FinAlgoritmo
```



NOTA: En la inmensa mayoría de los lenguajes de programación actuales, cuando se quiere indicar que un determinado bloque de texto es **solamente texto** y no contiene órdenes que deben interpretarse y ejecutarse, **se encierra dicho texto entre comillas dobles (" ") o comillas simples (' ')**.
De lo contrario el compilador o el intérprete del lenguaje que estemos usando intentará ejecutar **una a una las palabras del texto**, provocando un error de ejecución.

De esta manera nuestro algoritmo **holaMundo** queda completo y podemos ejecutarlo. Si lo ejecutamos en **PSeInt** obtendremos el siguiente resultado en nuestra pantalla: