

1. INTRODUCCIÓN AL ECOSISTEMA DE LA PROGRAMACIÓN

1.1 ¿Qué es un lenguaje de programación?

- CM C++, Java, JavaScript, HTML
- Reglas gramaticales bien definidas que nos permiten escribir instrucciones a las que llamamos algoritmos; Estas instrucciones harán aquellas tareas que nosotros queremos que realice un pc, móvil, cualquier pieza que se pueda programar (no necesariamente informático)

1.2 Tipos de paradigmas de programación

Imperativos vs Declarativos

Imperativo:

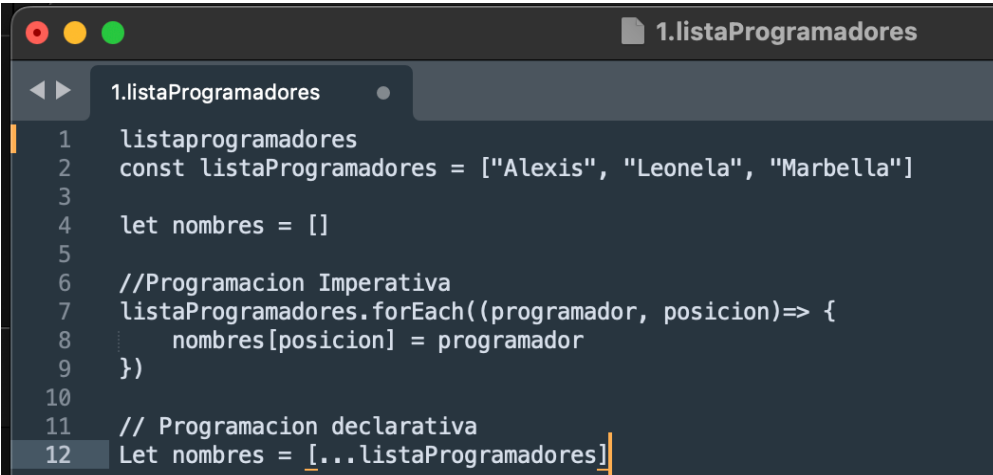
Es el paradigma más antiguo que tenemos y se trata de ir definiendo una serie de instrucciones, una secuencia de instrucciones que van definiendo paso a paso de lo que queremos que haga un sistema. Ejemplo

1. Paso: Carga el navegador "Chrome"
2. Paso: Escribe en la barra de navegación "google.com"
3. Paso: Busca "como aprender a programar JavaScript", etc
4. Paso: Has clic en buscar...

Declarativo:

Se centra en el qué en lugar del cómo, básicamente en el cual será el resultado final que queremos obtener y después el sistema se encarga de ir llegando hasta esa situación. Ejemplo: En una receta de comida el imperativo serían los pasos seguir (la receta) y el declarativo sería la foto final de la receta terminada.

Ejemplo IMPERATIVO y DECLARATIVO JAVASCRIPT



```
1.listaProgramadores

1  listaprogramadores
2  const listaProgramadores = ["Alexis", "Leonela", "Marbella"]
3
4  let nombres = []
5
6  //Programacion Imperativa
7  listaProgramadores.forEach((programador, posicion)=> {
8    nombres[posicion] = programador
9  })
10
11 // Programacion declarativa
12 Let nombres = [...listaProgramadores]
```

Ventajas e inconvenientes

Programación Imperativa:

legibilidad y facilidad de aprendizaje: es más fácil aprender una secuencia lógica de instrucciones mas que una serie de operadores especiales que nos permiten utilizar la programación declarativa. Es más sencillo aprender el paso a paso que las funciones específicas que nos servirían para la programación declarativa, pero es un código más difícil de mantener.

Programación declarativa:

Código más avanzado, pero más fácil de mantener.

1.2 Paradigmas de programación

Funcionales vs Procedimentales

Procedimentales:

Consiste en algo similar al paradigma de programación imperativo que sería ir línea por línea dando instrucciones de que hacer (se necesitaría especificar cada vez que necesitemos una función)

Funcional:

Se encarga en ir creando una serie de funciones, una serie de bloques de código en los cuales nosotros introducimos un parámetro y obtenemos otro, para en base a esos bloques de código ya predefinidos ir creando nuestro programa. (Parámetro predefinido, ahorra tiempo y espacio a largo plazo)

Ejemplo JavaScript

```
2.procedimental vs funcional
1 // Suma Procedimental *La procedimental especifica lo que el codigo ejecutara en 1 ocasion
2 let suma = 0
3
4 for (let i = 1; i <= 10; i++) {
5     suma = suma + i
6 }
7
8 // suma = 55
9
10 ///////////////////////////////////////////////////
11
12 // Suma funcional *Se define una funcion que podremos ejecutar las veces que queramos a lo
13 // largo del programa sin necesidad de volver a escribir todo el codigo
14 // PRIMERO - Definimos la funcion
15 function sumar_los_diez_primeros_enteros() {
16     let suma = 0
17
18     for (let i = 1; i <= 10; i++) {
19         suma = suma + i
20     }
21
22     return suma
23 }
24
25 // 2do - Utilizamos la funcion
26 let suma = sumar_los_diez_primeros_enteros()
27 // suma = 55
```

1.3 Niveles de lenguajes de la programación

Los niveles de programación se dividen por la especificidad de cada lenguaje. Cuanto más bajo el nivel más características específicas podemos tocar en la “maquina”.

0. Lenguaje máquina: Lenguaje binario, comunicación básica de una maquina (ilegible)
1. Lenguaje ensamblador: Lenguaje intermedio entre humano-máquina (fue el primer lenguaje de programación).
2. Bajo nivel: El lenguaje máquina y ensamblador son de bajo nivel porque están relacionados directamente con el hardware y la arquitectura.
3. Medio bajo-nivel: Surge “C”, tiene capacidades de alto y bajo nivel; tiene una sintaxis muy humana.
4. Medio-alto nivel: Todos los lenguajes actuales como Java, JavaScript, PHP, Python, etc.
5. Alto nivel: Frameworks basados en lenguajes de medio nivel; son una serie de librerías para facilitar el desarrollo, incluye distintos lenguajes.

1.4 Proceso de conversión

¿Qué es un traductor?

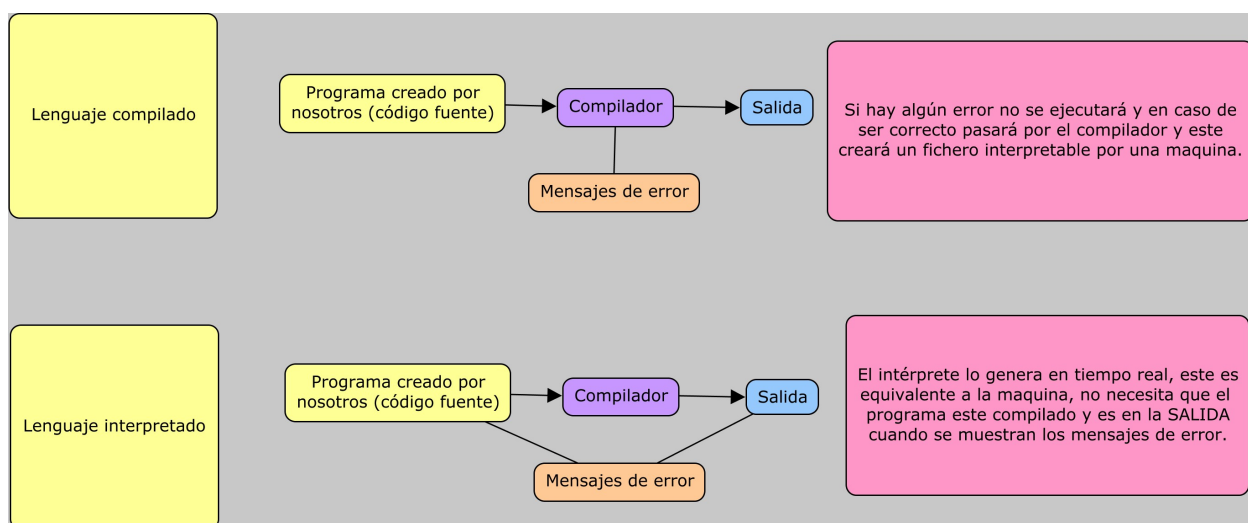
Programa que convierte el lenguaje de programación en binarios para ser entendido por una computadora.

¿Qué es un compilador?

Programa que “compila” un lenguaje como C++ para que una maquina lo entienda.

¿Qué es un intérprete?

JavaScript o Python



1.5 IDE (Entorno de Desarrollo Integrado)

Programas con características y capacidades que nos ayudan a escribir código.

Aplicaciones para desarrollo de código (programación)

- Visual Studio Code (El mejor para desarrollo web)
- Atom
- Sublime text
- PyCharm (muy bueno para desarrollar Python)
- Notepad++

1.6 Control de versiones (imprescindible para programar)

Hacer copias de seguridad por sí el archivo que editamos se corrompe y tener un control (solo si programamos nosotros).

La mejor opción es usar servicios de nube para modificar ficheros, como “Git”

CVS (Current Version System)

fue un sistema para ayudar a programadores a lidiar con el control de versiones.

SVN (SubVersion)

Software con mejoras de CVS

Git

Sistema moderno de gestión de versiones, tiene 2 versiones

Github

Sirve para crear repositorios públicos donde puedes consultar archivos nuevos y viejos (sirve como punto de restauración y partida)

Gitlab

Se centra en repositorios privados.

