**Lenguajes de Programación**

**TP1**

**Lenguajes de programación – Criterios y características de evaluación**

1)**¿Cómo podría clasificar en familias a los paradigmas asociados a los lenguajes de programación?**

La división de nivel superior distingue entre los lenguajes declarativos, en los cuales el foco está en lo que la computadora debe hacer, y los lenguajes imperativos, en los cuales el foco está en cómo debe hacerlo la computadora.

***Declarativos :***

***Los lenguajes funcionales emplean un modelo computacional basado en la definición recursiva de funciones. Se inspiran en el cálculo lambda, un modelo computacional formal desarrollado por Alonzo Church en la década de 1930.***

***Los lenguajes de flujo de datos modelan la computación como el flujo de información (tokens) entre nodos funcionales primitivos. Proporcionan un modelo inherentemente paralelo: los nodos son activados por la llegada de tokens de entrada y pueden operar simultáneamente.***

***Los lenguajes lógicos o basados en restricciones se inspiran en la lógica de predicados. Modelan la computación como un intento de encontrar valores que satisfagan ciertas relaciones específicas, utilizando la búsqueda dirigida a un objetivo a través de una lista de reglas lógicas.***

Imperativos:

Mientras que los lenguajes funcionales se basan en expresiones que tienen valores , los idiomas von Neumann se basan en declaraciones (asignaciones en particular) que influyen en el cálculo posterior a través del efecto secundario de cambiar el valor de la memoria.

En lugar de representar el cálculo como la operación de un procesador monolítico en una memoria monolítica, los lenguajes orientados a objetos lo representan como interacciones entre objetos semiindependientes, cada uno de los cuales tiene su propio estado interno y subrutinas para administrar ese estado.

Los lenguajes de script se distinguen por su énfasis en coordinar o "pegar" componentes extraídos de algún contexto.

2. ¿Cuál es la principal diferencia conceptual entre los lenguajes imperativos y los lenguajes declarativos?

La división de nivel superior distingue entre los lenguajes declarativos, en los cuales el foco está en lo que la computadora debe hacer, y los lenguajes imperativos, en los cuales el foco está en cómo debe hacerlo la computadora.

3. Explique brevemente a qué se denomina paradigmas *no convencionales*.

**PARADIGMAS NO CONVENCIONALES:** Las formalizaciones abstractas (MT, cálculo lambda, etc.) brindan formas de expresión y razonamiento que permiten obtener construcciones útiles y adecuadas para descubrir problemas. Redescubrir los formalismos como una forma de resolver problemas, utilizándolos en el diseño y la implementación.

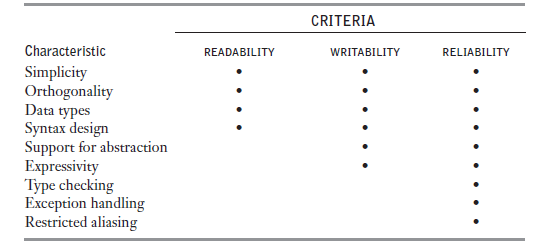
Reciben el nombre de *no convencionales* porque su forma de especificar y ejecutar está “alejada” de la arquitectura. Están alejados totalmente del control del paradigma imperativo.

Los no convencionales están más alejados de la arquitectura relacionados con la expresiones matemáticas y el razonamiento lógico. Además estos tienen que obligatoriamente respetar el paradigma ósea no se pueden salir , en contrapocision por ejemplo con java que es orientado a objeto pero nadie obliga a implentar un programa con pararigma impertivo.

Son los paradigmas que están más alejados de la arquitectura de la computadora y mas cercanos al lenguaje humano. Incluye a los paradigmas lógico, funcional y de computacion paralela

4)

¿Cuáles son los principales criterios y características utilizados para evaluar los lenguajes? Describa brevemente cada uno de ellos.



Readability:

Simplicidad: Cuando un lenguaje no es simple tiene un gran numero de constructores por lo tanto es mucho más difícil de aprender que uno con un numero menor. Él problema de legibilidad ocurren cada vez que el autor del programa ha aprendido un subconjunto diferente de aquel con el que el lector esta familiarizado.

**feature**

**multiplicity**—Tener más de una forma de realizar una operación en particular.

**operator overloading = el + se puede recargar con string e integer.**

Ortogonalidad : Básicamente significa que un conjunto pequeño de constructores

primitivos, puede ser combinado en número relativamente pequeño a la

hora de construir estructuras de control y datos. Cada combinación es legal

y con sentido.

Data type: algunos datos no son tan representativos.

5)

Es posible que un elemento de un lenguaje de programación beneficie una característica o criterio y a su vez perjudique a otro? Justifique, y en caso afirmativo muestre ejemplos.

* “C” y sus descendientes usan llaves para especificar declaraciones compuestas. Todos estos lenguajes tienen una **legibilidad disminuida** porque los grupos de sentencias siempre terminan de la misma manera, lo que dificulta determinar qué grupo se termina cuando aparece un “end” o una llave derecha.

Fortran 95 y Ada (ISO / IEC, 2014) aclaran esto usando un Sintaxis de cierre distinta para cada tipo de grupo de instrucciones. Por ejemplo, Ada usa “end if” para terminar una construcción de selección y “end loop” para terminar una construcción de bucle. Este es un ejemplo del conflicto entre **la simplicidad** que resulta en menos palabras reservadas, como en Java, y **la mayor legibilidad** que puede resultar del uso de palabras más reservadas, como en Ada.

6)

¿Por qué cree que es importante que un programador/desarrollador tenga conocimiento sobre los aspectos de diseño conceptuales de un lenguaje de programación?

El alcanze y La mayoría de estos conceptos son el tema de los capítulos de este libro: tipos, control (iteración, selección, recursión, no determinación, concurrencia), abstracción y denominación. Pensar en términos de estos conceptos hace que sea más fácil asimilar la sintaxis (forma) y la semántica (significado) de los nuevos lenguajes, en comparación con aprenderlos en el vacío. La situación es análoga a lo que sucede en los idiomas naturales: un buen conocimiento de las formas gramaticales facilita el aprendizaje de un idioma extranjero.

Sistaxis y Semantica :

7) Un lenguaje de programación es una notación formal que describe un modelo de la resolución de un problema. La especificación precisa de un LP es esencial para describir el comportamiento computacional del mismo. Sin una notación clara del efecto de los constructores del lenguaje es imposible determinar dicho comportamiento. En general, la especificación formal de los lenguajes es útil a varios propósitos:

- Ayuda a la comprensión del lenguaje

–Brinda soporte para la estandarización de lenguajes

- Guía en el diseño de lenguajes

- Ayuda al sistema de escritura del compilador y lenguaje

- Soporta la verificación de correctitud de los programas

- Modela la especificación de software

8) Describa claramente las diferencias entre la semántica estática y la semántica dinámica.

Lo que debe verificarse en tiempo de ejecución (o en las fases posteriores de un intérprete) se conoce como la **semántica dinámica.**Es el significado de las expresiones, declaraciones y unidades de programa de un lenguaje de programación**.**

**Ejemplos:**

* Las variables nunca se usan en una expresión a menos que se les haya dado un valor.
* Los punteros nunca se des referencian a menos que se refieran a un objeto válido.
* Las expresiones de subíndice de matriz se encuentran dentro de los límites de la matriz.
* Las operaciones aritméticas no se desbordan.

En cambio la **semántica estática,** es definida por la gramática de atributos y puede verificarse en tiempo de compilación. Algunos ejemplos de semántica estática:

* La regla común de que todas las variables deben declararse antes de hacer referencia a ellas.
* Compatibilidad de tipos : no se puede asignar un valor de coma flotante a un tipo entero variable, aunque lo contrario es legal.

9) ¿Qué tipo de gramática son utilizadas para describir la sintaxis de un lenguaje de programación? ¿Por qué ?

La *sintaxis* de un LP es el conjunto de reglas que describen la estructura de de la mayoría de los programas válidos en el lenguaje. Afecta a la legibilidad, pero no a la expresabilidad del lenguaje.

Los métodos para describir la sintaxis son:

- *Gramáticas Libres de Contexto* – BNF – Diagramas sintácticos

- *BNF Extendida* (mejora la facilidad de lectura y escritura de las BNF tradicionales)

- *Gramáticas y Reconocedores*

**(Consultar en consulta(preguntar ¿Por qué?)**

**10)**

**Resumen**

Una descripción operacional concreta usa un autómata intérprete que define la semántica de cada constructor.

El intérprete actúa como una computadora virtual: cada constructor del lenguaje se traduce a una secuencia de instrucciones que van a ser ejecutadas por una computadora virtual.

La *especificación semántica operacional de un LP* L a partir de un autómata concreto requiere la definición de:

- Un LP de bajo nivel M que brinde un repertorio simple y reducido

- Un traductor que convierta una instrucción en el lenguaje L al lenguaje de bajo nivel M

- Una máquina virtual que ejecute el LP M

El cambio de estado en la máquina virtual causado por la ejecución del código traducido de una sentencia en el

lenguaje L, define el significado de la sentencia  Este método describe el significado de las sentencias de un

lenguaje de alto nivel en términos de sentencias en un lenguaje de bajo nivel más simple.

Otra definición: (libro 158/159 libro concept of programming)

La idea detrás de la semántica operativa es describir el significado de una declaración o programa especificando los efectos de ejecutarlo en una máquina, los efectos en la máquina se ven como la secuencia de cambios en su estado, donde el estado de la máquina es la colección de valores en su almacenamiento.

La descripción semántica, entonces, se da al ejecutar una versión compilada de el programa en una computadora.

**Otra cosa importante**

En lo mas alto nivel, el interés está en el resultado final de la ejecución de un programa completo.

Esto a veces se llama semántica operacional natural.

En el nivel más bajo, la semántica operativa se puede usar para determinar el significado preciso de un programa a través de un examen de la secuencia completa de cambios de estado que ocurrir cuando se ejecuta el programa. Este uso a veces se llama estructural semántica operacional.

Ventajas:

La mayoría de los programadores tienen, al menos en una ocasión, un pequeño programa de prueba para determinar el significado de lo que construyó.

Desventajas:

Existen varios problemas con el uso de este enfoque para completar formalmente Descripciones semánticas. Primero, los pasos individuales en la ejecución de la máquina. el idioma y los cambios resultantes en el estado de la máquina son demasiado pequeños y demasiado numeroso.

Segundo, el almacenamiento de una computadora real es demasiado grande y complejo.

Por lo general, hay varios niveles de dispositivos de memoria, así como conexiones a

enumerables otras computadoras y dispositivos de memoria a través de redes. Por lo tanto,

los lenguajes de máquina y las computadoras reales no se usan para operaciones formales

semántica

ejemplo: pagina 160(concept of programming)