Clase 11

Excepciones

Un **paradigma de programación** es un estilo de desarrollo de programas, un modelo para resolver problemas computacionales. Los lenguajes de programación usan varios paradigmas a la vez a partir del tipo de órdenes que permiten implementar, está relacionado directamente con su sintaxis

Los principales son:

- Imperativo: Sentencias + sentencias de comandos
- Declarativo: Los programas describen los resultados esperados sin listar los pasos a llevar a cabo para alcanzarlos. Uno declarativo puede ser a su vez lógico, que se basa en aserciones lógicas.
- Funcional: Los programas se componen de funciones.
- Orientado a objetos: métodos + mensajes

Otra clasificación reciente los divide en:

- Dirigido por eventos: El flujo está determinado por sucesos externos (como un usuario que interactúa)
- Orientado a aspectos: Apunta a dividir el programa en módulos independientes, cada uno con comportamiento y responsabilidad bien definido

Programación lógica

Es un tipo de paradigma dentro del paradigma declarativo. Los programas son una serie de aserciones lógicas. El conocimiento se representa a través de reglas y hechos, los objetos se representan por términos, los cuales tienen constantes (determinado) y variables (indeterminado). PROLOG es el lenguaje más utilizado.

“humano(juan)” . Las constantes son string de letras en minúsculas o string de dígitos

Termino compuesto: Son un functor seguido de un número fijo de argumentos entre paréntesis los cuales son términos. Aridad es el número de argumentos, estructura es como se le dice a un término cuyos argumentos no son variables

Ejemplo:

padre -> constante (cadena en minúscula)

Longitud -> Variable (no es una cadena en minúscula)

Tamaño(4,5) -> estructura (un functor y los parámetros entre paréntesis, que son dígitos ósea que son constantes)

La constante [] representa una lista vacía, el “.” Es una lista de un elemento. Entonces .(Alpha,[]) es una lista con un solo elemento que es alpha. También se puede usar [] en vez de .() y usar un | en vez de una coma

Entonces queda [Alpha | []]

Hecho

son relaciones entre objetos y siempre son verdaderas:

tiene(coche,ruedas): representa el hecho que un coche tiene ruedas

longuitud([],0): representa el hecho que una lista vacía tiene longitud cero

moneda(peso): representa el hecho que peso es una moneda.

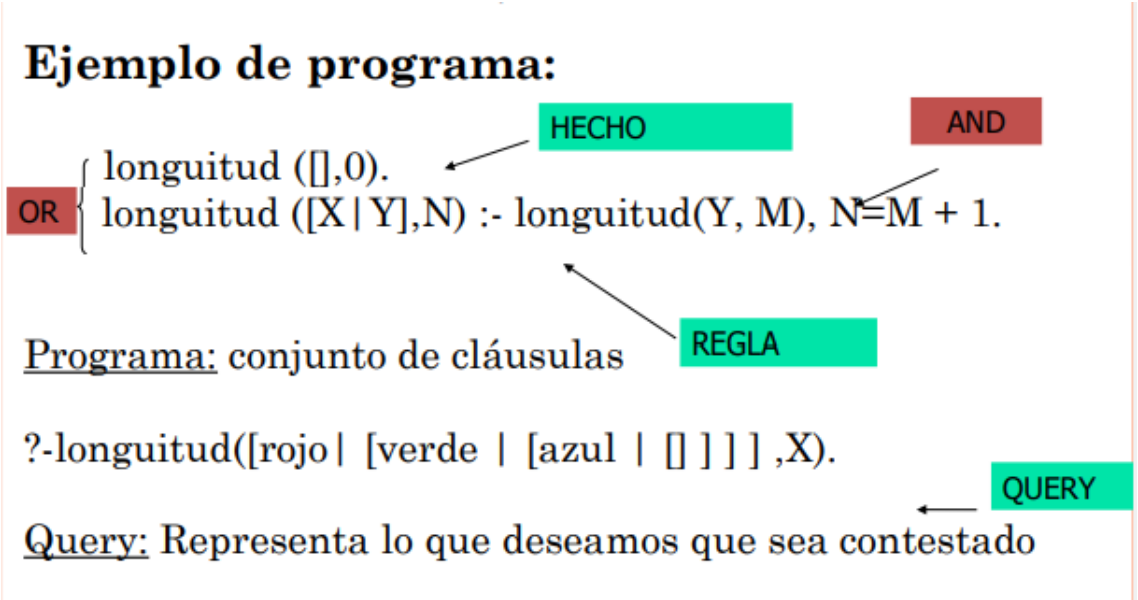
Regla

Cláusulas de Horn

La cláusula tiene la forma conclusión :- condición

Donde :- es If, conclusión es un predicado y condición es una conjunción de predicados separados por comas, representando un AND lógico.

Seria como decir if condición else conclusión.



En estos programas se van haciendo consultas, que es un Query. El programa va deduciendo a partir de los hechos y reglas y con eso responde la consulta.

Programación orientada a objetos

- Un programa escrito con un lenguaje OO es un conjunto de objetos que interactúan mandándose mensajes.
- Los elementos son: objetos, mensajes, métodos y clases.
- Los objetos son datos abstractos, que tienen estado interno y comportamiento.
- Los mensajes son peticiones de un objeto a otro para que este se comporte de una determinada manera.
- Los métodos son programas asociados a un objeto y cuya ejecución solo puede desencadenarse a través de un mensaje recibido por este o por sus descendientes
- Las clases son tipos definidos por el usuario que determina las estructuras de datos y las operaciones asociadas con ese tipo. Cada objeto pertenece a una clase y este tiene su funcionalidad.

- Instancia de clase: Cada vez que se construye un objeto se crea una instancia de esa clase. Es un objeto individual por los Valores que tomen sus atributos
- También hay herencia, donde una clase A hereda todas las propiedades de su superclase B
- El polimorfismo es la capacidad que tienen los objetos de distintas clases de responder a mensajes con el mismo nombre.

Paradigma funcional

- Basado en el uso de funciones. Muy popular en la resolución de problemas de inteligencia artificial, matemática, lógica, procesamiento paralelo.
- Ventajas: Vista uniforme de programa y función, se tratan las funciones como datos, liberación de efectos colaterales, manejo automático de memoria.
- Desventajas: Ineficiencia en ejecución.
- Proveen un conjunto de funciones primitivas, así como un conjunto de formas funcionales. Además tienen una semántica basada en valores, transparencia referencial, una regla de mapeo basada en combinación o composición y las funciones son de primer orden.
- El VALOR más importante en la programación funcional es el de una FUNCION. Matemáticamente una función es una correspondencia A → B. Las funciones son tratadas como valores, pueden ser pasadas como parámetros, retornar resultados, etc. Básicamente una función es el valor que retorna.
- Se debe distinguir entre el valor y la definición de una función. Se puede definir una función de muchas formas pero siempre dará el mismo valor (a diferente sintaxis el resultado siempre es el mismo)
- La expresión es la noción central de la programación funcional. Una expresión es su valor. El valor de una expresión depende UNICAMENTE de los valores de las sub-expresiones que la componen, las expresiones pueden contener variables.
- Las variables son variables matemáticas, no la de celda de memoria.
- No existen efectos laterales y dos expresiones sintácticamente iguales darán el mismo valor.
- Un script es una lista de definiciones y pueden someterse a evaluación.
- Algunas expresiones pueden NO llegar a reducirse del todo, a estas expresiones se las denominan canónicas , pero se les asigna un valor indefinido y corresponde al símbolo bottom(^)
- Por lo tanto toda expresión siempre denota un valor.

PROGRAMACIÓN FUNCIONAL

Evaluación de las expresiones:

La forma de evaluar es a través de un mecanismo de REDUCCIÓN o SIMPLIFICACIÓN

Ejemplo:

cuadrado (3 + 4)
=> cuadrado 7 (+)
=> 7 * 7 (cuadrado)
=> 49 (*)

Otra forma sería:

cuadrado (3 + 4)
=> (3 + 4) * (3 + 4) (cuadrado)
=> 7 * (3 + 4) (+)
=> 7 * 7 (+)
=> 49 (*)

52

"No importa la forma de evaluarla, siempre el resultado final será el mismo"

- Existen dos formas de reducción, una es con orden aplicativo donde siempre evalúa los argumentos los necesite o no, y la otra es con orden normal donde no se calcula más de lo necesario, la expresión no es evaluada hasta que su valor se necesite y una expresión compartida no es evaluada más de una vez. También tenemos Haskell en donde se mantiene todo en memoria.
- Los tipos pueden ser básicos , como los int y float, los bool y los char. O derivados (como si fuera definido por el usuario)
    - Toda función tiene asociado un tipo. No debería haber errores de tipo
- Se consideran expresiones de tipo polimórficas a las funciones que no es fácil deducir su tipo, se utilizan letras griegas para tipos polimórficos.
- La currificación es el mecanismo que reemplaza argumentos estructurados por argumentos más simples.
- Calculo lambda: modelo para definir funciones.