# Paradigmas de programación

01/02/2024

1. Unidad: Introducción
   1. Paradigmas de programación
   2. Clasificación de paradigmas
2. Unidad: El paradigma orientado a objetos
   1. Datos y lógica
   2. Clases y objetos
   3. Abstracción y encapsulamiento
   4. Herencia
   5. Polimorfismo
3. Unidad: El paradigma funcional
   1. Expresiones e instrucciones
   2. Funciones puras
   3. Inmutabilidad
   4. Funciones recursivas
   5. Procesamiento de datos y listas
4. Unidad: El paradigma lógico
   1. Expresiones y predicados
   2. Variables y predicados
   3. Argumentos y predicados
   4. Consultas y predicados
   5. Procesamiento de datos y listas

Bibliografía:

* Programación en C++  
  Deitel  
  Pearson
* Functional programming Principiles  
  Russel  
  Prestice hall
* Logic programming in depth  
  Cowsell  
  Pearson

02/02/2024

**Unidad 1: Antecedentes**

Paradigma de programación se refiere a como consideramos que un programa debe ser hecho y cual debería ser su significado y así existen dos estilos de paradigmas (a modo muy general).

Paradigmas imperativos y paradigmas declarativo. Un paradigma imperativo se asume que el programa este hecho de instrucciones, y el declarativo este hecho de expresiones.

Desde el punto de vista Imperativo un programa está hecho de instrucciones (la lógica del programa) y esas instrucciones deben manipular los datos del programa. Es decir que el paradigma imperativo existe una clara separación entre instrucciones y datos, las primeras manipulan a las segundas.

Por instrucción entendemos una orden o comando al procesador

Y luego el paradigma imperativo tiene (al menos) tres subparadigmas: Estructurado, procedural y orientado a objetos.

En el estructurado el programa esta hecho de una secuencia de instrucciones, la primera, llamada “punto de entrada” y hasta una última instrucción. Las instrucciones pueden ser:

* Instrucciones para “mover” datos
* Instrucciones matemáticas
* Instrucciones lógicas
* Instrucciones para control de flujo
* Otras

Normalmente un programa estructurado hace una clara separación entre instrucciones y datos. Y luego esta el paradigma procedural (y conste que procedural no es lo mismo que funcional), aquí el programa esta hecho de funciones o procedimientos que se combinan entre si para darle lógica al programa. Aquí existe:

Se podría considerar que el paradigma procedural es una evolución del paradigma estructurado.

* Una función principal que contiene el “punto de entrada” y le da salida a la lógica general del programa.
* Y una o mas funciones auxiliares que llevan a cabo una pequeña parte del propósito general del programa, y que son “invocadas” o “llamadas” por la función principal.

Normalmente el paradigma procedural se sigue haciendo esa clara distinción entre instrucciones y datos, pues estos últimos ahora existen como:

* Variables globales
* Variables locales
* Parámetros de funciones
* Valores de retorno de funciones.

Y entonces llega el paradigma Orientado a objetos. Ahora el programa esta hecho de objetos que interactúan entre si. Un objeto es un dato, pero es un dato que:

* Contiene su propia lógica (instrucciones)
* y sus propios datos

Ósea que el paradigma orientado a objetos ya no separa tan claramente a las instrucciones y a los datos. Sino que ahora las instrucciones son parte de los datos.

Y de la nada llega el paradigma declarativo. Ahora resulta que un paradigma no esta hecho de instrucciones, sino de expresiones.

Una instrucción es algo que al ejecutarse no necesariamente regresa un resultado, una expresión, en cambio, al ocurrir (¿ejecutarse?) siempre regresa algo.

Las expresiones son un concepto matemático, son asuntos de alto nivel de abstracción. Por esa razón la programación declarativa siempre tiene más sentido matemático y menos de procedimiento.

Y el paradigma declarativo luego se divide en dos sub-paradigmas:

* Funcional
* Lógico

Según el paradigma funcional los programas están hechos como la composición de “funciones puras”, algo así como:

Réstale (súmale (10,30), multiplícale (10,10))

(10+30) – (10\*10)

Y luego por allí anda el paradigma Lógico en donde los programas tienen forma de argumento lógico constituido con:

* Hechos
* Reglas  
  Y a partir de ellas podemos hacer:
* Consultas

Por ejemplo:

Hecho 1 Chucha es humana  
humana (Chucha)

Hecho 2 Pepa es humana

Humano(pepa)

Regla 1 Todos los humanos son mortales

Vx: [Humano(x) -> mortal(x)]

Consulta: ¿Hay algún mortal por allí?

Ǝx: mortal(x)

Y luego el lenguaje lógico responde que si, hay dos mortales aquí:

| X = Chucha

| X = pepa

| - yes

Todo lo que aquí se expone se resume en el “documento oficial” de la unidad disponible en la plataforma, justo en la página del curso y en la pestaña de la segunda unidad.

09/02/2024

Unidad 2: El paradigma orientado a objetos

El paradigma orientado a objetos o programación orientada a objetos (POO) es uno mas de los paradigmas de programación imperativa que:

1. Divide el paradigma en lógica (instrucciones) y datos.
2. Sigue considerando que el paradigma es un conjunto de instrucciones que modifican datos.

Los datos son prioritarios y cada dato tiene su propia lógica.

Pero es lo que es único y diferente es en que:

La principal razón de porque la POO tiene un punto de vista de lo que programar significa es por la seguridad de los datos: dado que cada dato tiene su propia lógica no se puede hacer lo que a uno le de la gana con los datos, solo lo que los datos nos permitan hacer.

Y para asegurar que la POO cumpla con este propósito, existen cuatro principios de diseño que se deben cumplir:

1. Clases y objetos
2. Abstracción y encapsulamiento
3. Herencia
4. Polimorfismo

Puesto que el enfoque de la POO son los datos, todo inicia dando forma a los datos, y esto se logra con clases y objetos. Luego, cada dato nos presenta solo aquella parte de el que quiera que conozcamos, otras partes las oculta, a esto se le llama abstracción y encapsulamiento.

Además, está el asunto del reúso de código, si ya una clase definió una lógica que alguien mas puede usar, pues que la “herede” del primero y que no la redefina de nuevo, y de esto se trata la herencia. Por último, si la lógica de una función puede aplicarse en diferentes situaciones y con diferentes datos, pues esa función debe ser polimórfica.

En plataforma podemos encontrar Ruby, como descargarlo, instalarlo y usarlo.

Ruby, nuestro lenguaje de programación POO

Ruby es un lenguaje de programación muy popular, algún día domino el mundo de los lenguajes POO (Hasta que llego Python) Que hace especial a Ruby?

* Como Python es un lenguaje muy fácil de dominar.
* Como Python es un lenguaje de “tipificación dinámica”.
* A diferencia de Python se aplica casi perfectamente a los principios de la POO.
* Es un lenguaje interpretado.

Interactuar con Ruby es sencillo:

* Hacemos un programa (“script”) de Ruby en un archivo de texto con extensión \*.rb
* Luego abrimos la consola / terminal / CMD de nuestro S.O., y en la carpeta de trabajo hacemos:
  + Ruby elscript.rb *enter*Eso de *enter* significa teclear ENTER

Esto hace que el intérprete de Ruby primero haga un análisis léxico y semántico al script “elscript.rb” (el programa a ejecutar), y si no hay errores que corregir, pues lo ejecuta (de manera interpretada).

La programación en Ruby no tiene una estructuración fija, pero en este caso todos nuestros programas lucen de la siguiente manera:

## Aquí definimos módulos y clases

## Y aquí la función principal

def ppal

puts “Hola mundo!!”

end

## El punto de entrada al programa

Ppal ()

elscript.rb