# LENGUAJES DE PROGRAMACIÓN.

Para que la parte física de un sistema de computación (hardware) funcione es necesario utilizar programas (software), los cuales le indican cuál es la tarea que se tiene que hacer. Un lenguaje de programación es el que se utiliza para escribir dichos programas. Posteriormente estos se introducirán en la memoria de la computadora y éste último ejecutará todas las operaciones que se incluyen.

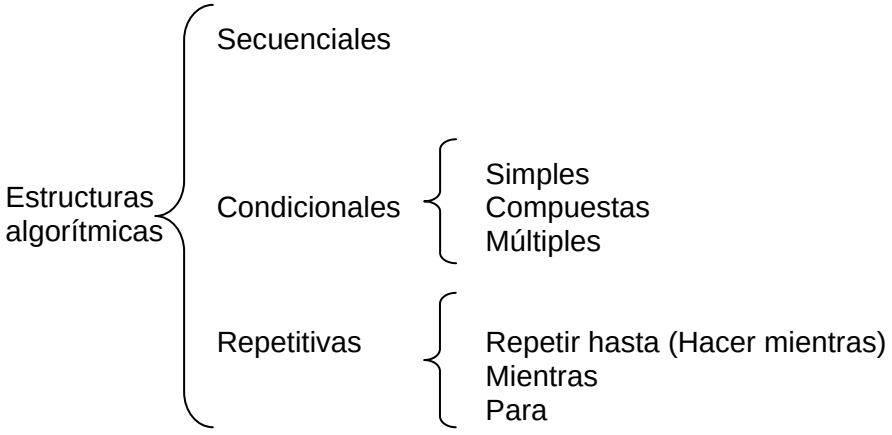
**Los lenguajes de programación constan de:**

a) Un conjunto finito de símbolos, a partir del cual se define el **léxico** o vocabulario del lenguaje.
b) Un conjunto finito de reglas, la **gramática** del lenguaje, para la construcción de las sentencias "correctas" del lenguaje. (**Sintaxis**).
c) **Semántica**, que asocia un significado (la acción que debe llevarse a cabo) a cada posible construcción del lenguaje.

Así, podemos decir que un lenguaje de programación consta de un conjunto de símbolos y un conjunto de reglas válidas para componerlos, de forma que formen un mensaje con significado para la computadora.

Para poder desarrollar los distintos programas se requieren de las estructuras algorítmicas para poder lograr el resultado deseado, teniendo en cuenta la toma de decisiones que debemos tomar para su desarrollo.

**Estas estructuras se clasifican de acuerdo con su complejidad en:**




**Estructura secuencial**: es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

**Estructuras condicionales**: comparan una variable contra otro(s) valor(es), para que, sobre la base del resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Existen dos tipos básicos, las simples y las múltiples.

**Estructuras repetitivas**: Se utilizan cuando se necesita resolver un problema que para resolverlo sea es necesario utilizar un mismo conjunto de acciones que se puedan ejecutar una cantidad específica de veces. Esta cantidad puede ser fija (previamente determinada por el programador) o puede ser variable (estar en función de algún dato dentro del programa).

## DEFINICIONES IMPORTANTES

**Programa:** Es el conjunto de instrucciones escritas de algún lenguaje de programación y que ejecutadas secuencialmente, resuelven un problema especifico.

**Lenguaje:** Es una serie de símbolos que sirven para transmitir uno o más mensajes (ideas) entre dos entidades diferentes. A la transmisión de mensajes se le conoce comúnmente como **comunicación**.

**Comunicación:** Es un proceso complejo que requiere una serie de reglas simples, pero indispensables para poderse llevar a cabo. Las dos principales son las siguientes:

- Los mensajes deben correr en un sentido a la vez.
- Debe forzosamente existir 4 elementos: Emisor, Receptor, Medio de Comunicación y Mensaje.

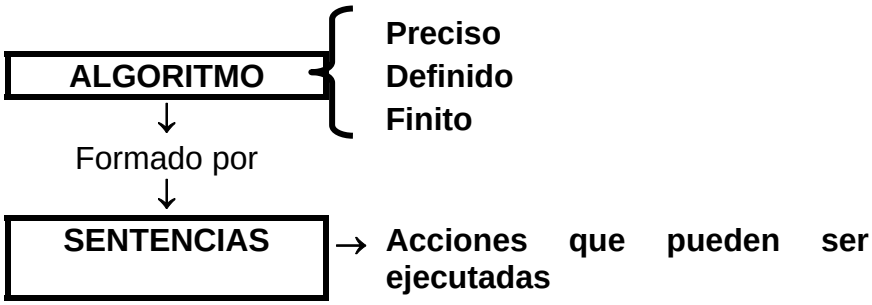
**Algoritmo:** La palabra algoritmo se deriva de la traducción al latín de la palabra árabe alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

Es un método para resolver un problema mediante una serie de pasos precisos, definidos y finitos.

- **Preciso:** No se presta a interpretaciones ambiguas.
- **Definido:** Si se siguen 2 o más veces los pasos, se obtiene el mismo resultado.
- **Finito:** Tiene comienzo y fin; tiene un número determinado de pasos.

Los algoritmos se pueden expresar en forma de diagramas de flujo, por fórmulas matemáticas y en Pseudocódigo.




**Tipos de algoritmos:**

- **Cualitativos:** Son aquellos en los que se describen los pasos utilizando palabras.
- **Cuantitativos:** Son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso.

## CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

- Según el **sistema de computación**: El **Lenguaje Máquina** es específico para cada máquina. Este lenguaje utiliza como código al sistema binario de numeración que consta de los símbolos “0" y “1".
    - Las órdenes que se dan a un sistema de computación han de ir codificadas en instrucciones, y estas forman los programas. Las instrucciones tienen dos partes diferenciadas: código de operación y código(s) de operando(s).
    - En la primera, se codifica la operación que realiza la instrucción. Este código de operación siempre es único para cada instrucción. En la segunda se indica(n) la(s) dirección(es) de memoria en la que se encuentra él operando, hasta un máximo de tres, sobre el/(los) que se aplicará la operación.
    - Puesto que cada tipo de sistema de computación tiene su código máquina específico, para programar en este lenguaje el programador debe conocer la arquitectura física de la computadora con cierto detalle (registros de la CPU, palabras de memora,...).

- Según el **nivel de abstracción**, o sea, según el grado de cercanía a la máquina:
  - Lenguajes de **bajo nivel**: La programación se realiza teniendo muy en cuenta las características del procesador, por cada instrucción en este de lenguaje existe una instrucción en lenguaje de máquina asociada. Ejemplo: Lenguaje ensamblador (Assembler).
  - Lenguajes de **nivel medio**: Permiten un mayor grado de abstracción pero al mismo tiempo mantienen algunas cualidades de los lenguajes de bajo nivel. Ejemplo: El lenguaje C puede realizar operaciones lógicas y de desplazamiento con bits, tratar todos los tipos de datos como lo que son en realidad a bajo nivel (números). Otras de las características que marcan una diferencia con los lenguajes de bajo nivel, es que la mayoría de las instrucciones son funciones, y cada función es en sí un pequeño programa que consta de un juego de instrucciones generalmente de bajo nivel.
  - Lenguajes de **alto nivel**: Más parecidos al lenguaje humano. Manejan conceptos, tipos de datos, etc., de una manera cercana al pensamiento humano ignorando (abstrayéndose) del funcionamiento de la máquina. Ejemplos: Java, Ruby.
- Según el **propósito,** es decir, el tipo de problemas a tratar con ellos:
  - Lenguajes de propósito **general**: Aptos para todo tipo de tareas: Ejemplo: C.
  - Lenguajes de propósito **específico**: Hechos para un objetivo muy concreto. Ejemplo: Csound (para crear ficheros de audio).
  - Lenguajes de **programación de sistemas**: Diseñados para realizar sistemas operativos o drivers. Ejemplo: C.
  - Lenguajes de **script**: Para realizar tareas varias de control y auxiliares. Antiguamente eran los llamados lenguajes de procesamiento por lotes (Batch) o JCL (”Job Control Lenguages”). Se subdividen en varias clases (de shell, de GUI, de programación web, etc.). Ejemplos: bash (shell), Lingo (Macromedia Director), mIRC script, JavaScript (programación web).
- Según la **evolución histórica**: Se va incrementando el nivel de abstracción, pero en la práctica, los de una generación no terminan de sustituir a los de la anterior:
  - Lenguajes de **primera generación**: Código máquina.
  - Lenguajes de **segunda generación**: Lenguajes ensamblador.

- Lenguajes de **tercera generación**: La mayoría de los lenguajes modernos, diseñados para facilitar la programación a los humanos. Ejemplos: C, Java.
- Lenguajes de **cuarta generación**: Diseñados con un propósito concreto, o sea, para abordar un tipo concreto de problemas. Ejemplo: NATURAL, Mathematica.
- Lenguajes de **quinta generación**: La intención es que el programador establezca el qué problema ha de ser resuelto y las condiciones a reunir, y la máquina lo resuelve. Se usan en inteligencia artificial. Ejemplo: Prolog.

- Según la **manera de ejecutarse**:
  - Lenguajes **compilados**: Un programa traductor traduce el código del programa (código fuente) en código máquina (código objeto). Otro programa, el enlazador, unirá los archivos de código objeto del programa principal con los de las librerías para producir el programa ejecutable. Ejemplo: C.
  - Lenguajes **interpretados**: Un programa (intérprete), ejecuta las instrucciones del programa de manera directa. Ejemplo: Lisp, java ...

- Según la **manera de abordar** la tarea a realizar:
  - Lenguajes **imperativos**: En ciencias de la computación se llama **lenguajes imperativos** a aquellos en los cuales se le ordena a la computadora cómo realizar una tarea siguiendo una serie de pasos o instrucciones, por ejemplo:
    - Paso 1, solicitar número.
      Paso 2, multiplicar número por dos.
      Paso 3, imprimir resultado de la operación.
      Paso 4, etc.
    - Algunos ejemplos de lenguajes imperativos son: BASIC, C, C++, Java, Clipper, Dbase, C# y Perl.
  - Lenguajes **declarativos**: Se les conoce como lenguajes declarativos en ciencias computacionales aquellos lenguajes de programación en los cuales se le indica a la computadora que es lo que se desea obtener o que es lo que se está buscando, por ejemplo: Obtener los nombres de todos los empleados que tengan más de 32 años. Algunos ejemplos de lenguajes declarativos son el Datatrieve, SQL y las expresiones regulares.

- Siglas de **Structured Query Language** (**Lenguaje Estructurado de Consultas**). Es un lenguaje declarativo que aúna características del Álgebra y el Cálculo Relacionales que nos permite lanzar consultas contra una Base de Datos para recuperar información de nuestro interés, almacenada en ella.
- Ejemplos de consultas SQL:
  - **SELECT** Nombre **FROM** Datos Personales **WHERE** Edad >=18**;**
  - Muestra el Campo "Nombre" de todos los individuos mayores de 18 años de la tabla "Datos Personales"

- Según el **paradigma de programación:** Un paradigma de programación provee (y determina) la visión y métodos de un programador en la construcción de un programa o subprograma. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de problemas (con la solución de múltiples “problemas” se construye una aplicación). Los principales son:
  - Lenguajes de **programación procedural**: Divide el problema en partes más pequeñas, que serán realizadas por subprogramas (subrutinas, funciones, procedimientos), que se llaman unas a otras para ser ejecutadas. Ejemplos: C, Pascal.
  - Lenguajes de **programación orientada a objetos**: Crean un sistema de clases y objetos siguiendo el ejemplo del mundo real, en el que unos objetos realizan acciones y se comunican con otros objetos. Ejemplos: C++, Java.
  - Lenguajes de **programación funcional**: La tarea se realiza evaluando funciones, (como en Matemáticas), de manera recursiva. Ejemplo: Lisp.
  - Lenguajes de **programación lógica**: La tarea a realizar se expresa empleando lógica formal matemática. Expresa qué computar (calcular). Ejemplo: Prolog.
- En sistemas distribuidos, según **dónde se ejecute**:
  - Lenguajes de **servidor**: Se ejecutan en el servidor. Ejemplo: PHP es el más utilizado en servidores web.
  - Lenguajes de **cliente**: Se ejecutan en el cliente. Ejemplo: JavaScript en navegadores web.

- Según **admitan o no concurrencia** de procesos, esto es, la ejecución simultánea de varios procesos lanzados por el programa:
  - Lenguajes **concurrentes**: Ejemplo: Ada.
  - Lenguajes **no concurrentes**. Ejemplo: C.
- Según la **interactividad** del programa con el usuario u otros programas:
  - Lenguajes **orientados a sucesos**: El flujo del programa es controlado por la interacción con el usuario o por mensajes de otros programas/sistema operativo, como editores de texto, interfaz gráfica de usuario (GUI) o kernels. Ejemplos: Visual Basic, lenguajes de programación declarativos.
  - Lenguajes **no orientados a sucesos**: El flujo del programa no depende de sucesos exteriores, sino que se conoce de antemano, siendo los procesos batch el ejemplo más claro (actualizaciones de bases de datos, colas de impresión de documentos, etc.). Ejemplo: Lenguajes de programación imperativos.
- Según la **realización visual** o no del programa:
  - Lenguajes de **programación visual**: El programa se realiza moviendo bloques de construcción de programas (objetos visuales) en una interfaz adecuado para ello. No confundir con entornos de programación visual, como Microsoft Visual Studio y sus lenguajes de programación textuales (como Visual C#). Ejemplo: Mindscript.
  - Lenguajes de **programación textual**: El código fuente del programa se realiza escribiéndolo. Ejemplos: C, Java, Lisp.

## HISTORIA Y CARACTERÍSTICAS DE ALGUNOS LENGUAJES

| | |
|---|---|
| Ada | Objetos, Imperativo híbrido. http://labsopa.dis.ulpgc.es/ada/ |
| Smalltalk | Objetos puro. http://www.smalltalk.org/ |
| Java | Objetos, Imperativo híbrido. http://www.java.org/ |
| C++ | Objetos, Imperativo, Hipertexto híbrido. http://www.masternet.com.co/prod/delphi.htm |
| Pascal | Objetos, Imperativo híbrido busca un link en google, hay demasiados, el mejor es **Borland C**. |
| Delphi | Objetos, Imperativo híbrido. http://www.masternet.com.co/prod/delphi.htm |
| Ocaml | Objetos, Imperativo y Funcional híbrido http://www.ocaml.org/ |
| Haskell | Funcional puro http://www.haskell.org/ |
| Lisp | Funcional híbrido http://www.lisp.org/ |
| Prolog | Lógico puro http://www.prolog.org/ |
| Perl | Objetos, Imperativo, Hipertexto, Expresiones Regulares híbrido. http://www.perl.org/ |
| PHP : | Imperativo, Hipertexto, híbrido. http://www.php.org/ |
| SQL | Lenguaje Declarativo, Expresiones Regulares. http://www.sql.org/ |
| UML | Lenguaje Modelado. http://www.uml.org/ |

## SITIOS WEB UTILIZADOS PARA ESTE INFORME

http://qbitacora.wordpress.com/2007/09/21/clasificacion-de-lenguajes-de-programacion/
http://www.frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm
http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n
http://www.alegsaonline.com/art/13.php
http://wwwdi.ujaen.es/asignaturas/fundTopo/TEMA8.pdf