Lenguajes del Programación
Unidad 1: Introducción

Historia de los Lenguajes de Programación

Manuel Soto Romero*

Semestre 2026-1
Facultad de Ciencias UNAM

De forma, intuitiva podemos decir que un lenguaje de programación es un conjunto de reglas y símbolos que permiten a las personas escribir instrucciones que una computadora puede entender y ejecutar. Estos lenguajes proporcionan una forma estructurada y comprensible de comunicarse con una máquina, permitiendo a los humanos especificar tareas y algoritmos de manera eficiente. Estos lenguajes, como veremos más adelante, pueden variar en su nivel de abstracción y propósito, desde lenguajes de bajo nivel como el lenguaje ensamblador, que están más cerca del lenguaje de la máquina, hasta lenguajes de alto nivel como Python o Java, que ofrecen una sintaxis más intuitiva y abstracta para expresar problemas y soluciones. También pueden clasificarse en diferentes estilos (o paradigmas), como la programación estructurada, funcional, orientada a objetos y lógica, cada estilo con sus propias características y formas de programación. En general, podemos ver a los lenguajes de programación como herramientas poderosas que nos permiten crear una amplia variedad de software, desde simples *scripts* hasta aplicaciones complejas y sistemas computacionales avanzados.

La teoría de lenguajes de programación se dedica al estudio de los principios fundamentales que subyacen a la creación, diseño, implementación y uso de los lenguajes de programación. Esta área de estudio se centra en comprender cómo funcionan los lenguajes de programación, cómo se pueden diseñar de manera efectiva para expresar problemas computacionales y cómo pueden ser analizados y evaluados desde una perspectiva teórica. La teoría de lenguajes de programación abarca una variedad de temas, como la sintaxis y semántica de los lenguajes, las gramáticas formales, la teoría de autómatas, los estilos de programación y la verificación de programas. Proporciona el marco conceptual necesario para entender y desarrollar lenguajes de programación, lo que permite expresar soluciones a problemas computacionales de manera clara y efectiva.

En esta nota de clase, daremos un breve repaso por la historia de esta teoría con el fin de entender qué se ha hecho, qué errores se han cometido y hacia donde vamos en el mundo de los lenguajes de programación.

1. Orígenes

Década de 1830

- ⋆ **1833.** Ada Lovelace colabora con Charles Babbage en el diseño de la Maquina Analítica y escribe el primer algoritmo diseñado para ser procesado por una máquina, lo que la convierte en la primera programadora de la historia

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

Década de 1930

- ⋆ **1936.** Alan Turing desarrolla la Máquina de Turing, sentando las bases de la teoría de la computabilidad
- ⋆ **1936.** Alonzo Church introduce el Cálculo λ como un sistema formal para describir la computación

Década de 1950

- ⋆ **1950.** Se publica el artículo *Computing Machinery and Itelligence* de Alan Turing, que propone el famoso *Test de Turing* para evaluar la inteligencia artificial
- ⋆ **1950.** Se desarrollan los primeros lenguajes ensambladores, que permiten una representación simbólica de las instrucciones de máquina, facilitando la programación a bajo nivel
- ⋆ **1954.** John Backus lidera el desarrollo de FORTRAN (*Formula Translation*), uno de los primeros lenguajes de programación de alto nivel
- ⋆ **1956.** Se celebra la conferencia de Darmouth, marcando el inicio del campo de la inteligencia artificial y el desarrollo de lenguajes como LISP

Década de 1960

- ⋆ **1960.** Peter Naur introduce el término *lenguaje de programación* en su libro *Programming Languages:*

  Un lenguaje de programación es un medio para expresar definiciones de procesos de manera que pueden ser ejecutadas por un procesador de datos
- ⋆ **1960.** ALGOL 60 (*Algorithmic Language 1960*) se establece como un estándar internacional para los lenguajes de programación
- ⋆ **1962.** John McCarthy desarrolla LISP (*List Processing*), uno de los primeros lenguajes de programación funcional
- ⋆ **1966.** Corrado Böhm y Guiseppe Jacopini plantean la idea de que cualquier programa puede ser escrito usando únicamente tres estructuras de control básicas: secuencia, decisión y repetición. Esta idea es conocida como el Teorema del Lenguaje Estructurado o Teorema de Böhm-Jacopini
- ⋆ **1968.** Edsger Dijkstra en su artículo *A Case against the GO TO Satatement* demostró que estas tres estructuras de control son suficientes para expresar cualquier algoritmo, proporcionando una base sólida para la programación estructurada

Década de 1970

- ⋆ **1972.** Dennis Ritchie desarrolla el lenguaje de programación C en los laboratorios Bell, lo que lleva a la propularización de la programación estructurada

Década de 1980

- ⋆ **1983.** Bjare Stroustrup desarrolla el lenguaje C++, una extensión del lenguaje C que introduce la programación orientada a objetos

- ⋆ **1987.** Se publica el estandar ANSI C, estableciendo una especificación para el lenguaje C
- ⋆ **1987.** Barbara Liskov introduce el principio de sustitución de Liskov, que establece requisitos para la subtipificación en sistemas de tipos polimórficos en su artículo *Data Abstraction and Hierarchy*

Década de 1990

- ⋆ **1991.** Guido van Rossum lanza Python, un lenguaje de programación de alto nivel conocido por su sintaxis clara y legible
- ⋆ **1995.** Brendan Eich desarrolla JavaScript en Netscape Navigator, transformando la web al permitir la interactividad del lado de las personas usuarias

Década de 2000

- ⋆ Se siguen desarrollando y refinando lenguajes de programación existentes, así como también surgen nuevos lenguajes para abordar desafíos emergentes en las ciencias de la computación como la inteligencia artificial, la computación cuántica y la criptografía y seguridad.

2. ¿Por qué estudiar Lenguajes de Programación?

Es esencial que las personas que se dedican a la ciencia de la computación conozcan la teoría de lenguajes de programación por varias razones importantes.

- ⋆ Proporciona fundamentos sólidos necesarios para comprender cómo funcionan los lenguajes de programación, incluyendo sus estructuras sintácticas, semánticas y de ejecución. Esta comprensión es esencial para escribir código eficiente, robusto y fácilmente mantenible.
- ⋆ Con una comprensión profunda de la teoría de lenguajes de programación, se pueden diseñar y evaluar nuevos lenguajes de programación o variantes de lenguajes existentes. Esto es útil para adaptar los lenguajes a necesidades específicas o para aprovechar nuevas tecnologías y estilos de programación.
- ⋆ Al entender los conceptos teóricos detrás de los lenguajes de programación, se puede optimizar el rendimiento del código mediante técnicas avanzadas de compilación, interpretación y ejecución. Esto es especialmente importante en aplicaciones críticas para el rendimiento, como sistemas embebidos o aplicaciones de alto rendimiento.
- ⋆ La teoría de lenguajes de programación proporciona herramientas y técnicas para abordar problemas complejos de manera efectiva y eficiente. Esto incluye el uso de estilos de programación específicos, como la programación funcional o la programación orientada a objetos, así como la aplicación de principios de diseño de lenguajes para resolver problemas de manera elegante y concisa.
- ⋆ Con una comprensión profunda de la teoría de lenguajes de programación se puede tener una mejor preparación para innovar y avanzar en el campo de las ciencias de la computación. Se puede contribuir al desarrollo de nuevos estilos de programación, herramientas de desarrollo de software más efectivas y lenguajes especializados para aplicaciones específicas.

3. Conclusión

En conclusión, el estudio de los lenguajes de programación no solo nos permite comunicarnos con las máquinas, sino que también nos ofrece una ventana al desarrollo histórico, conceptual y técnico de la computación misma. Comprender sus orígenes, desde Ada Lovelace hasta la creación de lenguajes modernos como Python y JavaScript, nos permite situar las herramientas actuales en un contexto más amplio de evolución tecnológica e intelectual. Más allá del uso práctico, conocer la teoría detrás de los lenguajes fortalece nuestras habilidades para escribir, analizar, optimizar e incluso diseñar nuevos lenguajes, lo cual resulta fundamental para enfrentar los retos actuales y futuros de la disciplina. Esta comprensión teórica es, en última instancia, una invitación a la creatividad computacional y a la reflexión crítica sobre cómo codificamos el mundo.

Referencias

[1] R. W. Sebesta, *Concepts of Programming Languages*, 11th. Pearson, 2016.
[2] B. C. Pierce, *Types and Programming Languages*. MIT Press, 2002.
[3] N. Nisan y S. Schocken, *The Elements of Computing Systems: Building a Modern Computer from First Principles*. MIT Press, 2005.
[4] J. Backus, "Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs", *Communications of the ACM*, vol. 21, n.º 8, págs. 613-641, 1978.
[5] P. E. Ceruzzi, *Computing: A Concise History*. MIT Press, 2012.
[6] B. Liskov, "Data Abstraction and Hierarchy", en *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages and Applications (Addendum)*, ép. OOPSLA '87, New York, NY, USA: ACM, 101987 de 1987.