Informe de Lenguajes, Paradigmas y Estándares de Programación

Alumno: Pedro José Gómez Reyes

Índice

Introducción

El presente informe trata de profundizar los conocimientos teóricos en materia de lenguajes de programación, los paradigmas que los caracterizan y los estándares de programación vigentes actualmente.

Antes de comenzar a entrar en materia, sería conveniente definir qué es un lenguaje de programación, o al menos dar una serie de pinceladas que nos permita elucubrar una serie de características que son propias de los lenguajes de programación.

Como suele ser frecuente en el mundo académico, no existe una definición unívoca en relación con el lenguaje de programación. No obstante, para este ensayo hemos decidido elaborar una definición basándonos en la información recogida por Michael L. Scott en su obra titulada Programming Language Pragmatics. Según este escrito, podríamos entender un lenguaje de programación como un medio que expresa algoritmos y procesos computacionales precisos y que está compuesto por un conjunto de símbolos y reglas que son los que definen su gramática y semántica (Scott, 2011). Estos lenguajes son utilizados por los programadores para comunicarse con la computadora en cuestión y describir como deben llevar a cabo las tareas específicas.

De lo anterior se desprende que los lenguajes de programación suponen la piedra angular de las herramientas de todo programador, estos lenguajes están sujetos y clasificados en torno distintos paradigmas y estándares dentro del mundo de la programación.

Tipos de lenguajes de programación

Los lenguajes de programación pueden ser clasificados en función de su nivel de abstracción y proximidad al lenguaje máquina. Dentro de esta clasificación tenemos los lenguajes de alto nivel, medio nivel y bajo nivel. Cuánto más bajo sea el nivel, mayor proximidad existe hacia el lenguaje máquina.

Para aclarar conceptos, sería interesante esbozar una definición de lenguaje máquina. El lenguaje máquina puede entenderse como un conjunto de instrucciones específicas dirigidas a una arquitectura de procesador concreta. Las instrucciones mencionadas anteriormente están codificadas en formato binario y consisten en operaciones básicas que el procesador puede ejecutar de forma directa (Patterson y Hennessy, 2017).

Una vez teniendo claro en qué consiste el concepto de lenguaje de programación y le concepto de lenguaje máquina, resulta vital recalcar cual es la mayor diferencia que existe entre ambos. Las diferencias entre lenguaje de programación y el lenguaje máquina recae eminentemente sobre dos vertientes: su nivel de abstracción y su relación con el hardware de la computadora en cuestión (Patterson y Hennessy, 2017).

El lenguaje máquina posee un nivel de abstracción bajo, lo cual implica que las operaciones que se realicen son ejecutadas directamente por la CPU, afectan al hardware también de forma directa y además el lenguaje máquina tiene una serie de instrucciones únicas que varían en función de la arquitectura del CPU en cuestión. El lenguaje de programación por su parte, comparado con el lenguaje máquina, es de alto nivel y por ende ofrecen una mayor abstracción, están pensados para que estos sean más comprensibles para el programador. El lenguaje de programación a su vez es más portable que el lenguaje máquina ya que estos pueden ejecutarse en distintos hardwares con ayuda de un compilador que es el que hace de intérprete para el hardware (Patterson y Hennessy, 2017).

Ahora que hemos desgranado las principales características de los lenguajes de programación con carácter general y del lenguaje máquina, procederemos a profundizar en torno a los lenguajes de alto, medio y bajo nivel.

Los lenguajes de programación de alto nivel son aquellos que se caracterizan por su alto nivel de abstracción, son los que más permiten a los programadores expresar una lógica más cercana al pensamiento humano y más alejada de la arquitectura de la máquina. Estos lenguajes son los que ofrecen una mayor legibilidad, eficiencia y productividad a la persona que los esté utilizando (Scott, 2011).

Uno de los lenguajes de programación de alto nivel más representativo es Python. Este lenguaje de programación es conocido por ofrecer una sintaxis clara y concisa muy favorable a la legibilidad del código enfocada a la resolución de problemas de forma práctica. Python se usa en campos muy diversos, solo centrarnos en desarrollar de forma extensa todos los usos de este lenguaje de programación daría para un ensayo propio, lo que sí haremos será presentar algunas de las aplicaciones que tiene el lenguaje presentado. Python es utilizado en el ámbito del desarrollo web siendo aplicado en especial en el framework Flask (Grinberg, 2018), en el ámbito de la ciencia de datos y el análisis de datos para el uso y creación de bibliotecas (como pandas) que permiten interacción con

otras bibliotecas como IPython (McKinney, 2017), en el campo de la inteligencia artificial y el machine learning para bibliotecas como TensorFlow y scikit-learn (Rachka y Mirjalili, 2017) o el ámbito del desarrollo de videojuegos con la biblioteca Pygame que permite crear de forma sencilla escenarios en 2D (Matthes, 2019).

Java supone otro lenguaje de programación de alto nivel bastante representativo, ejemplos de uso de Java tenemos también múltiples. Java es utilizado para el desarrollo de aplicaciones empresariales (Bloch, 2008), para el desarrollo de aplicaciones móviles en Android (Griffiths y Griffiths, 2017) o para el desarrollo de servicios web utilizando el Spring Framework o también sistemas embebidos y dispositivos móviles por poner otro ejemplo (Walls, 2019).

Continuando con los niveles de los lenguajes de programación, los lenguajes de programación de medio nivel son aquellos que se encuentran a medio camino entre los lenguajes de alto nivel y los lenguajes de bajo nivel, este tipo de lenguajes de programación favorecen una mezcla de control directo sobre el hardware y abstracción. Los lenguajes de programación de medio nivel permiten un mayor control sobre la gestión de la memoria que los lenguajes de alto nivel mediante la asignación de la memoria y la manipulación directa de punteros, a su vez permiten mayor control sobre sobre los recursos del sistema en comparación con los lenguajes de alto nivel, no obstante es importante destacar que ni el grado de posibilidad de manipulación sobre la memoria ni sobre los recursos llega a ser comparable con los lenguajes de bajo nivel (Scott, 2011)(Stourstroup, 2009).

En lo que a lenguajes de programación de medio nivel más representativos respecta, podríamos hablar de C y C++. En lenguaje de programación C está caracterizado por ser un lenguaje de propósito general que proporciona ese término medio entre la abstracción y el control que suponen los lenguajes de bajo nivel (Kernighan y Ritchie, 1988). El lenguaje mencionado se utiliza por ejemplo para programación de sistemas (C en su momento fue diseñado para implementar el sistema operativo conocido como Unix), para programación de redes (uso de sockets, creación de servidores, protocolos de red…) o para el desarrollo de compiladores, por mencionar algunos ejemplos.

C++ por su parte es una extensión de C, el primero introduce una serie de características adicionales que inciden directamente en la ampliación de funciones del segundo. C++ permite la programación orientada a objetos (POO) cosa que C no permitía

ya que este era un lenguaje procedural, C++ a su vez permite la definición de funciones miembro dentro de las clases y la sobrecarga de operadores (esto está orientado a facilitar la manipulación de datos personalizados), permite de igual forma otras características como lo son las plantillas, los namespaces y el manejo de excepciones (entre otras muchas mejoras respecto a C) (Kernighan y Ritchie, 1988).

Por último, antes de pasar al siguiente apartado de este escrito, describiremos las características de los lenguajes de programación de bajo nivel. El lenguaje de bajo nivel es el lenguaje de programación que se comunica directamente con el hardware y la arquitectura de una computadora. Los lenguajes de bajo nivel facilitan un control muy detallado y directo sobre los recursos del sistema, son los que menos nivel de abstracción tienen (y por tanto supone mayores dificultades de comprensión para el programador) y además también son los lenguajes menos portátiles de la clasificación (Patterson y Hennessy, 2017).

El lenguaje máquina que hemos definido al principio de este apartado supone un lenguaje de bajo nivel, para no volver a mencionarlo pondremos como ejemplo el lenguaje ensamblador. El lenguaje ensamblador eminentemente supone un nexo entre el lenguaje de máquina y los lenguajes de más alto nivel, el lenguaje ensamblador es más legible que el lenguaje máquina y por lo tanto facilitan un poco más la labor a los programadores que el lenguaje máquina cuando lo que se busca es trabajar de forma más directa con el hardware. Como características reseñables, el lenguaje ensamblador se rigen mediante nemotécnicos que se traducen en operaciones específicas dentro de la CPU (Irvin, 2018).

Paradigmas de programación

Dentro del contexto de la informática, los paradigmas de programación se entienden como modelos conceptuales que limitan la estructura y el enfoque que ha de seguirse a la hora de crear un software. Los paradigmas dentro de la programación suponen un conjunto de restricciones y principios en torno a la forma en la que se puede concebir, mantener e interpretar el software en cuestión (Van Roy y Haridi, 2004)(Sebesta, 2015).

Los principales paradigmas dentro de la programación son los siguientes: el paradigma imperativo, el paradigma declarativo, el paradigma orientado a objetos, el

paradigma funcional y el paradigma lógico. A continuación detallaremos en qué consiste cada uno así como los lenguajes representativos de cada uno de los paradigmas.

El paradigma imperativo está enfocado a la realización de una tarea mediante una secuencia de instrucciones. Este paradigma tiene como característica la ejecución de comandos que modifican variables y estados. La programación procedural (como mencionamos más arriba en relación con C) es aquella que organiza el código en torno a rutinas o procedimientos, esta forma parte del paradigma imperativo (Scott,2011) (Van Roy y Haridi, 2004).

Algunos de los lenguajes de programación más representativos dentro del paradigma imperativo serían Pascal y C. Respecto a C cabe mencionar que es representativo para el paradigma imperativo en cuanto a la forma de acceso a la memoria en tanto que permite el acceso directo, también permite la manipulación aritmética y de bits de punteros proporcionando un control preciso sobre los recursos del sistema (algo clave del paradigma imperativo), entre otras cuestiones. Pascal por su parte proporciona una serie de estructuras de control fácilmente legibles y una manipulación directa de sus variables. Ambos lenguajes de programación son representativos del paradigma imperativo en tanto que presentan una sintaxis clara y directa, estructuras de control y permiten la manipulación de variables y estados (Scott, 2011) (Roy y Haridi, 2004).

El paradigma declarativo está caracterizado por expresar la lógica de un programa pero sin entrar en detalles explícitos en torno a la manera de seguir los pasos para alcanzar el resultado deseado (no sigue el mismo proceso que el paradigma imperativo). Algunos lenguajes representativos del paradigma declarativo son Haskell y SQL. ¿Por qué estos lenguajes de programación son representativos del paradigma declarativo? Porque permiten a los programadores centrarse en la descripción de forma concisa de la lógica de sus programas sin tener que prestar atención a la implementación del mismo paso a paso y permitiendo un alto grado de abstracción (Scott, 2011) (Roy y Haridi, 2004).

El paradigma orientado a objetos por su parte consiste en un modelo que estructura el código en torno a entidades que se conocen como objetos, estos objetos lo que hacen es encapsular datos y funciones relacionadas. Los principios fundamentales de este paradigma son el de encapsulamiento, polimorfismo y el principio de herencia. Este paradigma tiene lenguajes representativos tales como Java y Python, ya hemos dado alguna pincelada un poco más arriba en este mismo escrito así que no nos detendremos

mucho describiéndolos pero sí destacaremos que sendos lenguajes son representativos para el paradigma orientado a objetos porque permiten la abstracción de entidades del mundo real en el código, porque permiten la modularidad del código y además la reutilización del mismo materializado en la herencia (Scott, 2011) (Roy y Haridi, 2004).

El paradigma funcional trata a las funciones computacionales como si se tratase de una evaluación de funciones matemáticas, evita la mutación de los datos y el cambio de estado. Este paradigma tiene como base una serie de conceptos teóricos relacionados con el cálculo lambda y la teoría de funciones. Algunos lenguajes de programación representativos serían Lisp y Erlang. Los lenguajes mencionados anteriormente ostentan transparencia referencial, funciones de primera clase, inmutabilidad y recursión como características esenciales relacionadas con el paradigma funcional (Scott, 2011) (Roy y Haridi, 2004).

Por último, el paradigma lógico consiste en un tipo paradigma de programación que se encuentra contenido dentro del paradigma declarativo, este paradigma se basa en la lógica matemática y en la teoría de conjuntos. Algunos de los lenguajes de programación más representativos dentro del paradigma lógico son el Prolog y el Mercury. Estos lenguajes de programación son representativos en tanto que son capaces de expresar relaciones lógicas, proporcionar herramientas relacionadas con la manipulación declarativa de conocimiento y soluciones en un dominio específico (Scott, 2011) (Roy y Haridi, 2004).

Estándares de programación

Los estándares son normas establecidas que buscan regularizar prácticas dentro de distintos ámbitos. En el contexto concreto que estamos tratando, los estándares de programación buscan estandarizar una serie de estructuras, prácticas y procesos dentro del campo del desarrollo del software con el objetivo mantener una coherencia y calidad en el código que se está escribiendo de forma que se facilite tanto su comprensión como su mantenimiento a largo plazo.

Existen muchos estándares para la programación que son muy reconocidos, a continuación pasaré a describir algunos que presentan cierta relevancia dentro de esta categoría; El JavaScript Standard Style define una serie de reglas de estilo para el código que esté escrito en JavaScript, el PEP 8 es similar a JavaScript Standard Style pero enfocado en Python, este consiste en una guía de estilo oficial para el código del lenguaje

de programación Python, establece una serie de convenciones para la escritura del mismo, Google Java Style Guide supone otra guía de estandarización del estilo de código para Java, esta se centra en notaciones recomendables en torno a los comentarios, nombres de variables y otros aspectos (Python Software Foundation, 2023)(Feross, 2023)(Google, 2023).

Adherirse a los estándares dentro de la programación es importante ya que genera una serie de beneficios a la vez que perjudica si nunca nos guiamos por esos estándares. Los principales beneficios radican en la legibilidad del código que garantiza que otro usuario o programador sea capaz de entender el trabajo desarrollado hasta el momento (facilitando a su misma vez la colaboración), seguir los estándares también hace que sea más sencillo la interoperabilidad, la revisión del mismo (relacionado con la legibilidad del código) y la eficiencia (la estandarización facilita que todo el mundo se rija por las mismas reglas). Las consecuencias de no seguir los estándares dentro de la programación pueden dificultar la legibilidad del mismo, dar lugar a problemas de seguridad, menor reutilización del código en cuestión, entre otros problemas derivados de la falta de seguir una serie de normas comunes que implica la estandarización.

Conclusión

Tener un profundo entendimiento sobre los lenguajes de programación, paradigmas y estándares es crucial para desarrollarse profesionalmente dentro del mundo del desarrollo de software y la programación en general. Considero que es muy importante sobre todo tener en mente cuál es nuestro objetivo a la hora de programar para de esa forma poder seleccionar de forma cuidadosa el lenguaje de programación que es adecuado para realizar la tarea que deseamos. Una vez que sabemos qué es lo que queremos hacer, es importante seguir los estándares relacionados con esa actividad concreta de forma que podamos desarrollar nuestro trabajo de forma óptima y convencional, teniendo en cuenta todo lo expuesto en los distintos puntos a lo largo de este ensayo.

También sería interesante recalcar que debemos tener metas y objetivos realistas a la hora de programar y encontrar un nicho en el que nos interese desempeñarnos e ir dominándolo dando pasos cortos, resulta altísimamente complicado si no imposible dominar en profundidad todos los lenguajes de programación a todos los niveles.

El desarrollo del software en cualquiera de los casos requiere de un enfoque integral y diverso y el conjunto de herramientas que el desarrollador desee utilizar debe adaptarse a los objetivos que éste tenga para con su tarea a realizar.

Bibliografía

Bloch, J. (2008). *Effective Java*. Addison-Wesley.

Feross. (2023). *JavaScript Standard Style*. https://standardjs.com/

Grinberg, M. (2018). *Flask Web Development*. O'Reilly Media.

Griffiths, D., & Griffiths, D. (2017). *Head First Android Development*. O'Reilly Media.

Google. (2023). *Google Java Style Guide*. https://google.github.io/styleguide/javaguide.html

Irvine, K. R. (2018). *Assembly Language for x86 Processors*. Pearson.

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language*. Prentice Hall.

Matthes, E. (2019). *Python Crash Course*. No Starch Press.

McKinney, W. (2017). *Python for Data Analysis*. O'Reilly Media.

Python Software Foundation. (2023). *PEP 8 -- Style Guide for Python Code*. https://pep8.org/

Rachka, S., & Mirjalili, V. (2017). *Python Machine Learning*. Packt Publishing.

Scott, M. L. (2011). *Programming Languages Pragmatics*. Morgan Kaufmann.

Sebesta, R. W. (2015). *Concepts of Programming Languages*. Pearson.

Stroustrup, B. (2009). *Programming: Principles and Practice Using C++*. Pearson Education.

Sweigart, A. (2015). *Automate the Boring Stuff with Python*. No Starch Press.

Van Roy, P., & Haridi, S. (2004). *Programming Paradigms*. MIT Press.

Walls, C. (2019). *Spring in Action*. Manning Publications.

Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann.