LENGUAJES DE PROGRAMACIÓN

- Los programas son conjuntos de instrucciones que se desarrollan según o utilizando un Lenguaje de Programación.
- Es legible por el usuario (programador).
- Es ininteligible para el ordenador.
- Necesita de un proceso de traducción: Lenguaje → Hardware
- Componentes:
  - Caracteres: Vocabulario o léxico → símbolos permitidos.
  - Sintaxis: son las reglas que combinan los caracteres → construcciones con los símbolos.
  - Semántica: son las reglas que definen los efectos de cualquier construcción.
  - Palabras reservadas.
- Tenemos distintos criterios para establecer distintas clasificaciones.

Según el nivel de abstracción

Lenguaje de bajo nivel

- Son instrucciones complejas e ininteligibles → secuencias de 0's y 1's.
- Es entendible por el ordenador (hardware).
- Es dependiente del hardware del ordenador (procesador, memoria, …).
- No son portables entre distinto hardware (clónico, Apple, etc.).
- Trabaja con los registros de Memoria Física.

Lenguaje de Medio Nivel o Ensamblador

- Está cercano al Hardware.
- Surge por la poca portabilidad del L. Máquina.
- Usa mnemotécnicos → Más legibles por el usuario.
- Se trabaja con los registros del procesador y direcciones físicas.
- No es fácil de programar.
- El programador además de conocer el lenguaje ha de conocer el HW.
- Para hacerlo legible al ordenador:
  - Se compila.
  - Se traduce al lenguaje máquina.

Lenguaje de Alto Nivel

- Son la mayoría de los lenguajes actuales.
- Más cercano al lenguaje humano, son más intuitivos.
- Se ayudan de librerías y funciones.
- Para ejecutarse necesitan de un traductor (intérprete o compilador)
- Son independientes de la máquina.
- Ejemplo: C++, Cobol, PHP, Java, …
- Pueden ofrecer un framework (entorno de trabajo o desarrollo):
  - La programación es más eficiente y rápida.
  - Formado por un conjunto de elementos personalizables y adaptables a las necesidades de la aplicación.
  - Ejemplo: Bootstrap.
    - Utilizado en Twitter.

- Cualquiera puede realizar páginas y aplicaciones web.

- Tenemos distintos tipos en función de la forma de ejecutarse:

Forma de ejecución

Compilados

- Necesitan de un traductor o compilador.
- Convierten de código fuente a código máquina.
- Más rápidos que los interpretados o virtuales.
- Si se producen errores hay que modificar y volver a compilar.
- Utilizan un enlazador o linker:
  - Unión (código objeto programa; código objeto de las librerías).

Interpretados

- Se carga en Memoria el intérprete.
- Lee las instrucciones + las interpreta + las ejecuta.
- Es más lento.
- No requiere volver a compilar ante un error.
- Ejemplo: Java → combina compilación e interpretación.

Paradigma de programación

- Paradigma = enfoque particular para construir el SW → reglas, patrones, estilos de programación, ….
- Un lenguaje de programación puede usar más de un paradigma → se usa el que interesa.
- Categorías:

Imperativos

- Instrucciones simples.
- Detallan como se manipula la información de la memoria.
- Sentencia principal → la asignación.
- Ejemplo: Basic, Java, Fortran, C++.
- Engloba: programación estructurada, modular y la POO.

Funcionales

- Se basan en el concepto de función y los argumentos a aplicar.
- Ejemplo: Lisp, Schema, ML.

Lógicos

- Los podemos ver como una BB. DD. De declaraciones lógicas → reglas.
- Ejecución: consiste en preguntar (BB. DD.) de forma interactiva.
- Ej: Prolog → sistemas expertos, prospecciones.

Estructurados

- Utilizan 3 construcciones lógicas → secuencial, condicional y repetitiva.
- De lectura y seguimiento fácil.
- Si es grande su lectura y manejo se complica.
- Se puede dividir en partes manejables o módulos (con sus E/ y S/):
  - Son independientes.
  - Hay uno raíz que controla al resto.

  - Permite un desarrollo más rápido.
  - Es reutilizable.
  - Ej. Pascal, Fortran, C, Modula-2.

Orientado a Objetos (O.O.)

- Un programa es visto como un conjunto de objetos.
- Objeto:
  - Es una estructura de datos + método u operaciones (comportamiento) que los manipulan.
  - Los datos son propiedades o atributos de los objetos.
  - Los métodos cambian las propiedades o atributos.
  - La comunicación se lleva a cabo mediante paso de mensajes.
- Clase:
  - Es una plantilla para crear objetos → instanciación.
  - Ejemplo: Java.
  - Se asemeja a un módulo (programación estructurada).
- Permite reutilizar código, trabajar en equipo y mantenimiento del software.
- Su programación no es tan intuitiva.
- Ej.: C++, Java, Ada, Smaltalk.

-

EL PROCESO DE TRADUCCIÓN/COMPILACIÓN (Rev)

- Pasan de un lenguaje de alto nivel a uno de bajo nivel (ensamblador o máquina).
- Tipos:
  - Intérpretes:
    - Traduce una línea y la ejecuta.
    - Tiene que estar en la Memo junto con el código fuente.
  - Compilador:
    - Solo está en la máquina de desarrollo.
    - El software generado solo funciona con un HW y SW determinado.
    - Hay que recompilar ante un cambio del HW o SW.
- Funcionamiento:
  - Se preprocesa el código → se genera un código intermedio.
  - El código intermedio se compila → código ensamblador, código objeto o código máquina.
  - El código objeto se enlaza con las librerías y se optimizan los programas.

OBTENER CÓDIGO EJECUTABLE

- Diseño:
  - Componente SW <-> nivel de detalle
  - Herramientas: {Pseudocódigo | DFD}
  - Se codifica con un lenguaje de programación
- Codificación: se genera el código fuente.
- Tipos de código:
  - Fuente:
    - Escrito por el programador.
    - En lenguaje de alto nivel.
    - Mediante editor de texto o herramienta de programación.
    - Se parte de lo pseudocódigos o DFD.
    - No se ejecuta por el ordenador.
  - Objeto:
    - Resultado de compilar el código fuente.
    - No se ejecuta en el ordenador.
    - Ininteligible por el humano.
    - Representación a bajo nivel.
  - Ejecutable:
    - Resultado de enlazar: código objeto + rutinas + librerías.
- Compilación:
  - Se utilizan 2 programas.
    - Compilador:
      - Si existe un error en la compilación → no se genera el código objeto → modificar el código fuente + compilar.
      - Etapas:
        - Análisis léxico.
        - Análisis sintáctico.
        - Análisis semántico.
        - Genera código intermedio.
        - Optimizar código.
        - Generar código.