Nombre del alumno: Johanne Joaquín Arriaga Díaz

Nombre del profesor: María Isabel Roblero Ordoñez.

Nombre del trabajo: Súper nota de unidad III.

Materia: Programación lógica.

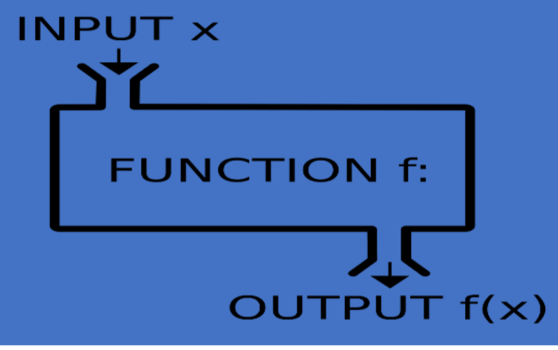
Grado: Octavo cuatrimestre

Grupo: ISC13SDC0119-F

Frontera Comalapa, Chiapas a 04 de Abril de 2022

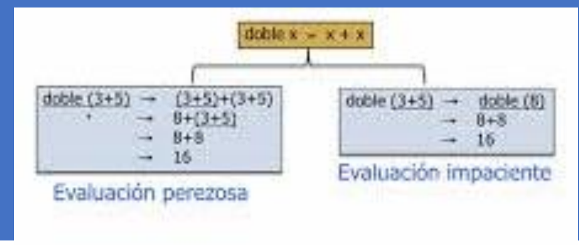
La estrategia de evaluación perezosa.

Programar en funcional es pensar en funciones. Cualquier algoritmo puede pensarse como un encadenamiento de expresiones funcionales. Cualquier lenguaje de programación dentro de este paradigma se caracteriza por un comportamiento específico acerca de cómo evalúa estas construcciones.

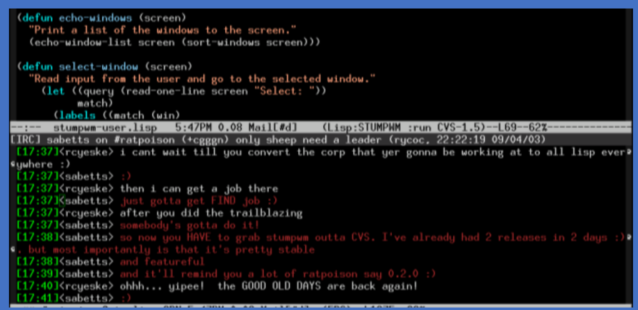


Evaluación Perezosa: prescribe el momento del tiempo en el que se deben evaluar las expresiones en el código fuente. Posterga la evaluación de la expresión hasta que su valor es realmente demandado.

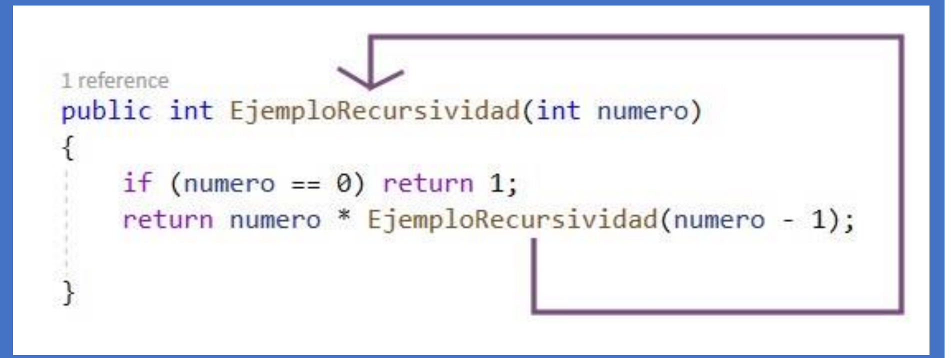
La Evaluación impaciente: evalúa cada expresión en el momento exacto del tiempo en que ésta es encontrada dentro del código fuente.



Codificación: La evaluación perezosa permite alcanzar un modelo de programación defensivo basado en aserciones que resulta generalmente más sinóptico y sencillo de leer y entender.



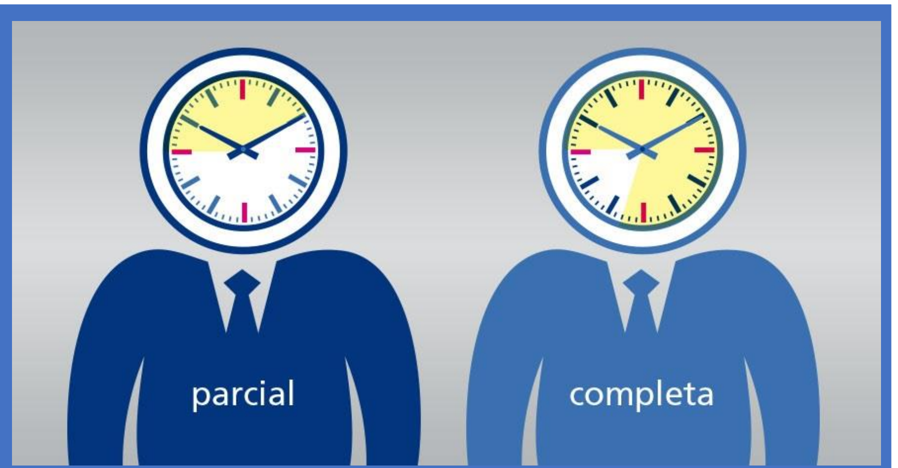
Recursión: Los esquemas de diseño funcional recursivo también se prestan mucho a hacer uso de este tipo de construcciones.



Rendimiento: La evaluación perezosa es relevante en el rendimiento. La ventaja de la evaluación perezosa es que el proceso de encadenamiento compositivo por operaciones de disyunción lógica se para automáticamente en cuanto se encuentra la primera coincidencia ahorrando ciclos de cómputo.

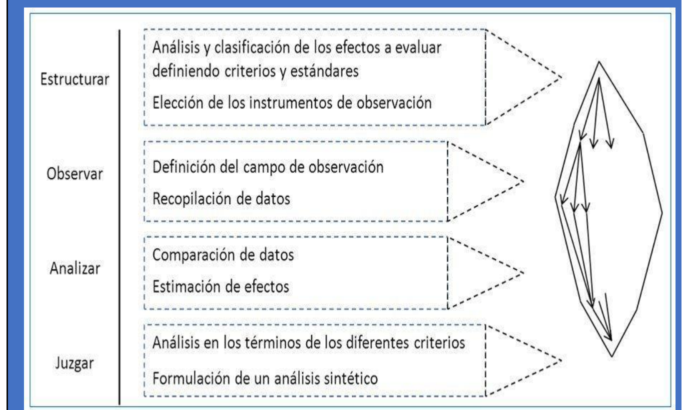
Evaluación completa, requiere disponer de todos los argumentos actuales antes de invocar a una función.

La evaluación parcial, por el contrario, permite proporcionar sólo parte de los argumentos para obtener otra función con aridad reducida.



Evaluación por Fases: Independendiente de que se desarrolle un diseño por fases o por currificación, se resuelven las dimensiones paramétricas de una función en distintas fases del tiempo.

Evaluación por Pares: Existen 2 agentes que articulan una colaboración. El agente cliente pide una función al agente proveedor y éste la resuelve de forma parcial antes de devolverla. Así se fija parcialmente el comportamiento de dicha función para adaptarlo a las condiciones ambientales del cliente.



Evaluación Inmediata vs Evaluación Diferida

Momento del tiempo en que una función se ejecuta tras su invocación. Nuevamente, en esta dimensión del discurso, distinguimos dos modelos contrapuestos.
La **Evaluación Inmediata** lanza a ejecución una función en el mismo momento en que se procesa su invocación. La **Evaluación Diferida**, por el contrario, permite trabajar con funciones cuya invocación puede postergarse en el tiempo hasta un momento más conveniente.

Técnicas de programación funcional perezosa.

Los beneficios de la evaluación perezosa son:

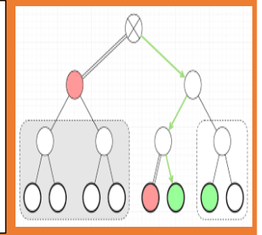
1. Incremento en rendimiento al evitar cálculos innecesarios, y en tratar condiciones de error al evaluar expresiones compuestas.
2. La capacidad de construir estructuras de datos potencialmente infinitas.
3. La capacidad de definir estructuras de control como abstracciones, en lugar de operaciones primitivas.
4. Puede reducir el consumo de memoria, ya que los valores se crean solo cuando se necesitan.

Técnica de Backtracking

Si una alternativa falla, el flujo retrocede hasta la última posición o intenta de nuevo.

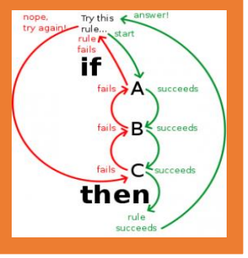
¿Qué es backtracking?

Cuando un problema no tiene un método algorítmico para resolverse, la única forma de resolverlo es la búsqueda exhaustiva de soluciones entre todas las posibiles; este método se conoce como fuerza bruta: se generan todos los casos posibles y se testean uno a uno hasta encontrar las soluciones (a veces basta con encontrar una, en otras ocasiones hay que encontrar todas ellas, o quedarse con la mejor).



¿Cómo funciona backtracking?

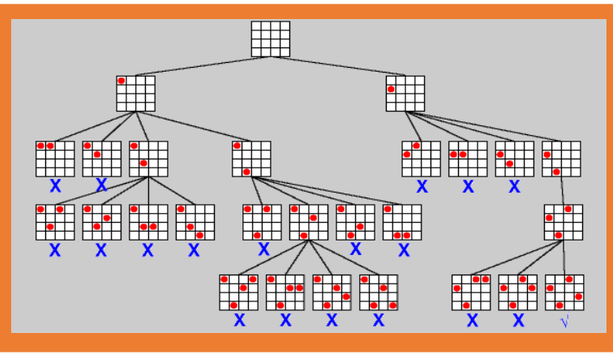
Se asemeja a un recorrido del árbol de expansión; se recorre en preorden: se evalúa el nodo raíz o actual, y después sus hijos de izquierda a derecha. Y hasta que no se termina de generar una solución parcial (válida o no) no se evalúa otra solución distinta.
En los nodos del nivel k del árbol se encuentran las soluciones parciales formadas por k etapas o decisiones.



Costes y eficiencia en backtracking

La forma de generar y escoger entre las posibilidades determina la forma del árbol, la cantidad de descendientes de un nodo, la profundidad del árbol, etc., y determina la cantidad de nodos y la eficiencia del algoritmo, pues el tiempo de ejecución depende del número de nodos que se generen y se visiten. En general, la eficiencia depende de:

1. el tiempo necesario para generar las decisiones posibles en el punto (2)
2. la cantidad de decisiones posibles que se obtienen en (2)
3. incorporar una decisión a una solución parcial en el punto (3)
4. el número de soluciones parciales que satisfacen es_completable
5. y en mucha menor medida, del tiempo que se tarde en comprobar es solución y tratar solución.

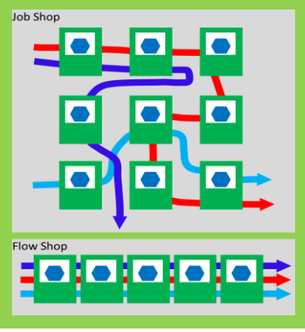


Métodos

Variante de flujo regular o flow shop scheduling

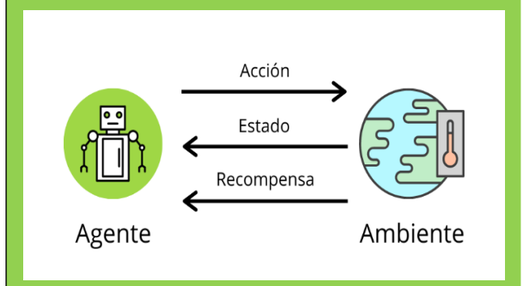
En la teoría de scheduling se pueden distinguir un gran número de problemas. Se tiene un conjunto de N trabajos procesados sobre un conjunto de M máquinas físicas siguiendo un patrón de flujo o ruta tecnológica.

Una secuenciación consiste en encontrar para cada trabajo un tiempo o un intervalo de tiempos en los que este pueda procesarse en una o varias máquinas [4, 27]. El objetivo es encontrar una secuencia sujeta a una serie de restricciones que optimice una o varias funciones objetivo [20, 24, 27].



Aprendizaje reforzado

El AR es un enfoque de la IA en el que los agentes aprenden a partir de su interacción con el ambiente. Es aprender qué acción tomar dada una situación determinada con el objetivo de maximizar una señal numérica de recompensa que da la medida de cuán buena fue la acción elegida por el agente. En el paradigma del AR un agente se conecta a su ambiente mediante la percepción y acción. Los métodos del aprendizaje reforzado exploran el ambiente todo el tiempo para obtener una política deseada.



Adaptación del algoritmo q-learning para la solución del FSSP

QL es un algoritmo del AR que se basa en la capacidad de aprendizaje de los agentes asociados al método. Este algoritmo trabaja aprendiendo de una función acción-valor que da la utilidad esperada de tomar una acción en un estado determinado. El núcleo del algoritmo es la actualización de un q-valor en cada iteración.