CLASE 3

1. ¿Qué son los Tipos Algebraicos Recursivos?

Son tipos de datos que se definen en términos de sí mismos. Esto permite crear estructuras de datos que pueden crecer indefinidamente, como listas o árboles.

- **Componentes Esenciales:**
    - **Caso Base:** Al menos un constructor que **no** es recursivo. Sirve como punto de finalización de la estructura. Ejemplos: [] en las listas, Prepizza en el tipo Pizza, o Armario en Dungeon.
    - **Caso Recursivo:** Al menos un constructor que tiene un argumento del **mismo tipo** que se está definiendo. Ejemplos: (x:xs) en las listas, Capa Ingrediente Pizza en Pizza, o Habitacion Objeto Dungeon Dungeon en Dungeon.
- **Clasificación:**
    - **Recursivos Lineales:** El constructor recursivo se llama a sí mismo solo una vez (ej: Pizza).
    - **Recursivos Ramificados (Árboles):** El constructor recursivo se llama a sí mismo múltiples veces (ej: Dungeon), creando una estructura de árbol.

---

2. El Principio de Recursión Estructural: La "Receta" para Resolver Funciones

Es la técnica principal para trabajar con tipos recursivos. Sigue una "receta" simple y poderosa que se adapta a la estructura del dato:

1. **Un Caso por Constructor:** Tu función debe tener una definición (una ecuación) para cada constructor del tipo de dato.
    - f Prepizza = ...
    - f (Capa ing p) = ...
2. **Llamada Recursiva en Partes Recursivas:** En el caso recursivo, **siempre** se llama a la misma función sobre los argumentos que son del tipo recursivo.
    - Para f (Capa ing p), la llamada recursiva es f p.
    - Para g (Habitacion obj d1 d2), hay dos llamadas recursivas: g d1 y g d2.
3. **Combina los Resultados:** El paso final es pensar cómo combinar el resultado de la llamada recursiva con los otros datos del constructor para obtener el resultado final.
    - En f (Capa ing p) = ... (f p) ..., debes decidir qué hacer con el ingrediente ing y el resultado de f p.

---

3. Ejemplos Prácticos y Patrones Comunes

- **Ejemplo Pizza (Lineal):**
    - **Definición:** data Pizza = Prepizza | Capa Ingrediente Pizza
    - **Función cantQueso:**

        - **Caso Base:** cantQueso Prepizza = 0 (Una prepizza no tiene queso).
        - **Caso Recursivo:** cantQueso (Capa ing p) = unoSiQueso ing + cantQueso p (La cantidad de queso es 1 si el ingrediente actual es queso, más la cantidad de queso en el resto de la pizza).
    - **Uso de Funciones Auxiliares:** Es muy común y útil crear funciones auxiliares para procesar los datos no recursivos (como el Ingrediente).
- **Ejemplo Dungeon (Árbol Binario):**
    - **Definición:** data Dungeon = Armario | Habitacion Objeto Dungeon Dungeon
    - **Función cantidadDeOro:**
        - **Caso Base:** cantidadDeOro Armario = 0
        - **Caso Recursivo:** cantidadDeOro (Habitacion obj d1 d2) = unoSiEsOro obj + cantidadDeOro d1 + cantidadDeOro d2 (La cantidad de oro es 1 si el objeto actual es oro, más el oro de la rama izquierda, más el oro de la rama derecha).
- **Ejemplo objsDelCaminoMasLargo (Función más compleja):**
    - Esta función muestra un patrón clave: a veces, la combinación de resultados no es una simple suma.
    - Aquí, se comparan las longitudes de los resultados recursivos para decidir cuál elegir: obj : elegirEntre (objsDelCaminoMasLargo d1) (objsDelCaminoMasLargo d2).
    - La función auxiliar elegirEntre contiene la lógica para comparar los caminos y devolver el más largo.

---

4. Árboles Binarios Genéricos (Tree a)

Para representar solo la estructura de un árbol sin atarse a un tipo de dato específico (como Objeto), se usan parámetros de tipo.

- **Definición:** data Tree a = EmptyT | NodeT a (Tree a) (Tree a)
    - a es un parámetro que puede ser reemplazado por cualquier tipo (Int, String, etc.).
    - Esto permite crear árboles de enteros, árboles de caracteres, etc., con la misma definición estructural.