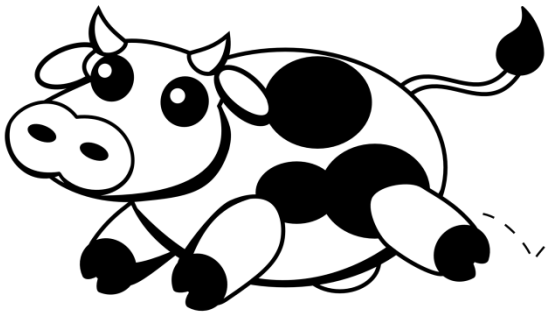
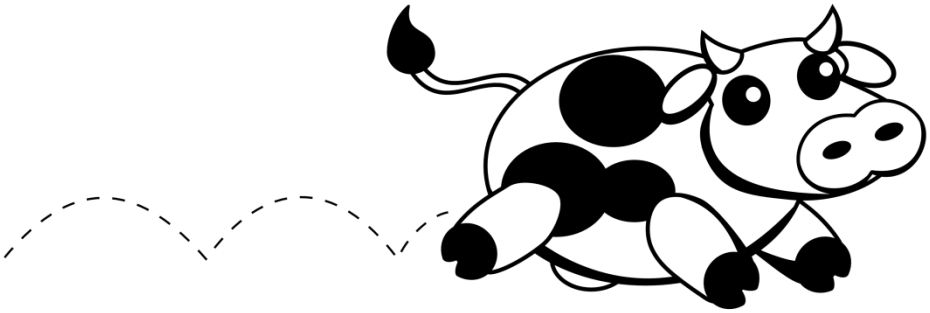
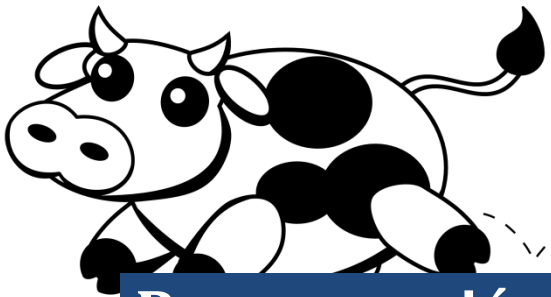






Programación Estructurada usando PSeInt y Java

Para 1° año de EMT de Informática de UTU/CETP

Luis Sebastián de los Angeles

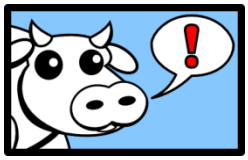
1 – Introducción a la Programación

1.1 - Los inicios de la Programación: Un Poco de Historia…

La programación tal como la conocemos hoy en día se ha desarrollado a partir de la década del año 1940 con el surgimiento de las primeras formas de lenguaje ensamblador (o ***assembler*** como se le conoce en inglés).

Sin embargo, las técnicas y conceptos asociados a la programación se han venido desarrollado desde hace siglos, conjuntamente con la lógica, la matemática y la ciencia en general.

La programación tiene, entre otros orígenes remotos, relación con las primeras formas de trabajo sistemático como ser la costura y el hilado.



Un avance clave en la historia de la programación se logró en 1725 con el desarrollo de telares semi-automatizados que se "programaban" con tarjetas perforadas que indicaban el patrón del tejido que se deseaba realizar.
Estos telares, desarrollados por el francés Basile Bouchon, fueron perfeccionados en 1804 por Joseph Marie Jaquard, que los hizo completamente automatizados, y dio origen a la tela que lleva su nombre.

Las matemáticas tienen mucho que ver en esta historia, ya que la idea de que es posible solucionar un problema en una serie de pasos concretos proviene de esta antigua ciencia. Son muy importantes los avances que realizarían en este sentido **Euclides** en Alejandría (s. IV AC) y **al-Khwārizmī** en Persia (s. IX DC, la palabra "algoritmo" deriva de su nombre).

Otros avances provendrían de **Leibniz**, en el s. XVII, quien propondría una idea revolucionaria llamada *Calculus Ratiocinator,* según la cual todo cálculo se puede reducir a elementos básicos que permitan su resolución por medios mecánicos**.**

En el siglo XIX, **Charles Babbage**, diseña sus máquinas diferenciales analíticas, las cuales usaban tarjetas perforadas para almacenar los procedimientos y también para expresar sus resultados. Con él trabajaba la condesa **Ada Lovelace**, diseñadora de lo que hoy en día se considera el primer programa de computadora: un algoritmo que permitía calcular la secuencia de números de Bernoulli.

Otro importante avance del siglo XIX fue el trabajo sobre lógica binaria realizado por **George Boole**, ahora conocida como álgebra de Boole, la cual es la base teórica para el desarrollo de la electrónica actual.

En el siglo XX, las guerras mundiales y sus carreras armamentísticas permitieron que matemáticos como **Alan Turing** y **John von Neumann** establecieran los modelos teóricos que permitieron construir las primeras computadoras electrónicas.

Los primeros lenguajes de programación propiamente dichos surgirán a mediados del siglo XX: muchos consideran que FORTRAN (1955), LISP (1958) y COBOL (1959) son los primeros lenguajes de "alto nivel", semejantes en su sintaxis al lenguaje humano.

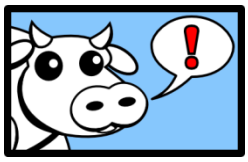
Hacia la década de los 60's surgen conceptos clave como ser la programación estructurada y las bases de la programación orientada a objetos. A partir de esta década los lenguajes comienzan a multiplicarse rápidamente debido a las nuevas aplicaciones que surgen para la informática en todos los niveles de la actividad humana.

1.2 - ¿Qué es Programar?

En términos simples, programar es **definir los pasos necesarios para realizar una tarea concreta**. Visto de esta forma, la programación no está limitada solamente al entorno de las computadoras, sino que **cualquier procedimiento sistemático que tiene como objetivo el solucionar un problema determinado puede llamarse programa**.

En términos de programación, llamamos a este proceso "*Creación de algoritmos*". Todo programa informático estará compuesto, por lo menos, por un algoritmo.

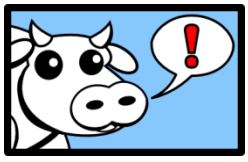
Un algoritmo es, entonces, una serie de instrucciones con un propósito determinado. Ahora bien, dicha serie de instrucciones debe cumplir con algunas restricciones para que se considere como un algoritmo formal:



Definición 1.1 : Características de un Algoritmo

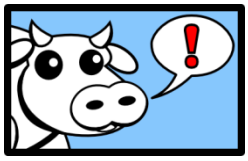
- **Específico**: cada algoritmo solo permite realizar una única tarea.
- **Ordenado:** las acciones que conforman al algoritmo deben ejecutarse en ese orden preciso, o el algoritmo no llegará al resultado esperado.
- **Restringido:** cada algoritmo posee restricciones que regulan su ejecución. Un algoritmo puede requerir **datos o condiciones de entrada** para su ejecución, y el resultado se devuelve en la forma de **datos o condiciones de salida**.

Para lograr esto con un resultado constante, se debe utilizar un lenguaje o una notación especializados para el diseño de algoritmos. Ese rol lo cumplen los ***Lenguajes de Programación***.



Definición 1.2 : Lenguaje de Programación:
Conjunto de reglas sintácticas (o sea, el orden que deben tener los elementos para tener sentido) y semánticas (el conjunto de expresiones que tienen significado en un determinado lenguaje) que permiten declarar algoritmos comprensibles por una computadora.

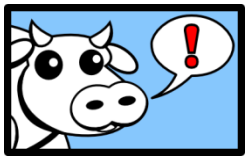
En general, cuando tenemos un algoritmo expresado en un determinado lenguaje de programación, decimos que tenemos un ***código fuente***, o código a secas.



Definición 1.3 : Código Fuente:
Conjunto de todos los algoritmos relacionados a un determinado programa, escritos en un determinado lenguaje de programación. Si se modifica el código fuente, el programa resultante del mismo también se habrá modificado.

Conviene aclarar que, si bien, el código fuente contiene todas las instrucciones que dan forma al programa, en general, **el código fuente en sí no es ejecutable.**

Esto se debe a que, en muchos lenguajes de programación, para obtener un archivo ejecutable (lo que en MS Windows solemos llamar un "*archivo punto exe*"), nuestro código fuente debe pasar por un **proceso de interpretación** llamado **compilación**, mediante un programa especializado llamado, justamente, **compilador**.



Definición 1.4 : Compilación:

Cada lenguaje de programación tiene, al menos, un compilador y linkeditor dedicados para generar sus ejecutables.

Otra particularidad del proceso de compilación es que, normalmente, al compilar un código fuente en un determinado sistema operativo, el ejecutable que obtenga al final del proceso, será compatible solo con ese sistema operativo. Esto no es una norma absoluta, ya que hay muchos compiladores que permiten generar ejecutables para varios sistemas operativos desde una misma plataforma, pero depende en gran medida del lenguaje de programación con el que estemos trabajando.

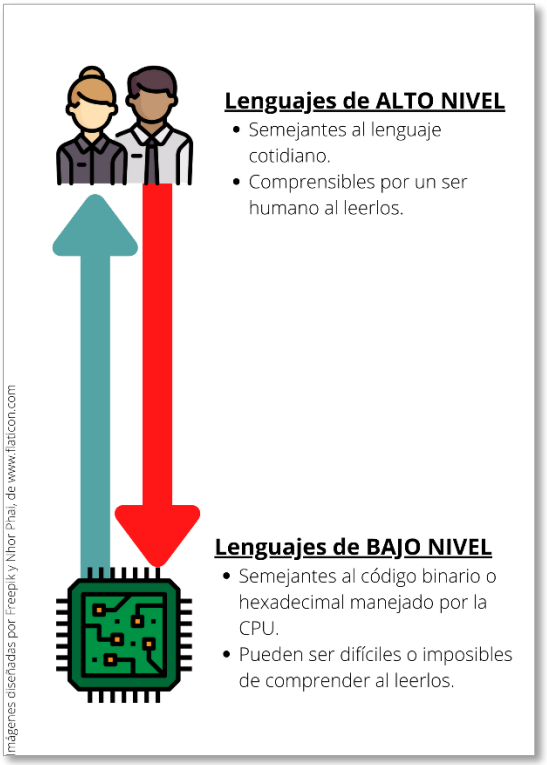
Existen, además, lenguajes de programación que no requieren compilación. Estos lenguajes son llamados "lenguajes interpretados" y dependen, para su ejecución, de un programa intérprete que analiza el código fuente y ejecuta las instrucciones que encuentra en el mismo.

Otra clasificación, muy importante, para clasificar un lenguaje de programación, tiene que ver con su "**nivel**". En la documentación muchas veces se habla de **lenguajes de "alto nivel" y lenguajes de "bajo nivel"**. Esta

nomenclatura puede resultar un tanto engañosa, ya que el concepto de nivel que se maneja en este caso es algo diferente a lo que se podría pensar.

Cuando se habla de nivel, en el contexto de lenguajes de programación, nos referimos a la semejanza que guarda dicho lenguaje con el lenguaje cotidiano utilizado por el usuario humano.

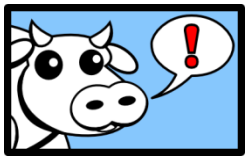
Cuanto más alto sea el nivel del lenguaje, más parecido será, en su sintaxis, al lenguaje cotidiano. De la misma manera, un lenguaje de bajo nivel es un lenguaje cuya sintaxis es muy parecida al código binario utilizado por el procesador mismo en su ejecución.



Los lenguajes con los que se suele programar aplicaciones comunes son de alto nivel, mientras que los lenguajes de bajo nivel se utilizan en casos en que es necesario programar bajo restricciones muy precisas del funcionamiento del hardware o intentando optimizar al máximo el uso de los recursos físicos del hardware.

1.3 El Pseudocódigo

Ahora bien, no existe un lenguaje de programación de mayor nivel que el pseudocódigo. El pseudocódigo es una forma de expresar un algoritmo en lenguaje cotidiano, pero con elementos y restricciones de forma que permiten, con muy poco esfuerzo, transformar el algoritmo escrito en pseudocódigo a la sintaxis de cualquier otro lenguaje de programación.



Definición 1.5 : Pseudocódigo:
Es una sintaxis que utiliza las convenciones estructurales de un lenguaje de programación, pero que no está pensado para su compilación o interpretación mediante una computadora, sino **para que sea leído y comprendido por un ser humano**.

No existe una sintaxis estandarizada del pseudocódigo, aunque hay algunas convenciones al respecto. En este material de estudio utilizaremos como base, para los algoritmos que trabajemos, la sintaxis utilizada por la herramienta ***PSeInt***[1].

[1]Una herramienta de desarrollo de algoritmos en pseudocódigo. Permite ejecutar el código escrito. Se puede obtener de forma gratuita en http://pseint.sourceforge.net/