Paradigmas de programación

Programming Paradigms

Autores: Juan Camilo Salazar Quiroz y Miguel Ángel Ocampo Morales

*Ingeniería de sistemas computación, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correos: [j.salazar5@utp.edu.co](mailto:j.salazar5@utp.edu.co) a.ocampo2@utp.edu.co

***Resumen*— Un lenguaje de programación es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que le proporciona a una persona, en este caso el programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico y/o lógico de una computadora, de manera que se puedan obtener diversas clases de datos. A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina programa.**

***Palabras clave—* Paradigmas de programación, imperativo, declarativo, lógico, funcional, matemática, filosofía, cómputo, punto flotante, sistema binario**

***Abstract*—** **A programming language is a formal language (or artificial, i.e., a language with well-defined grammatical rules) that gives a person, in this case the programmer, the ability to write (or program) a series of instructions or sequences of commands in the form of algorithms in order to control the physical and/or logical of a computer, so that different kinds of data can be obtained. This whole set of commands written in a programming language is called a program.**

***Key Word* —** **Programming paradigms, imperative, declarative, logical, functional, mathematical, philosophy, computation, floating point, binary system**

1. INTRODUCCIÓN

Un paradigma de programación consiste en un método para llevar a cabo cómputos y la forma en la que deben estructurarse y organizarse las tareas que debe realizar un programa.​ Se trata de una propuesta tecnológica adoptada por una comunidad de programadores, y desarrolladores cuyo núcleo central es incuestionable en cuanto que únicamente trata de resolver uno o varios problemas claramente delimitados; la resolución de estos problemas debe suponer consecuentemente un avance significativo en al menos un parámetro que afecte a la ingeniería de software. Representa un enfoque particular o filosofía para diseñar soluciones. Los paradigmas difieren unos de otros, en los conceptos y la forma de abstraer los elementos involucrados en un problema, así como en los pasos que integran su solución del problema, en otras palabras, el cómputo. Tiene una estrecha relación con la formalización de determinados lenguajes en su momento de definición. Es un estilo de programación empleado.

Un paradigma de programación está delimitado en el tiempo en cuanto a aceptación y uso, porque nuevos paradigmas aportan nuevas o mejores soluciones que lo sustituyen parcial o totalmente.

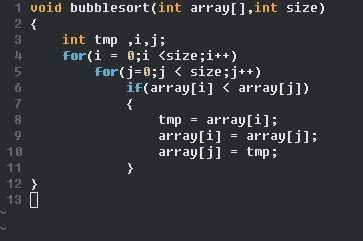
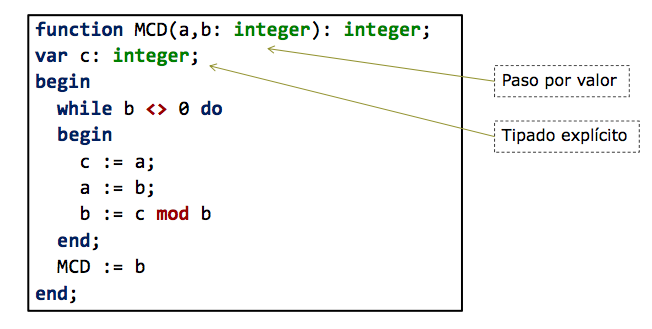
1. CONTENIDO

El procesamiento de textos no nace de la tecnología informática. Se desarrolló de las necesidades de escritores más bien que de las de matemáticos, aunque más adelante se combinara con el campo de las computadoras. La historia del procesamiento de textos es la historia de la automazación gradual de los aspectos físicos de la escritura y la edición, y el refinamiento de la tecnología para ponerla a disposición los usuarios individuales y corporativos.

La invención de la imprenta y de los tipos móviles en el final de la edad media fue el paso inicial en esta automatización. Pero el mayor avance desde la escritura manual lo fue la máquina de escribir. Henry Mall, ingeniero inglés de principios del siglo XVII, es considerado su inventor. El hecho de que hoy casi no se sabe nada sobre su invento es evidencia de su carencia del éxito.  
Christopher Latham Sholes, con la ayuda de dos colegas, inventó la primera máquina de escribir aceptada, en 1867. Comenzó a comercializarse en 1874, por una compañía de fabricación de armas, llamada Remington & Sons, aunque esta información es todavía improbable La desventaja principal de este modelo era que imprimía en la superficie inferior del rodillo, de modo que el mecanógrafo no podía ver su trabajo hasta que había acabado.

La aceptación de la máquina de escribir fue lenta al principio, pero se facilitó durante los años próximos gracias a varias mejoras. Los paradigmas de programación son los siguientes:

Imperativo

Los primeros lenguajes imperativos fueron los lenguajes de máquina de los equipos originales. En estos lenguajes, las instrucciones eran muy simples, lo que hizo más fácil la aplicación de hardware, pero dificultaron la creación de programas complejos. FORTRAN, desarrollado por John Backus en IBM a partir de 1954, fue el primer lenguaje de programación principal para eliminar los obstáculos presentados por el código de la máquina en la creación de programas complejos. FORTRAN es un lenguaje compilado que permitió que las variables con nombre, expresiones complejas, subprogramas, y muchas otras características ahora comunes en los lenguajes imperativos. Las dos décadas siguientes vieron el desarrollo de una serie de otros lenguajes imperativos de alto nivel importantes. A finales de los años 1950 y 1960, ALGOL fue desarrollado con el fin de permitir que los algoritmos matemáticos que se expresan más fácilmente, e incluso sirvió como lengua de destino del sistema operativo para algunos equipos. PAPERAS llevaron el paradigma imperativo a un extremo lógico, al no tener ninguna declaración en absoluto, basándose puramente en comandos, incluso hasta el punto de hacer que los IF y ELSE comandos independientes el uno del otro, conectados sólo por una variable intrínseca denominada $ test. COBOL y BASIC eran dos intentos de hacer sintaxis de programación se parecen más a inglés. En la década de 1970, Pascal fue desarrollado por Niklaus Wirth y C fue creado por Dennis Ritchie mientras trabajaba en los Laboratorios Bell. Wirth pasó a diseñar Modula-2 y Oberon. Para las necesidades de los Estados Unidos Departamento de Defensa, Jean Ichbiah y un equipo de Honeywell Ada comenzó a diseñar en 1978, después de un proyecto de 4 años para definir los requisitos para la lengua. La especificación fue publicada por primera vez en 1983, con revisiones en 1995 y 2005/6.

En programación imperativa la computación se realiza cambiando el estado del programa por medio de sentencias que definen pasos de ejecución. Las dos características principales del paradigma imperativo son, por tanto, la existencia de estado modificable y la ejecución de sentencias de control del programa.

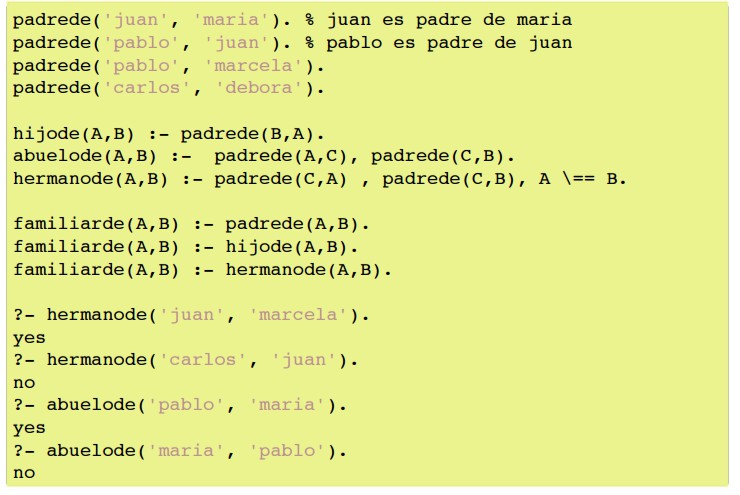
Declarativa

El origen de la palabra paradigma entendida como un marco general en el que se desarrollan teorías científicas se encuentra en el trabajo de 1962 del filósofo e historiador de la ciencia Thomas S. Kuhn, La estructura de las revoluciones científicas. Esa palabra ha sido después adoptada por el mundo de la computación para definir un conjunto de ideas y principios comunes de grandes grupos de lenguajes de programación.

1954 – El primer lenguaje de programación científica, el Fortran I, fue diseñado por John Backus en IBM en 1954. La programación científica se preocupa especialmente de la realización de cálculos complejos de forma muy rápida y precisa, definidos por modelos matemáticos que representan fenómenos científicos.

Programación declarativa, en contraposición a la programación imperativa, es un paradigma de programación que está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla (tan solo se le indica a la computadora qué es lo que se desea obtener o qué es lo que se está buscando). No existen asignaciones destructivas, y las variables son utilizadas con transparencia referencial.

Lógico

Uno de los precursores de la lógica matemática y, en consecuencia, de la programación lógica fue Aristóteles (384-322 a.C.) con su teoría silogística. Esta teoría estudia una clase particular de implicaciones con dos premisas y una conclusión. También fue tratada por los filósofos contemporáneos a Aristóteles y largamente estudiada en siglos posteriores, aunque no se produjeron innovaciones de interés hasta el siglo XVII con los trabajos de René Descartes y Gottfried Leibnitz.

Dos siglos después el matemático y lógico británico George Boole (1815-1864) dio un paso importante en el sistema de razonamiento aristotélico poniendo en relación la lógica y el álgebra. Los trabajos de Boole fueron modificados y ampliados más tarde por otros matemáticos y lógicos como Jevon, Pierce, Schroeder y Huntington, entre otros.

Llegamos así a finales del siglo XIX y principios del XX con la revolución de la fundamentación de las Matemáticas gracias a los trabajos de Frege, Cantor, Peano, Russell, Whitehead, entre otros, que marcan el periodo más apasionante y de mayor actividad en la historia de la lógica matemática.

En la mitad del siglo XX descubrimos que de forma paralela al desarrollo de la lógica se ha producido un espectacular avance

de las llamadas “máquinas de calcular”, avance sobre el que reflexiona Alan Turing en un artículo titulado “¿Pueden pensar las máquinas?”, publicado en 1950 y que podemos dar como punto de partida de lo que después se llamará Inteligencia Artificial.

La programación lógica es un tipo de paradigmas de programación dentro del paradigma de programación declarativa. El resto de los subparadigmas de programación dentro de la programación declarativa son: programación funcional, programación con restricciones, programas DSL (de dominio específico) e híbridos. La programación funcional se basa en el concepto de función (que no es más que una evolución de los predicados), de corte más matemático. La programación lógica gira en torno al concepto de predicado, o relación entre elementos.

La programación lógica encuentra su hábitat natural en aplicaciones de inteligencia artificial o relacionadas:

Sistemas expertos, donde un sistema de información imita las recomendaciones de un experto sobre algún dominio de conocimiento.

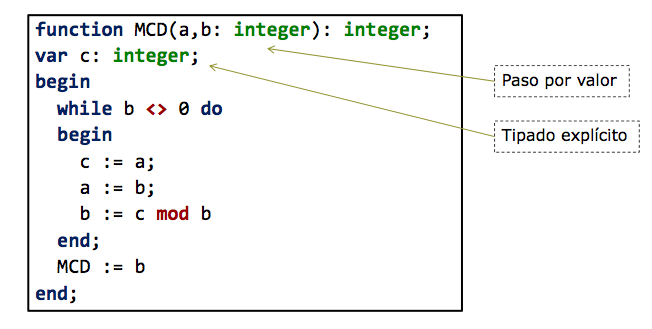
Demostración automática de teoremas, donde un programa genera nuevos teoremas sobre una teoría existente.

Reconocimiento de lenguaje natural, donde un programa es capaz de comprender (con limitaciones) la información contenida en una expresión lingüística humana.

La programación lógica también se utiliza en aplicaciones más "mundanas" pero de manera muy limitada, ya que la programación tradicional es más adecuada a tareas de propósito general.

Funcional.

El objetivo es conseguir lenguajes expresivos y matemáticamente elegantes, en los que no sea necesario bajar al nivel de la máquina para describir el proceso llevado a cabo por el programa, y evitar el concepto de estado del cómputo.



Todo empiezó con Gottfried Leibniz, quien creó la máquina mecánica de cálculo en el siglo XVII. Esta máquina fue el primer prototipo del dispositivo soñado por Leibniz: una máquina capaz de manipular símbolos y determinar si una frase matemática era o no un teorema, es decir, si una proposición que partía de un supuesto (hipótesis), afirmaba una verdad (tesis) que no es evidente por sí misma.

Para el año de 1928, los matemáticos David Hilbert y Wilhelm Ackermann propusieron el problema de la decisión, que consiste en encontrar un proceso o algoritmo (aún no se tenía la definición formal de algoritmo como tal) general, que decidiera si una fórmula de cálculo de primer orden es un teorema retomando la idea desarrollada por Leibniz.

En 1936, Alonzo Church desarrolló la definición formal de algoritmo bajo el concepto de “calculabilidad efectiva” y diseñó una solución al problema planteado por Hilbert y Ackermann utilizando un modelo de computación denominado por él mismo como Cálculo Lambda, la base fundamental de este paradigma.

En este mismo año, Alan Turing -al igual que Church- desarrollaba una definición de algoritmo y daba solución al problema de la decisión, pero usando las máquinas de Turing, otro modelo de computación que se convertiría en la base de la computación actual, bajo un concepto completamente diferente al cálculo lambda: el problema de la parada. Cabe aclarar que los dos modelos computacionales son equivalentes ya que ambos pueden dar solución a los mismos tipos de problemas.

El paradigma funcional se empezó a desarrollar por el matemático John McCarthy en 1956, para programar los primeros proyectos de inteligencia artificial sobre un computador IBM 704 durante su desarrollo este crea el lenguaje de programcion lisp en 1958.

Lisp fue creado originalmente como una notación matemática práctica para los programas de computadora, basada en el cálculo lambda de Alonzo Church. apesar de no ser un lenguaje puramente funcional Se convirtió rápidamente un lenguaje de programación pionero en la investigación de la inteligencia artificial (AI) estableciendo las bases de lo que se conoceria hoy como el paradigma funcional.

Los programas escritos en un lenguaje funcional están constituidos únicamente por definiciones de funciones, entendiendo estas no como subprogramas clásicos de un lenguaje imperativo, sino como funciones puramente matemáticas, en las que se verifican ciertas propiedades como la transparencia referencial (el significado de una expresión depende únicamente del significado de sus subexpresiones), y por tanto, la carencia total de efectos colaterales.

Otras características propias de estos lenguajes son la no existencia de asignaciones de variables y la falta de construcciones estructuradas como la secuencia o la iteración (lo que obliga en la práctica a que todas las repeticiones de instrucciones se lleven a cabo por medio de funciones recursivas).

Existen dos grandes categorías de lenguajes funcionales: los funcionales puros y los híbridos. La diferencia entre ambos estriba en que los lenguajes funcionales híbridos son menos dogmáticos que los puros, al admitir conceptos tomados de los lenguajes imperativos, como las secuencias de instrucciones o la asignación de variables. En contraste, los lenguajes funcionales puros tienen una mayor potencia expresiva, conservando a la vez su transparencia referencial, algo que no se cumple siempre con un lenguaje funcional híbrido.

1. CONCLUSIONES

Los lenguajes de programación están formados por un conjunto de símbolos (llamado alfabeto), reglas gramaticales (léxico/morfológicas y sintácticas) y semánticas, que en conjunto definen las estructuras válidas del lenguaje y su significado. Existe el error común de trata como sinónimos los términos 'lenguaje de programación' y 'lenguaje informático'. Los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como por ejemplo HTML (lenguaje para el marcado de páginas web que no es propiamente un lenguaje de programación, sino un conjunto de instrucciones que permiten estructurar el contenido de los documentos).

El lenguaje de programación permite especificar de manera precisa sobre qué datos debe operar un software específico, cómo deben ser almacenados o transmitidos dichos datos, y qué acciones debe tomar el software bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural. Una característica relevante de los lenguajes de programación es precisamente que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos para realizar la construcción de un programa de forma colaborativa.

REFERENCIAS

1. <https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n>
2. <https://kevinldp.wordpress.com/paradigma-imperativo/>
3. <http://www.dccia.ua.es/dccia/inf/asignaturas/LPP/2010-2011/teoria/tema7.html>
4. <https://lh3.googleusercontent.com/-z2Qu4zWTzlY/Xd2F9LMUlII/AAAAAAAAAA0/DpcEd6FK32QPj9wWHxjpBdAUVXACwQkfgCK8BGAsYHg/s0/2019-11-26.jpg>
5. <https://willitc9888.wordpress.com/tag/paradigmas-declarativos/>
6. <https://www.ecured.cu/Programaci%C3%B3n_declarativa>
7. <https://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/introduccion.html>
8. <https://tuparadigma.files.wordpress.com/2015/08/seleccic3b3n_003.jpg>
9. <https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional#Utilidad>
10. <https://objetosweb.wordpress.com/2016/02/01/paradigmas-de-programacion/>
11. <http://ferestrepoca.github.io/paradigmas-de-programacion/progfun/funcional_teoria/index.html>