Paradigma de Programación

Autor 1: Juliana Restrepo Benavidez Autor 2: Sara Valentina Restrepo Ramírez

*Risaralda, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correos-e: j.restrepo4@utp.edu.co sara.restrepo1@utp.edu.co

***Resumen*—** es un modelo,teoría o conjunto de teorías que sirven para resolver problemas en cualquier campo de la vida,

Existen paradigmas de programación, lingüística, científicos, educativos y de la complejidad.

Esten documento trataremos los tipos de paradigmas de programación los cuales son:

Programación Imperativa.

Programación funcional.

Programación lógica.

Declarativo.

Orientado a Objetos.

Por procedimientos.

***Palabras clave—*** programación, paradigmas, lógica, modelo, función, objetos, procedimientos, conjunto, teoría

***Abstract*—**

it is a model, theory or set of theories that serve to solve problems in any field of life,

There are programming, linguistic, scientific, and educational and complexity paradigms.

This document will discuss the types of programming paradigms which are:

Imperative Programming

Functional Programming

Logic Programming

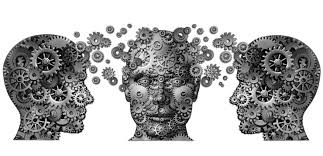
Declarative.

Object Oriented.

1. INTRODUCCIÓN

A continuación, se presentará un poco de información acerca de los tipos de paradigmas de programación y sus definiciones, ejemplos que te ayudaran a entender un poco más fácil estos paradigmas, los cuales sirven para resolver un problema, por medio de esquemas planteados a continuación.

1. CONTENIDO
2. Que es un paradigma
3. Que es un paradigma de Programación
4. ¿Cuál es la relación que existe entre las matemáticas y los paradigmas de programación?
5. Tipos de paradigmas y ejemplos.
6. Referencias



1. **¿Qué es un paradigma?**

Teoría o conjunto de teorías que sirven de modelo a seguir para resolver problemas o situaciones determinadas que se planteen en cualquier campo.

1. **¿qué es un paradigma de programación?**

Es un modelo o esquema fundamental que organiza nuestras opiniones con respecto a determinado problema, se convierte en un conjunto de reglas que rige la programación.

1. **¿Cuál es la relación que existe entre las matemáticas y los paradigmas de programación?**

Las matemáticas son quienes han proporcionado la base para que la programación de computadores sea una realidad, los paradigmas de programación son el reflejo de un modelo matemático o también de un conjunto especifico de conceptos matemáticos.

1. **tipos de paradigmas.**

****

**Paradigma imperativo**

Se rige por dos conceptos básicos para la construcción de programas: la estructura y el módulo. De ahí que se hable de programación estructurada y de programación modular.

Consiste en determinar qué datos son requeridos para el cálculo, asociar a estas direcciones de memoria y efectuar, paso a paso, una secuencia de transformaciones en los datos almacenados de forma tal que el estado final represente el resultado correcto.

La programación imperativa se basa en tres conceptos importantes:

1. Celda de memoria Variable: Donde se almacenan valores que sor referenciados y modificados mediante la ejecución del programa, a las que se suele asignar un nombre o identificador.
2. Operaciones de asignación: son las que permiten asociar los datos o valores a la posición de la memoria o variable que los contiene.
3. Operaciones de Repetición: Normalmente un programa iterativo realiza su tarea ejecutando una y otra vez una secuencia de pasos elementales.

**Ventajas:**

* Su relativa simplicidad y facilidad de implementación de los compiladores e interpretas.
* Capacidad de reutilizar el mismo código en diferentes lugares en el programa sin copiarlo.
* Una forma fácil seguir la pista de flujo del programa
* Capacidad de ser muy modular o estructura
* Necesita menos memoria.

**Desventajas:**

* Los datos son expuestos a la totalidad del programa, así que no hay seguridad para los datos
* Dificultad para relacionarse con los objetos del mundo real.
* Difícil crear nuevos tipos de datos reduce la extensibilidad.
* Se da importancia a la operación de datos en lugar de los datos mismos.

**Campos de aplicación:**

Los lenguajes imperativos pueden resolver prácticamente cualquier problema en cualquier área:

Desde simples hasta complejos cálculos matemáticos.

Se pueden hacer tipos de aplicaciones:

De nóminas, de control aéreo, de inteligencia artificial, de control de dosis de medicamentos, para cajeros automáticos, para naves espaciales, para dispositivos móviles, aplicaciones en línea y tiempo real.

Su filosofía marco la línea a seguir para resolver problemas de la vida cotidiana.

Ejemplos:

1. Función de JavaScript (solución imperativa para calcular el N-esimo término de la sucesión de Fibonacci
2. función fibonacci(n) {
3. var actual, ant1, ant2;
4. if (n === 0) {
5. actual = 0;
6. } else if (n === 1) {
7. actual = 1;
8. } else {
9. ant1 = ant2 = 1;
10. for (i = 2; i < n; i++) {
11. actual = ant1 + ant2;
12. ant2 = ant1;
13. ant1 = actual;
14. }
15. }
16. return actual;
17. }
18. Procedimiento para calcular la factorial de un numero natural en modulados.

PROCEDURE factorial(n:CARDINAL;VAR F:CARDINAL);

BEGIN

F: = 1;

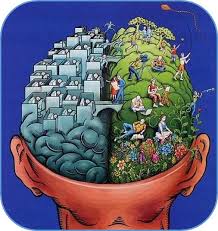
WHILE n>0 DO

F: = n\*F;

n:= n-1

END

END factorial;

****

**Paradigma funcional**

La programación funcional es un paradigma basado en la programación declarativa, a diferencia de los paradigmas basados en programación imperativa, el cual se basa en indicar paso a paso como resolver el problema, la programación declarativa en su lugar le cuentas al programa como resolver el problema, sin hacer enfoque en como realizamos tal tarea, esta tarea es delegada al compilador o interprete el indicar como resolver el problema.

La programación funcional además de poseer el estilo declarativo a su vez promueve la transparencia referencial, las funciones puras y en los cambios de estados mediante la mutación de variables. tiene sus raíces en el cálculo lambda, un sistema formal desarrollado en los años 1930 para investigar la definición de función, la aplicación de las funciones y la recursión.

Actualmente contamos con una gran cantidad de lenguajes en donde podemos hacer uso de la programación funcional, por ejemplo:

Java.

PHP.

Ruby.

Python.

Elixir.

Kotling.

Haskell.

Erlang.

**Ventajas**

* Permite definiciones simples.
* Facilita el estudio de aspectos computacionales.
* Su carácter formal facilita la demostración de propiedades.
* Más fácil de escribir.
* No hay asignación ni cambio de estado.
* No hay referencias: identificadores asociados a valores.
* Recursividad: las funciones recursivas se invocan a sí mismas, permitiendo que una operación se realice una y otra vez hasta alcanzar su caso base.

**Desventajas**

* Se quedan cortos en portabilidad, riqueza de librerías, interfaces con otros lenguajes y herramientas de depuración.
* La práctica, existen algunos inconvenientes a tener en cuenta.
* Falta de recursos: al estar tan poco extendido, faltan la gran cantidad de recursos (librerías, fromeworks, etc.)
* Mayor dificultad inicial: aunque sea muy fáciles de entender y mantener, suele ser más difícil escribir un programa funcionalmente, sobre todo para mentes acostumbradas a la imperativa.

Ejemplos:

1. Descontarle un 20% a algunos números decimales de una lista; los cuales deben ser mayores que 25 para sumarlos.

final BigDecimal sum =

numbers.stream()

filter(number ->

number.compareTo(BigDecimal.valueOf(25)) > 0)

map(number ->

number.multiply(BigDecimal.valueOf(0.8)))

.reduce(BigDecimal.ZERO,

BigDecimal::add);

System.out.println("Total: " + sum);

1. Identificar si un número es par:

(define (es-par n)

(define (par?)

(if (= (modulo n 2) 0) #t #f) )

(if (par?)

(print “es par”)

(print “es impar”) ) )



**Paradigma lógico**

Paradigma de programación basado en la lógica de primer orden. La programación lógica estudia el uso de la lógica para el planteamiento de problemas y el control sobre las reglas de inferencia para alcanzar la solución automática.

La programación lógica, junto con la funcional, forma parte de lo que se conoce como Programación Declarativa, es decir la programación consiste en indicar como resolver un problema mediante sentencias, en la Programación Lógica, se trabaja en una forma descriptiva, estableciendo relaciones entre entidades, indicando no como, sino que hacer, entonces se dice que la idea esencial de la programación lógica es

La programación lógica construye base de conocimientos mediante reglas y hechos.

**Características del Paradigma**

1.Unificación de términos.

2.Mecanismos de inferencia automática.

3.Recursión como estructura de control básica.

4.Visión lógica de la computación.

**Ventajas**

* Permite visualizar gráficamente el camino que sigue la solución a un problema.
* Por ser tan simplificado es muy entendible.
* No se necesitan muchos conocimientos técnicos para utilizar esta técnica.

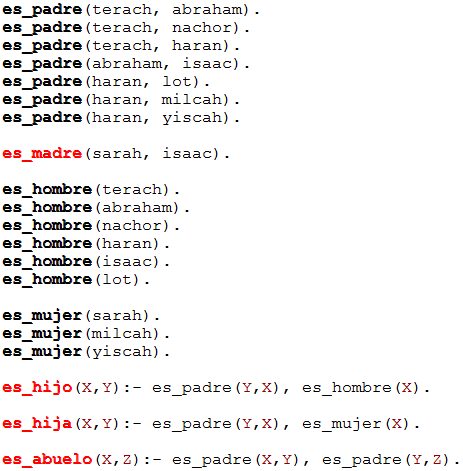
**Desventajas**

* Dado que los flujos (representados con flechas) pueden ir de cualquier lugar a cualquier lugar da espacio para que el diagrama llegue a ser casi inentendible
* Deben conocerse bien los símbolos que se van a utilizar
* No todos los símbolos están estandarizados
* Los ciclos deben ser reinterpretados para poder ser diagramados en esta técnica
* No siempre es muy entendible
* Algunas veces la analogía entre el diagrama y la codificación en el Lenguaje de Programación resulta ser compleja.



Ejemplos:

1.



**Paradigma Declarativo**

El paradigma declarativo o paradigma de programación lógica se basa en el hecho que un programa implementa una relación antes que una correspondencia. Debido a que las relaciones son más generales que las correspondencias (identificador - dirección de memoria), la programación lógica es potencialmente de más alto nivel que la programación funcional o la imperativa. El lenguaje más popular enmarcado dentro de este paradigma es el lenguaje PROLOG. El auge del paradigma declarativo se debe a que el área de la lógica formal de las matemáticas ofrece un sencillo algoritmo de resolución de problemas adecuado para, usarse en un sistema de programación declarativo de propósito genera

* Definición de reglas
* Unificación como elemento de computación
* Programación declarativa

**Características**

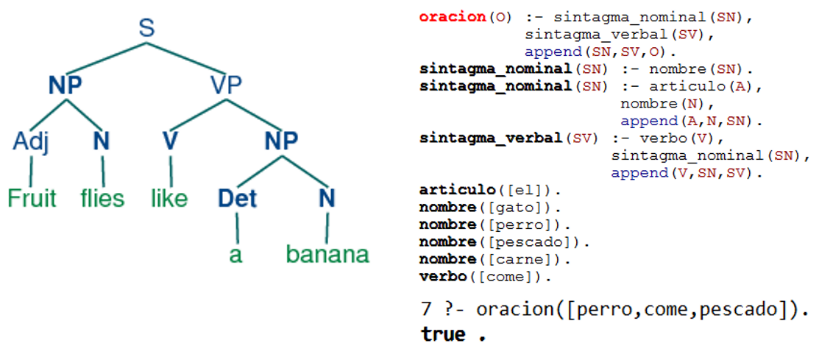
* Es un paradigma que se centra en la descripción del problema, más que los procedimientos necesitamos para legar a la solución del problema.
* Lo más importante, es descubrir un logaritmo general para la resolución del problema. Después de hacer esto solo tenemos que expresarlo en una forma compatible con dicho algoritmo y aplicarlo.
* Acá la tarea del programador es crear un enunciado preciso del problema, más que descubrir un algoritmo para resolverlo.

**Ventajas**

Entre las ventajas del paradigma declarativo se destaca que la solución de un problema se puede realizar con un nivel de abstracción considerablemente alto, sin entrar mucho en detalles de implementación, lo que hace a la solución más fácil de entender por las personas.

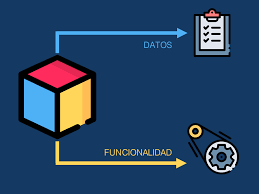
El paradigma declarativo encuentra numerosas aplicaciones industriales en campos como las bases de datos, ingeniería del software, procesadores de lenguajes, lenguaje natural, investigación operativa, seguridad de redes, etc.

**Desventajas**



El principal obstáculo del paradigma declarativo es el descubrimiento del algoritmo básico para resolver problemas. Por esta razón, los lenguajes declarativos tienden a ser de propósito específico, diseñados para usarse en aplicaciones particulares.

Otra desventaja de la programación declarativa está relacionada con la eficiencia.



Ejemplos:

1. Definición de la función factorial

factorial :: Integer -> Integer

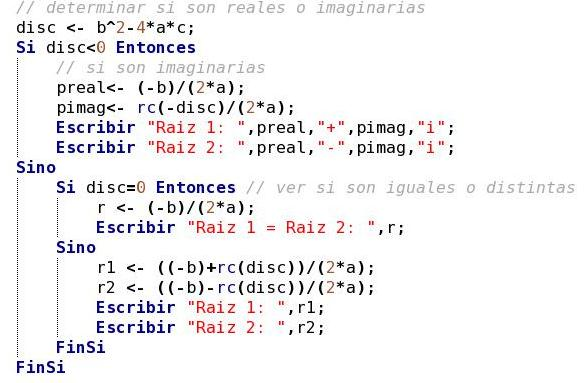
factorial n = product [1..n]

en términos recursivos:

factorial :: Integer -> Integer

factorial 0 = 1

factorial n = n \* factorial(n - 1)

1. 

**Paradigma Orientado a Objetos (POO)**

El paradigma orientado a objetos (OO) define los programas en términos de comunidades de objetos. Los objetos con características comunes se agrupan en clases (un concepto similar al de tipo abstracto de dato (TAD)). Los objetos son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos). Estos objetos se comunican entre ellos para realizar tareas. Es en este modo de ver un programa donde este paradigma difiere del paradigma imperativo o estructurado, en los que los datos y los métodos están separados y sin relación. El paradigma OO surge para solventar los problemas que planteaban otros paradigmas, como el imperativo, con el objeto de elaborar programas y módulos más fáciles de escribir, mantener y reutilizar. Entre los lenguajes que soportan el paradigma OO están Smalltalk, C++, Delphi (Object Pascal), Java y C#.

**Ventajas**

* Todo el código se encuentra en un solo lugar
* Los objetos pueden tener varios atributos, por ejemplo, que lea un sensor y a la vez encienda.
* Son más fáciles de entender los códigos.
* Le ejecución del programa es rápida y sencilla, todo se encuentra en una sola ventana.

**Desventajas**

* Los programas no pueden ser moldeados enteramente por la programación orientada a objetos.
* Para leer, modificar, o hacerles algo simplemente; en algunos programas debes realizar un paso extra para realizar estas acciones.
* Si se fuerza el lenguaje puede perder algunos objetos y características.
* Los objetos requieren una extensa documentación.
* Los objetos al ser abstracto pueden no coincidir la visión de un programador a otro

**Características**

* Abstracción: Cada objeto sirve como modelo abstracto que puede realizar trabajos. informar, cambiar su estado y comunicarse con otros objetos en el sistema sin revelar como se implementan estas características
* Encapsulación: También llamado ocultación de información asegura que no se pueda cambiar el estado de otros objetos, cada objeto tiene una interfaz diferente de los otros
* Polimorfismo: Pueden tener objetos de
* diferentes tipos y la invocación de un comportamiento correcto para el tipo real de referente esto ocurre durante el tiempo de ejecución.
* Herencia: Organiza y facilita el polimorfismo y el encapsulamiento permitiendo que los objetos sean definidos y creados especialmente.

**Algunos ejemplos de lenguajes orientada a objetos son:**

PHP

PYTHON

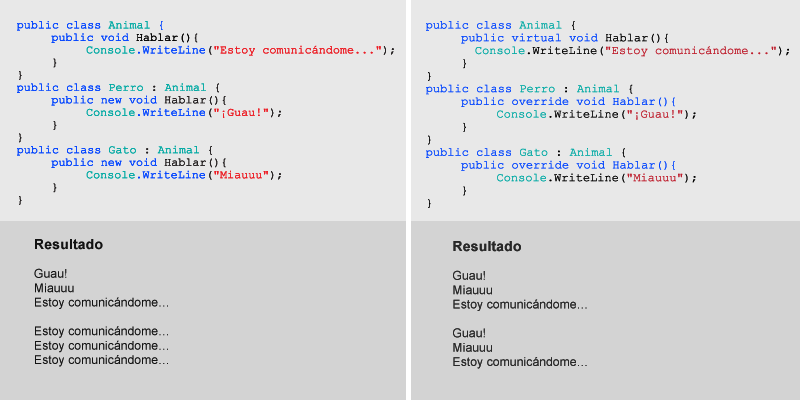
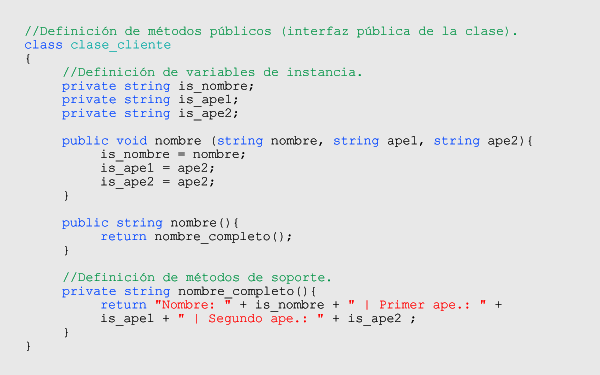
JAVA

C

C++

****

Ejemplos:

1. 
2. 

**Paradigma Por Procedimiento**

La programación procedimental o programación por procedimientos es un paradigma de la programación. Muchas veces es aplicable tanto en lenguajes de programación de bajo nivel como en lenguajes de alto nivel. En el caso de que esta técnica se aplique en lenguajes de alto nivel, recibirá el nombre de programación funcional. Esta técnica consiste en basarse de un número muy bajo de expresiones repetidas, englobarlas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse.

Esta técnica de programación ofrece muy buena respuesta con relación al tamaño de los programas, y en bruto casi no se nota en la velocidad de ejecución de los mismos (mientras que las variables, constantes o índices de vector estén en memoria, como suele ser normal, estos se relacionarán entre sí, sin una carga de memoria considerablemente alta para los procesadores modernos); aunque es muy complicado conseguir una programación por procedimientos pura.

**Ventajas**

* Mayor control en cuanto a la programación del desarrollo.
* Al tener control, se reduce el riesgo de excesos de gastos.

Ejemplos:

1. 

**Desventajas**

* El usuario no participa en el proceso de desarrollo.
* El proceso no se hace de forma secuencial.
* El tiempo de desarrollo excede al estimado.
* El usuario olvida aclarar pautas, esto puede significar, sobrecostos del proyecto.

CONCLUSIONES

* Un paradigma de programación es un estilo o lenguaje; en el cual, se desarrollará un programa.
* Existen muchos lenguajes de programación que son multiparadigma; es decir, que pueden ejecutar varios tipos de paradigmas en él o desarrollar diferentes estilos.
* El paradigma más clásico es el imperativo; ya que fue uno de los primeros en moldear y regir los esquemas o estructuras de la programación.
* La formas más recursiva de desarrollar programas en con la utilización de funciones.

REFERENCIAS

<https://www.google.com/search?q=paradigmas&rlz=1C1CHWL_esCO874CO874&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiVnc6-s4nmAhVSwlkKHUeEDOUQ_AUoAXoECBIQAw&biw=1034&bih=584#imgrc=OfqkbgrC4Fd2AM:>

<https://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/lang.html>

<https://paradigmaslenguajep.fandom.com/es/wiki/Paradigma_Declarativo>

<http://aprendeenlinea.udea.edu.co/lms/men_udea/mod/page/view.php?id=19537>