PARADIGMAS DE PROGRAMACIÓN

**PROGRAMMING PARADIGMS**

Autor 1: Juan Andrés Arias Tascón

Autor 2: John Stiven Acevedo Zapata

*Ingeniería de Sistemas, Universidad Tecnológica de Pereira*

Correo-e: [juanandres.arias@utp.edu.co](mailto:juanandres.arias@utp.edu.co)

[s.acevedo1@utp.edu.co](mailto:s.acevedo1@utp.edu.co)

***Resumen*— Un paradigma de programación es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. Los lenguajes de programación, necesariamente, se encuadran en uno o varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, algo que tiene una relación directa con su sintaxis.**

* **Programación Imperativa**
* **Programación funcional**
* **Programación lógica**
* **Declarativo**
* **Orientado a Objetos**

**Habitualmente se mezclan todos los tipos de paradigmas a la hora de hacer la programación. De esa manera se origina la programación multiparadigma, pero el que actualmente es más usado de todos esos paradigmas es el de la programación orientada a objetos.**

**Los paradigmas fundamentales están basados en diferentes**

**modelos de cómputo y por lo tanto afectan a las construcciones más básicas de un programa.**

**La división principal reside en el enfoque imperativo (indicar el**

**cómo se debe calcular) y el enfoque declarativo**

**El enfoque declarativo tiene varias ramas diferenciadas: el**

**paradigma funcional, el paradigma lógico, la programación**

**reactiva y los lenguajes descriptivos.**

**Otros paradigmas se centran en la estructura y organización de**

**los programas, y son compatibles con los fundamentales.**

***Palabras clave— paradigma, imperativo, funcional, lógica, declarativo, objetos, programación, computo.***

***Abstract*— A programming paradigm is a style of program development. That is, a model to solve computational problems. The programming languages, necessarily, are framed in one or several paradigms at the same time based on the type of orders they allow to implement, something that has a direct relationship with their syntax.**

**- Imperative Programming**

**- Functional programming**

**- Logic programming**

**- Declarative**

**- Object Oriented**

**Usually all types of paradigms are mixed when programming. That way multiparadigm programming originates, but the one that is currently most used of all these paradigms is that of object-oriented programming.**

**The fundamental paradigms are based on different**

**Computational models and therefore affect the most basic constructions of a program.**

**The main division lies in the imperative approach (indicate the**

**how to calculate) and the declarative approach**

**The declarative approach has several distinct branches:**

**functional paradigm, logical paradigm, programming**

**reactive and descriptive languages.**

**Other paradigms focus on the structure and organization of**

**the programs, and are compatible with the fundamentals.**

***Key Word* —** ***paradigm, imperative, functional, logical, declarative, objects, programming, computing.***

1. INTRODUCCIÓN

A lo largo de la historia, el término «paradigma» fue objeto de muchas interpretaciones. En su origen griego, significaba «modelo», «ejemplo» o «patrón». Sobre este punto de partida, podemos hablar de un paradigma como un conjunto de creencias, prácticas y conocimientos que guían el desarrollo de una disciplina durante un período de tiempo. En diversas ramas de la ciencia, un conjunto de ideas en vigencia puede ser reemplazado drásticamente por otro que entre en conflicto con él y se demuestre más acertado. La programación tiene sus propios paradigmas, pero el término «paradigma de programación» no necesariamente representa un modelo único que deba ser respetado hasta que aparezca otro mejor.

Los primeros lenguajes de programación fueron creados al mismo tiempo que aparecían los primeros computadores, hacia 1940. Se trataba de lenguajes denominados “de bajo nivel” cuya escritura era muy compleja. Posteriormente surgieron lenguajes que eran más fáciles de escribir y de interpretar. Se considera como primer lenguaje de alto nivel a Fortran, un lenguaje que surgió hacia finales de los años 50, diseñado para programación científica y cálculo numérico. Al contrario que otros lenguajes que cayeron en desuso, Fortran ha evolucionado y continuó siendo usado

**¿Que es un paradigma?**

La palabra paradigma en principio no tiene relación con la informática, viene de algo mucho mas complejo, el origen de la palabra es griego y hoy en día se utiliza en frases muy bonitas para vender como:

Rompamos paradigmas

Resolvamos paradigmas

El problema de estas frases en realidad es que no tienen sentido si no tienen un contexto, la palabra paradigma se podría aplicar a cualquier área y en cualquier contexto, ese es el verdadero problema.

Antes de definir la palabra paradigma te comparto lo siguiente: Para que exista un paradigma tiene que existir un problema

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.3​ 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.

Para que la computadora entienda nuestras instrucciones debe usarse un lenguaje específico conocido como código máquina, que la máquina lee fácilmente, pero que es excesivamente complicado para las personas. De hecho, solo consiste en cadenas extensas de números 0 y 1.

Para facilitar el trabajo, los primeros operadores de computadoras decidieron crear un traductor para reemplazar los 0 y 1 por palabras o abstracción de palabras y letras provenientes del inglés; este se conoce como lenguaje ensamblador. Por ejemplo, para sumar se usa la letra A de la palabra inglesa add (sumar). El lenguaje ensamblador sigue la misma estructura del lenguaje máquina, pero las letras y palabras son más fáciles de recordar y entender que los números.

La necesidad de recordar secuencias de programación para las acciones usuales llevó a denominarlas con nombres fáciles de memorizar y asociar: ADD (sumar), SUB (restar), MUL (multiplicar), CALL (ejecutar subrutina), etc. A esta secuencia de posiciones se le denominó "instrucciones", y a este conjunto de instrucciones se le llamó lenguaje ensamblador. Posteriormente aparecieron diferentes lenguajes de programación, los cuales reciben su denominación porque tienen una estructura sintáctica semejante a la de los lenguajes escritos por los humanos, denominados también lenguajes de alto nivel.

**Origen y Evolución**

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.

1. CONTENIDO

**Paradigma Imperativo**

**Que es?**

El paradigma imperativo debe su nombre al papel dominante que desempeñan las sentencias imperativas, es decir aquellas que indican realizar una determinada operación que modifica los datos guardados en memoria. Su esencia es resolver un problema complejo mediante la ejecución repetitiva y paso a paso de operaciones y cálculos sencillos con la asignación de los valores calculados a posiciones de memoria.

La programación en este paradigma consiste en determinar qué datos son requeridos para el cálculo, asociar a esos datos una dirección de memoria, y efectuar, paso a paso, una secuencia de transformaciones en los datos almacenados, de forma que el estado final represente el resultado correcto.

**Características**

Describe cómo debe realizarse el cálculo, no el porqué.

Un cómputo consiste en una serie de sentencias, ejecutadas según un control de flujo explícito, que modifican el estado del programa.

Las variables son celdas de memoria que contienen datos (o referencias), pueden ser modificadas, y representan el estado del programa.

La sentencia principal es la asignación.

Basado en el modelo de cómputo de máquinas de Turing y sobre todo en las máquinas RAM (registro + acceso aleatorio a memoria)

La gran mayoría de procesadores siguen un versión de ese modelo de cómputo + arquitectura Von Neumman.

**Lenguajes Representativos**

Basic, C, D, Fortra, Pascal, Perl, PHP, Lua

**Programación declarativo**

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.

**Características**

Se ha dicho que los lenguajes declarativos tienen la ventaja de ser razonados matemáticamente, lo que permite el uso de mecanismos matemáticos para optimizar el rendimiento de los programas.[1]

Son fiables y expresivos.

Lenguajes Representativos

Haskell (Programación funcional)

ML (Programación funcional)

Lisp (Programación funcional)

Prolog (Programación lógica)

F-Prolog (Programación lógica difusa)

Curry (Programación lógico-funcional)

SQL

QML

**Paradigma Funcional**

El paradigma funcional está basado en el modelo matemático de composición funcional. En este modelo, el resultado de un cálculo es la entrada del siguiente, y así sucesivamente hasta que una composición produce el resultado deseado. Así, un programa es un conjunto de funciones que cooperan entre ellas para el logro de un objetivo común.

Las funciones, basadas en el concepto matemático del término, son expresiones que establecen una relación de correspondencia o asociación entre miembros de un conjunto (el dominio) y miembros de otro conjunto (la imagen), en un determinado sentido. De esta manera, para cada elemento del conjunto dominio corresponde un determinado elemento del conjunto imagen, y no existen elementos del conjunto dominio que no tengan su correspondiente imagen. Generalizando, las funciones tienen su dominio conformado por varios conjuntos y existe un elemento en la imagen para cada elemento del producto cartesiano de todos ellos.

**Características**

No existe operación de asignación.

Las “variables” almacenan definiciones o referencias a expresiones.

Basado en el concepto (matemático) de función.

La operación fundamental es la aplicación de una función a una serie de argumentos. La evaluación se guía por el concepto de sustitución.

Un programa consiste en una serie de definiciones (de funciones, de tipos de datos.)

Las estructuras de control básicas (y generalmente únicas) son la composición y la recursión.

Lenguajes Representativos

Lisp

Haskell

Lógico

**Paradigma lógico**

El paradigma lógico tiene como característica principal la aplicación de las reglas de la lógica para inferir conclusiones a partir de datos. Conociendo la información y las condiciones del problema, la ejecución de un programa consiste en la búsqueda de un objetivo dentro de las declaraciones realizadas.Esta forma de tratamiento de la información permite pensar la existencia de “ programas inteligentes” que puedan responder, no por tener en la base de datos todos los conocimientos, sino por poder inferirlos a través de la deducción.

**Características**

Los programas para los lenguajes de programación lógicos son un conjunto de hechos y reglas.

La sintaxis de los lenguajes de programación lógicos es notablemente diferente de los lenguajes de programación imperativos.

Unificación de términos.

Mecanismos de inferencia automática.

Recursión como estructura de control básica.

Visión lógica de la computación.

La aplicación de las reglas de la lógica para inferir conclusiones a partir de datos.

El programa se transforma en un conjunto de declaraciones formales de especificaciones que deben ser correctas por definición.

No tiene un algoritmo que indique los pasos que detallen la manera de llegar a un resultado.

Las salidas son funcionalmente dependientes de las entradas.

Lenguajes Representativos

ALF logtalk.org\_400x400, CLP, Elf, Flang, Godel, KLIC, Logtalk, Prolog,

**Orientado a objetos**

El paradigma de objetos, o como se lo conoce generalmente, la Programación Orientada a Objetos, se fundamente en concebir a un sistema como un conjunto de entidades que representan al mundo real, los “objetos”, que tienen distribuida la funcionalidad e información necesaria y que cooperan entre sí para el logro de un objetivo común.

La Programación Orientada a Objetos es una “filosofía” de desarrollo de software que permite crear unidades funcionales extensibles y genéricas, de forma que el usuario las pueda aplicar según sus necesidades y de acuerdo con las especificaciones del sistema a desarrollar.

**Características (Teso, 2010)**

Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos.

Encapsulamiento: Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción.

Modularidad: la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

Principio de ocultación: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase.

Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

Herencia: Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación.

Recolección de basura: es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos.

Lenguajes Representativos

ABAP

ActionScript

Ada

C++

C#

Clarion

Clipper

D

Object Pascal

Gambas

Genie

Eiffel

Java

JavaScript

Lexico

Objective C

Ocaml

R

Perl

PHP

Python

Ruby

Smalltalk

Vala

NET

Visual FoxPro

Visual Basic 6.0

Visual Objects

XBase++

1. CONCLUSIONES

En el mundo de la Programación existen muchos Paradigmas que desconocemos, muchos de estos paradigmas pueden estar orientados a Base de Datos, Estructuras de un Proyecto, Servidores, etc. Es decir existen Paradigmas en varios ámbitos del Desarrollo de Software Web, Desktop, Móvil u otros, en la Programación existen Paradigmas.

La intención de este artículo es exponer los Paradigmas que se usan actualmente en la industria, ya que existen otros tipos Paradigmas en la Programación.

La evolución de los lenguajes de programación continúa, tanto en la industria como en investigación. Algunas de las tendencias actuales incluyen:

Aumentar el soporte para la programación funcional en lenguajes importantes utilizados comercialmente, incluida la programación funcional pura para hacer el código más fácil de razonar y de paralelizar (tanto en macro como en micro-niveles).

Construir lenguajes para apoyar la programación concurrente y distribuida.

Mecanismos para añadir al lenguaje verificación en cuanto a seguridad y confiabilidad: chequeo sintáctico extendido, control de flujo de información, seguridad de hilos.

Mecanismos alternativos de modularidad: mixins, delegados, aspectos.

Desarrollo de software orientado a componentes.

Mayor énfasis en cuanto a distribución y movilidad.

Integración con bases de datos, incluyendo XML y bases de datos relacionales.

Programación Orientada a Aspectos (AOP).

Lenguajes que soporten programar sobre los procesadores de la GPU en paralelo, como OpenCL.

CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

Los lenguajes de programación son clasificados de muchas formas, dentro de estas se encuentran:

**CLASIFICACIÓN SEGÚN NIVEL DE ABSTRACCIÓN**

Los lenguajes de bajo nivel:

Son lenguajes de programación que se acercan al funcionamiento de una computadora. El lenguaje de más bajo nivel es, por excelencia, el código máquina. A éste le sigue el lenguaje ensamblador, ya que al programar en ensamblador se trabajan con los registros de memoria de la computadora de forma directa.

Los lenguajes de medio nivel:

Hay lenguajes de programación que son considerados por algunos expertos como lenguajes de medio nivel (como es el caso del lenguaje C) al tener ciertas características que los acercan a los lenguajes de bajo nivel pero teniendo, al mismo tiempo, ciertas cualidades que lo hacen un lenguaje más cercano al humano y, por tanto, de alto nivel.

Los lenguajes de alto nivel:

Los lenguajes de alto nivel son normalmente fáciles de aprender porque están formados por elementos de lenguajes naturales, como el inglés.

En BASIC, el lenguaje de alto nivel más conocido, los comandos como "IF CONTADOR = 10 THEN STOP" pueden utilizarse para pedir a la computadora que pare si CONTADOR es igual a 10. Por desgracia para muchas personas esta forma de trabajar es un poco frustrante, dado que a pesar de que las computadoras parecen comprender un lenguaje natural, lo hacen en realidad de una forma rígida y sistemática.

Leer más: https://fundamentos-de-programacion97.webnode.cl/news/otendencias-actuales/

1. REFERENCIAS

* <https://kevinldp.wordpress.com/paradigma-imperativo/>
* <https://lenguajesdeprogramacion14.wordpress.com/2-paradigmas-de-la-programacion/paradigma-imperativo/>
* <https://www.ecured.cu/Programaci%C3%B3n_declarativa>
* <https://kevinldp.wordpress.com/3-paradigma-funcional/>
* <https://lenguajesdeprogramacion14.wordpress.com/2-paradigmas-de-la-programacion/paradigma-funcional/>
* <https://kevinldp.wordpress.com/4-paradigma-logico/>
* <https://lenguajesdeprogramacion14.wordpress.com/2-paradigmas-de-la-programacion/paradigma-logico/>
* <https://kevinldp.wordpress.com/5-paradigma-orientado-a-objetos/>
* <https://lenguajesdeprogramacion14.wordpress.com/2-paradigmas-de-la-programacion/paradigma-orientado-a-objetos/>