Tarea 1

Martin Felipe Espinal Cruces

4 de febrero de 2019

Lenguajes de programación

1. ¿Cuáles son las principales diferencias entre la programación imperativa y la declarativa?

La programación imperativa está basado en la máquina de Turing mientras que la declarativa se basa en el cálculo lambda. La programación imperativa describe paso a paso como resolver un problema mientras que el declarativo solo se encarga de describir el problema. El paradigma imperativo se describe en términos de instrucciones, condiciones, y pasos todo esto mientras va modificando su estado. Mientras que el declarativo se describe en términos de proposiciones y afirmaciones sin modificar su estado. En la programación imperativa nos encontramos con estructuras de control como *for, while, do while*, etc., en la programación declarativa el flujo del programa se da a través de la recursion o por medio de funciones de alto nivel como lo son *map, filter*, entre otros. En la programación declarativo nos encontramos con algo llamado *transparencia referencial*, esto quiere decir que al ejecutar una función determinada, esta no modificará ninguna fuera de ella, caso que no siempre ocurre en la programación imperativa, dado que si una variable se ve modificada por algún método, no tendremos la certeza de que algún otro método fuera de este se vea igualmente alterado. Finalmente en el paradigma declarativo no importa el orden de ejecución mientras que en el imperativo es fundamental el orden en el que se ejecuten las instrucciones.

2. Para los siguientes paradigmas de programación: POO, funcional, lógico y estructurado, investigue los siguiente:

- Características
- Diferencias
- Manejo de pila de ejecución
- Ejemplos de lenguajes

Programación orientada a objetos (POO)

- Características
  - Abstracción: Características y comportamientos esenciales de un objeto.
  - Encapsulamiento: Manera de contener los elementos de un objeto, que permite una alta cohesión.
  - Modularidad: Dividir el programa en pequeñas partes que se encargaran de dividir el problema.
  - Polimorfismo: Manera en que una clase hijo realiza de manera diferente comportamientos que hereda de una súper clase.
  - Herencia: Relación entre clases, con una jerarquía en la cual se transfieren características y comportamientos.
  - Recolección de basura: Por su traducción Garbage Collector es el encargado de designar memoria respecto a los objetos y las referencias a los mismos.
- Diferencias
  - POO vs funcional: La solución de la programación funcional se modela a través de describir el problema, sin indicar como lo hará, a diferencia de POO ya que en ella se debe indicar paso a paso la solución.
  - POO vs lógica: Mientras que POO se dedica a resolver un problema paso a paso, en la programación lógica se encarga los resuelve mediante sentencias.
  - POO vs estructurada: La gran diferencia que se observa entre estos paradigmas es la legibilidad del código, haciendo más fácil encontrar errores en el código, a diferencia de POO.
- Manejo de pila de ejecución
  - El manejo de la pila de ejecución es Last In First Out, esto quiere decir que la última llamada almacenada es la primera en salir.
- Ejemplos de lenguajes
  - Java
  - PHP
  - Phyton
  - Ruby
  - JavaScript

Programación funcional

- Características
  - Basado en el cálculo lambda.
  - Construido únicamente por funciones.
  - Capas de recibir como parámetro como argumentos funciones en funciones y devolver funciones como resultado (Funciones de orden superior).
  - Contienen funciones que no causan efectos secundarios (Funciones puras).
  - Funciones definidas en términos de si mismas (recursividad).
- Diferencias
  - Funcional vs POO: En algunos lenguajes de paradigma funcional como haskell existe el tipado estático fuerte, esto quiere decir que requiere que las variables sean declaradas con un tipo, mientras que en POO este tipado son expresados en tiempo de compilación.
  - Funcional vs Lógico: EL planteamiento del problema en la programación funcional es a través de la descripción del problema, mientras que en la lógica es a través de sentencias.
  - Funcional vs estructurado: Mientras que en ambos el código es legible (claro que esto depende del programador) en la programación estructurada nos encontramos con los famosos bucles como son while, for, y por el otro lado en la programación funcional estos son substituidos por la recursión.
- Manejo de pila de ejecución
  - En este paradigma podemos ver que como en muchos otros tenemos ventajas y desventajas, en este caso ganamos almacenamiento, puesto que no guardamos estado, pero perdemos en ocasiones rendimiento, puesto que los cálculos que se realizan son a través de lazy evaluation, que esto quiere decir que no se ejecutará una instrucción hasta que sea necesario.
- Ejemplos de lenguajes
  - Haskell
  - Racket
  - Elm
  - Erlang
  - Ocami

Programación Lógica

- Características
  - La mayoría de los lenguajes de programación se basan se basan en la teoría lógica del primer orden.
  - La manera de resolver los problemas es a través de sentencias.
  - construye base de conocimientos mediante reglas y hechos.
  - Reglas: Permite definir nuevas relaciones a partir de otras ya existentes.
  - Hecho:Es la forma más sencilla de sentencia.
  - Unificación de términos.
  - Mecanismos de inferencia automática.
  - Recursión como estructura de control básica.
- Diferencias
  - Lógica vs POO: Mientras que POO se dedica a resolver un problema paso a paso, en la programación lógica se encarga los resuelve mediante sentencias.
  - Lógica vs Funcional: EL planteamiento del problema en la programación funcional es a través de la descripción del problema, mientras que en la lógica es a través de sentencias.
  - Lógica vs estructurada: En la programación estructurada es muy importante la secuencia de instrucciones, mientras que en la programación lógica esto no tiene que ser precisamente así, ya que depende de la ejecución de las sentencias indicadas.
- Manejo de pila de ejecución
  - Puesto que este paradigma entra en el declarativo la evaluación y por ende la ejecución del código dependerá de las reglas de procedencia, las reglas de asociatividad y el uso de operadores, por consecuente la ejecución de una o más líneas de código se dará hasta que se cumplan las condiciones necesarias.
- Ejemplos de lenguajes
  - Prolog
  - Mercury
  - Logtalk
  - Curry
  - SequencelL

Programación estructurada

- Características
  - Estructura de control: Son aquellas partes de código que permite controlar el orden de ejecución de instrucciones establecidas.
  - Secuencia: Es el orden de ejecución de instrucciones del código, tal como están escritas.
  - Selección: Posibilidad de elegir entre la ejecución de dos instrucciones en base a la evaluación de una sentencia.
  - Iteración: Manera de repetir una instrucción, o un conjunto de ellas mientras se cumpla una condición.
  - Segmentación: Es modular el código de manera que guarden relación de forma jerárquica comunicándose a través de una lista de parámetros.
  - Identación: Permite la identificación de un bloque de código para ser ejecutado.
- Diferencias
  - Estructural vs POO: La gran diferencia que se observa entre estos paradigmas es la legibilidad del código, haciendo más fácil encontrar errores en el código, a diferencia de POO.
  - Estructural vs Funcional: Mientras que en ambos el código es legible (claro que esto depende del programador) en la programación estructurada nos encontramos con los famosos bucles como son while, for, y por el otro lado en la programación funcional estos son substituidos por la recuirción.
  - Estructural vs Lógica: En la programación estructurada es muy importante la secuencia de instrucciones, mientras que en la programación lógica esto no tiene que ser precisamente así, ya que depende de la ejecución de las sentencias indicadas.
- Manejo de pila de ejecución
  - Debido a que la ejecución del programa es en el orden en que fue escrito (a excepción de instrucciones como Go to, o las estructuras de control como los ciclos), será muy similar a como se muestre en el código.
- Ejemplos de lenguajes
  - Algol
  - Ada
  - Pascal
  - PL/I

Bibliografía

- Oscar Campos. (2011). Diferencias entre paradigmas de programación. 02/02/2019, de genbeta.com Sitio web: https://www.genbeta.com/desarrollo/diferencias-entre-paradigmas-de-programacion
- Alexander Rosales. (2012). Programacion Orientada a Objeto (POO) vs Programacion estructurada . 02/02/2019, de blogspot.com Sitio web: http://ingieneriadelsoftware2.blogsp orientada-objeto-y.html
- Sonia Joaquín Trejo Paola Marquez Alfredo. (2016). PROGRAMACIÓN LÓGICA Y FUNCIONAL . 02/02/2019, de blogspot.com Sitio web: https://programacionlogyfun.blogspo logica-y-funcional-la.html
- taniamcl. (2014). EVALUACIÓN DE EXPRESIONES.. 02/02/2019, de wordpress.com Sitio web: https://programacionlogicayfuncional.wordpress.com/2014/02/12/evaluacion-de-expresiones/
- S/A. (2018). Programación estructurada. 02/02/2019, de wikipedia Sitio web: https://es.wikipedia.org/wiki/Programaci %C3 %B3n_estructurada