Lenguajes del Programación
Unidad 1: Introducción

Clasificación de los Lenguajes de Programación

Manuel Soto Romero*

Semestre 2026-1
Facultad de Ciencias UNAM

En esta nota vamos a explorar cómo se pueden clasificar los lenguajes de programación desde tres perspectivas fundamentales. Primero, revisaremos la clasificación por nivel de abstracción, entendiendo qué significa que un lenguaje sea de bajo, medio o alto nivel y cómo esto influye en su cercanía al hardware. Después, veremos la clasificación según su propósito, distinguiendo entre lenguajes de propósito general y lenguajes diseñados para tareas específicas, con ejemplos claros de cada caso. Finalmente, abordaremos la clasificación por estilo o paradigma de programación, analizando las diferencias entre los enfoques imperativo y declarativo, así como sus subestilos más representativos. Esta nota nos dará un panorama general que servirá de base para comprender mejor el papel y las características de los distintos lenguajes que iremos estudiando a lo largo del curso.

1. Clasificación de acuerdo al nivel de abstracción

La clasificación de lenguajes de programación según el nivel de abstracción se refiere a cómo estos lenguajes se relacionan con el hardware subyacente y qué tan cercanos o alejados están de las operaciones de la máquina. Esta clasificación se puede dividir principalmente en tres categorías: (1) lenguajes de bajo nivel, (2) lenguajes de nivel medio y (3) lenguajes de alto nivel.

Lenguajes de bajo nivel Estos lenguajes se encuentran muy cerca del hardware y proporcionan poca abstracción de los detalles de la máquina. Aquí podemos encontrar algunos de los siguientes lenguajes:

- **Lenguaje de máquina:** Es el nivel más bajo de programación. Consiste en instrucciones codificadas en binario que el procesador puede ejecutar directamente. Cada tipo de procesador tiene su propio conjunto de instrucciones de máquina.

- **Lenguaje ensamblador:** Proporciona un nivel de abstracción ligeramente superior al lenguaje de máquina. Usa mnemotécnicos (palabras abreviadas) en lugar de códigos binario, lo que facilita la programación. Cada instrucción de ensamblador se traduce directamente a una instrucción de máquina. Ejemplos incluyen el ensamblador x86 y ARM.

Lenguajes de nivel medio Estos lenguajes ofrecen un balance entre abstracción y control sobre el hardware. Permiten manipular directamente la memoria y realizar operaciones de bajo nivel, pero también incluyen características de lenguajes de alto nivel.

*Un agradecimiento a Juan Mario Sosa Romo por sus aportes durante la revisión de esta nota como parte de su servicio social.

Un clásico ejemplo de este tipo de lenguajes es C. Este lenguaje proporciona acceso a operaciones de bajo nivel como la manipulación de la memoria mediante apuntadores, pero también soporta estructuras de alto nivel como funciones y estructuras de datos complejas. C es considerado de nivel medio debido a esta capacidad de realizar operaciones al hardware al mismo tiempo que sigue siendo más legible y manejable que el ensamblador.

Lenguajes de alto nivel Estos lenguajes están más alejados del hardware y proporcionan una mayor abstracción, facilitando la programación y la comprensión del código.

Lenguajes como Python, Java y Ruby caen dentro de esta clasificación. Manejan detalles de bajo nivel como la administración de memoria automáticamente (por ejemplo, a través de recolectores de basura) y proporcionan características avanzadas como manejo de excepciones, estructuras de datos integradas y bibliotecas extensas. La sintaxis y las abstracciones de estos lenguajes están diseñadas para ser más comprensibles y fáciles de usar para las personas que programan.

La clasificación de lenguajes de programación de acuerdo al nivel de abstracción es importante para entender la relación entre el software y el hardware, permitiendo a las personas elegir las herramientas adecuadas para cada tarea y facilitando tanto el desarrollo eficiente de software como la educación en Ciencias de la Computación. Esto es todo lo que diremos sobre este tipo de clasificación, el curso de Compiladores ahonda más en estos temas.

2. Clasificación de acuerdo al propósito

La clasificación de lenguajes de programación según su propósito se refiere a la categorización de los mismos basándose en el tipo de problemas o tareas para los cuales están diseñados y optimizados. Esta clasificación ayuda a las personas que desarrollan software a seleccionar el lenguaje más adecuado para sus necesidades específicas. Los principales tipos de lenguajes según su propósito son:

Lenguajes de propósito general Estos lenguajes están diseñados para ser versátiles y aplicables a una amplia variedad de problemas y dominios. Son adecuados para desarrollar una gama diversa de aplicaciones, desde software de sistemas hasta aplicaciones *web* y juegos.

Ejemplos incluyen Python, Java, C++, JavaScript. Python es conocido por su simplicidad y uso en *scripting*, desarrollo *web*, y ciencia de datos. Java es ampliamente utilizado en aplicaciones empresariales y desarrollo de aplicaciones Android. C++ es utilizado para desarrollo de sistemas, juegos y aplicaciones de alto rendimiento. JavaScript es el estándar para desarrollo *web frontend*.

Lenguajes de propósito específico Estos lenguajes están diseñados para resolver problemas en dominios particulares o para tareas específicas. Su sintaxis y características están optimizadas para esas aplicaciones particulares.

Ejemplos incluyen SQL, HTML y MATLAB.

Observación 1.

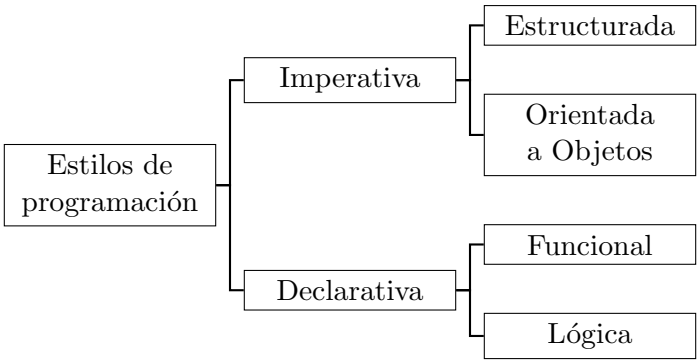
Es importante señalar que SQL y HTML no son considerados lenguajes de programación por gran parte de la comunidad. Sin embargo, ante la ausencia de una definición universalmente aceptada

de lo que constituye un lenguaje de programación, resulta válido estudiarlos bajo el mismo marco teórico. Además, no debe confundirse esta discusión con el concepto de lenguajes Turing-completos, que obedece a un criterio distinto de clasificación.

La clasificación de lenguajes según su propósito es fundamental para la Ciencias de la Computación y particularmente para la Ingeniería de Software, curso donde se profundiza más sobre estos temas, pues permite una selección más informada de herramientas, optimiza el desarrollo y mantenimiento del software, así como la enseñanza de la programación.

3. Clasificación de acuerdo al estilo

La clasificación de lenguajes de programación según su estilo (paradigma) se refiere a las diferentes filosofías y enfoques que estos lenguajes adoptan para estructurar y organizar el código. Los estilos de programación son modelos o patrones de diseño que determinan cómo se deben formular los problemas y las soluciones dentro de un lenguaje. Esta clasificación es crucial para entender cómo un lenguaje puede influir en la manera en que las personas piensan y resuelven problemas. El siguiente diagrama muestra una clasificación común de acuerdo al estilo.



Estilo de programación imperativa En este estilo, los programas se construyen mediante secuencias de instrucciones que cambian el estado del programa. Este enfoque se centra en describir cómo se debe hacer algo. Dentro de este estilo se encuentran dos "subestilos" ampliamente usados hoy en día:

- ⋆ **Programación Estructurada:** Organiza los programas al puro estilo imperativo mediante distintos tipos de estructuras: (1) secuencias, (2) estructuras de decisión y (3) estructuras de repetición. Promueve el uso de variables y referencias para cambiar y manipular datos en memoria. El mayor representante de este estilo: C.

- ⋆ **Programación Orientada a Objectos:** Organiza los programas en objetos, que son instancias de clases. Cada objeto contiene datos y métodos que operan sobre esos datos. La Programación Orientada a Objetos promueve la reutilización de código y la modularidad. Ejemplos clásicos: Java, C++ y Ruby.

Estilo de programación declarativa En este estilo, los programas describen qué se quiere lograr o qué son las cosas en lugar de cómo hacerlo. Los lenguajes declarativos se enfocan en la lógica del resultado deseado y el uso de definiciones y no en los pasos para obtenerlo. Dentro de este estilo se encuentran:

- ⋆ **Programación funcional:** Los programas se construyen mediante la aplicación de funciones. Se enfoca en el uso de funciones puras, sin efectos secundarios, y evita el uso de estados y datos mutables. Ejemplos clásicos: Haskell, Lisp y Scala.

- ⋆ **Programación lógica:** Se basa en la lógica matemática. Los programas se describen mediante un conjunto de hechos y reglas, y la computación se realiza mediante la aplicación de reglas de inferencia. El principal representante de este estilo: Prolog.

La clasificación de lenguajes de programación de acuerdo a su estilo proporciona una visión clara de las diferentes filosofías de diseño de software, lo que facilita la selección de lenguajes y técnicas apropiadas para resolver problemas específicos de manera eficiente y efectiva. Durante todo el curso estaremos mencionando distintas características de estos estilos y profundizaremos ampliamente en ellas.

4. Conclusión

En conclusión, comprender las distintas formas de clasificar los lenguajes de programación nos permite no solo identificar sus características técnicas, sino también valorar sus ventajas y limitaciones en distintos contextos de desarrollo. Estas clasificaciones (por nivel de abstracción, por propósito y por estilo) ofrecen un marco conceptual que facilita la elección del lenguaje adecuado para cada problema, fomenta una visión más crítica sobre las herramientas que utilizamos y fortalece nuestra capacidad para adaptarnos a diferentes paradigmas y entornos tecnológicos. Este conocimiento será un punto de referencia constante durante el curso y un recurso valioso en la práctica profesional.

Referencias

[1] S. Krishnamurthi, *Programming Languages: Application and Interpretation*. 2007, Disponible en línea.
[2] B. C. Pierce, *Types and Programming Languages*. MIT Press, 2002.
[3] M. Gabbrielli y S. Martini, *Programming Languages: Principles and Paradigms*. Springer, 2023.
[4] G. Winskel, *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, 1993.
[5] H. R. Nielson y F. Nielson, *Semantics with Applications: An Appetizer*. Springer, 2007.
[6] K. D. Lee, *Foundations of Programming Languages*. Springer, 2017.