Práctica 10: Paradigmas de Leng. de Programación

Ejercicio 1: Un programa en un lenguaje procedural es una secuencia de instrucciones o comandos que se van ejecutando y producen cambios en las celdas de memoria, a través de las sentencias de asignación. ¿Qué es un programa escrito en un lenguaje funcional? y ¿Qué rol cumple la computadora?

Un programa en un lenguaje funcional es una serie de funciones matemáticas que resuelven problemas mediante composición y aplicación de funciones sin efectos secundarios. La computadora interpreta y ejecuta estas funciones, realizando los cálculos para obtener los resultados deseados.

Ejercicio 2: ¿Cómo se define el lugar donde se definen las funciones en un lenguaje funcional?

Las funciones se definen en un script, que incluye un área para declarar funciones (explícitamente o con inferencia de tipos) y otra para ejecutarlas.

Ejercicio 3: ¿Cuál es el concepto de variables en los lenguajes funcionales?

Las variables en lenguajes funcionales son variables matemáticas que almacenan valores o resultados, no celdas de memoria modificables, y cumplen con transparencia referencial.

Ejercicio 4: ¿Qué es una expresión en un lenguaje funcional? ¿Su valor de qué depende?

Una expresión es un bloque de código que se evalúa para producir un valor sin efectos secundarios. Su valor depende de las subexpresiones que la componen y del entorno donde se definen las variables.

Ejercicio 5: ¿Cuál es la forma de evaluación que utilizan los lenguajes funcionales?

Los lenguajes funcionales usan evaluación perezosa (lazy), donde las expresiones se evalúan solo cuando su valor es necesario, o estricta, donde las subexpresiones se evalúan inmediatamente. El resultado es independiente del método.

Ejercicio 6: ¿Un lenguaje funcional es fuertemente tipado? ¿Qué tipos existen? ¿Por qué?

Sí, los lenguajes funcionales son fuertemente tipados para garantizar seguridad, corrección, optimización y razonamiento formal. Tipos:

- Básicos: Entero, Flotante, Booleano, Carácter.
- Derivados: Pares, funciones, tipos polimórficos (e.g., β -> β).

Razones: detección de errores en compilación, abstracción, eficiencia y base en lógica matemática.

Ejercicio 7: ¿Cómo definiría un programa escrito en POO?

Un programa en POO es un conjunto de objetos que interactúan mediante mensajes, instanciados desde clases con variables y métodos, utilizando encapsulamiento, herencia y polimorfismo para resolver problemas.

Ejercicio 8: Diga cuáles son los elementos más importantes y hable sobre ellos en la programación orientada a objetos.

- Clases: Plantillas que definen propiedades y comportamiento.
- Objetos: Instancias de clases con estado y comportamiento.
- Encapsulamiento: Oculta el estado interno, accesible solo por métodos.
- Herencia: Permite a una clase derivar propiedades de otra.
- Polimorfismo: Objetos de distintas clases responden diferente al mismo mensaje.
- Mensajes: Peticiones que activan métodos entre objetos.

Ejercicio 9: La posibilidad de ocultamiento y encapsulamiento para los objetos es el primer nivel de abstracción de la POO, ¿cuál es el segundo?

El segundo nivel de abstracción en POO es la herencia, que permite agrupar clases en jerarquías de superclases y subclases.

Ejercicio 10: ¿Qué tipos de herencias hay? Cuál usa Smalltalk y C++

- Simple: Una clase deriva de una sola clase base.
- Múltiple: Una clase deriva de varias clases base.

Smalltalk: Usa herencia simple.

C++: Soporta herencia simple y múltiple, además de multinivel, jerárquica e híbrida.

Ejercicio 11: En el paradigma lógico ¿Qué representa una variable? ¿y las constantes?

Variable: Representa elementos indeterminados que pueden sustituirse por constantes (e.g., X en humano(X) puede ser juan) y permite definir hechos.

Constante: Representa elementos fijos del dominio, como strings en minúsculas (juan) o números.

Ejercicio 12: ¿Cómo se escribe un programa en un lenguaje lógico?

- Hechos: Afirmaciones sobre relaciones que son verdaderas en el dominio. Se escriben como predicados con constantes o términos.
  - tiene(coche, ruedas)
- Reglas: Relaciones lógicas que definen cómo se derivan nuevos hechos a partir de otros hechos o reglas, usando implicaciones.
  - conclusión :- condición
- Consultas: Preguntas que el usuario plantea al sistema para obtener soluciones basadas en los hechos y reglas.

Ejercicio 13: Teniendo en cuenta el siguiente problema, se lee una variable entera por teclado y si es par se imprime “El valor ingresado es PAR” y si es impar imprime “El valor ingresado es impar”, implemente este ejemplo en cada uno de los paradigmas presentados en esta práctica.

Java (POO):

```java
public class parOImpar {
    public String esPar() {
        Scanner scanner = new Scanner(System.in);
        int num1 = scanner.nextInt();
        if (num1 % 2 == 0)
            return "El valor ingresado es PAR";
        return "El valor ingresado es impar";
    }
}
```

Haskell (Funcional):

```haskell
esPar :: Integer -> Bool
esPar n = n `mod` 2 == 0
main :: IO ()
main = do
    putStrLn "Ingrese un num"
    input <- getLine
    let n = read input :: Integer
    let resultado = if esPar n then "El valor ingresado es PAR" else "El valor ingresado es impar"
    putStrLn resultado
```

Prolog (Lógico):

```prolog
esPar(N) :- N mod 2 =:= 0.
esImpar(N) :- N mod 2 =\= 0.
main :-
    write('Ingrese numero: '),
    read(N),
    (esPar(N) -> write('El valor ingresado es PAR'), nl ;
    esImpar(N) -> write('El valor ingresado es impar'), nl).
```

Pascal (Imperativo):

```pascal
program ParOImpar;
var
  num: integer;
begin
  writeln('Ingrese un numero: ');
  readln(num);
  if (num mod 2 = 0) then
    writeln('El valor ingresado es PAR')
  else
    writeln('El valor ingresado es impar');
end.
```

Ejercicio 14: Describa las características más importantes de los Lenguajes Basados en Scripts. Mencione diferentes lenguajes que utilizan este concepto. ¿En general, qué tipificación utilizan?

- Ejecución interpretada, sin compilación previa.
- Sintaxis simple y flexible para fácil escritura y lectura.
- Alta productividad para prototipado y automatización.
- Integración con sistemas (SO, bases de datos, web).
- Gestión dinámica de recursos (garbage collection).
- Portabilidad multiplataforma.

Lenguajes: Python, JavaScript, Ruby, Perl, PHP, Bash, Lua.

Tipificación: Generalmente usan tipado dinámico, donde los tipos se determinan en tiempo de ejecución (e.g., x = 5 puede cambiar a x = "texto" en Python). Excepciones como TypeScript ofrecen tipado estático opcional.

Ejercicio 15: ¿Existen otros paradigmas? Justifique la respuesta

Sí, existen otros paradigmas:

- Imperativa: Basada en secuencias de comandos que modifican el estado (e.g., C, Pascal).
- Procedimental: Organiza código en procedimientos (e.g., Fortran, Ada).
- Estructurada: Usa estructuras de control para evitar código desorganizado (e.g., ALGOL, C).
- Basada en eventos: Responde a eventos externos (e.g., JavaScript, Visual Basic).
- Concurrente: Maneja tareas simultáneas con sincronización (e.g., Go, Erlang).
- Basada en agentes: Usa entidades autónomas que interactúan (e.g., IA).
- Basada en restricciones: Resuelve relaciones automáticamente (e.g., CPLEX, Prolog).
- Basada en datos: Manipula flujos de datos (e.g., SQL).
- Reactiva: Responde a flujos de datos o eventos en tiempo real (e.g., RxJava, Elm).
- Orientada a aspectos: Separa responsabilidades transversales (e.g., AspectJ, Spring AOP).