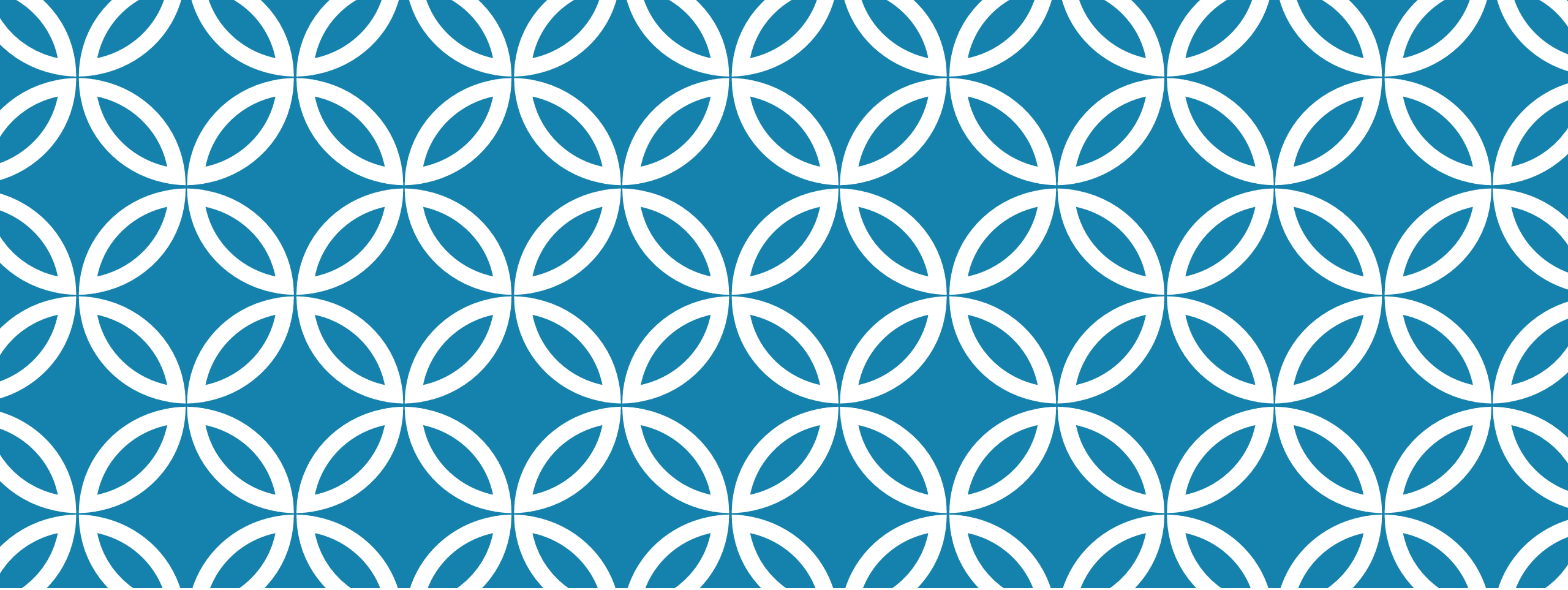


ESTRUCTURAS DISCRETAS

Estructuras de Datos y Funciones Recursivas

TUPLAS

Son estructuras de datos, es decir, sirven para almacenar elementos.

En Haskell son **heterogéneas,** almacenan elementos de diferentes tipos.

SINTAXIS Y NOMENCLATURA

Se utilizan paréntesis y los elementos son separados por comas.

(2.9, 5.6)

(“Juan”, 5566778899, True)

(1,2,3,4,5,6)

(‘x’, 1, ‘y’, 5, ‘z’, 7.4)

( )

Las tuplas de n elementos se conocen como n-tuplas.

Las tuplas de dos elementos (2-tuplas) se conocen usualmente como **pares.**

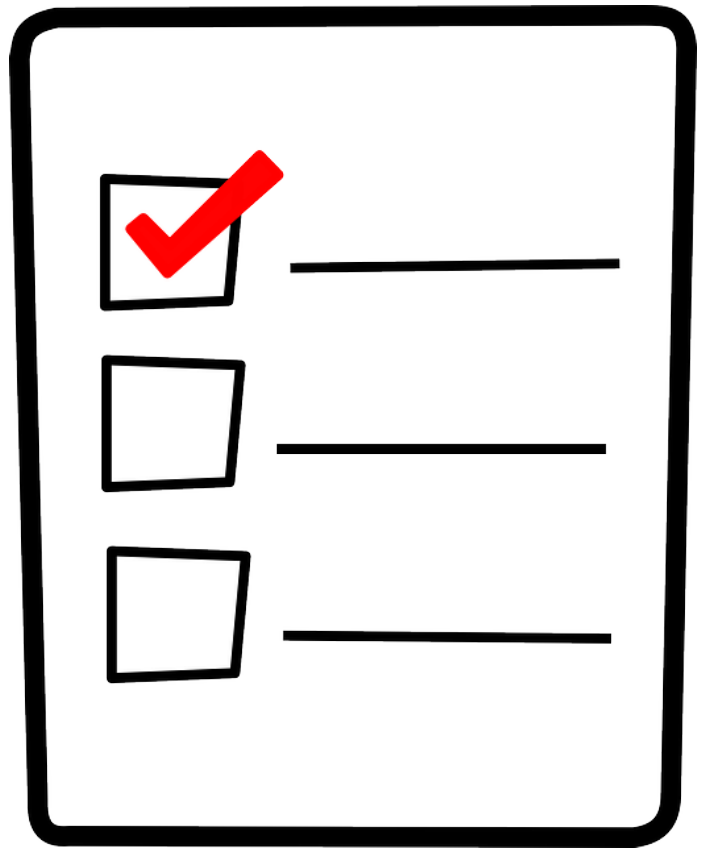
las tuplas de 3 elementos (3-tuplas) se conocen como **tercias.**

LISTAS

Son la estructura de datos mas utilizada en la programación funcional para modelar y resolver problemas.

Almacenan varios elementos.

En Haskell son **homogéneas** (los elementos son del mismo tipo).



SINTAXIS

Se utilizan corchetes y los elementos son separados por comas.

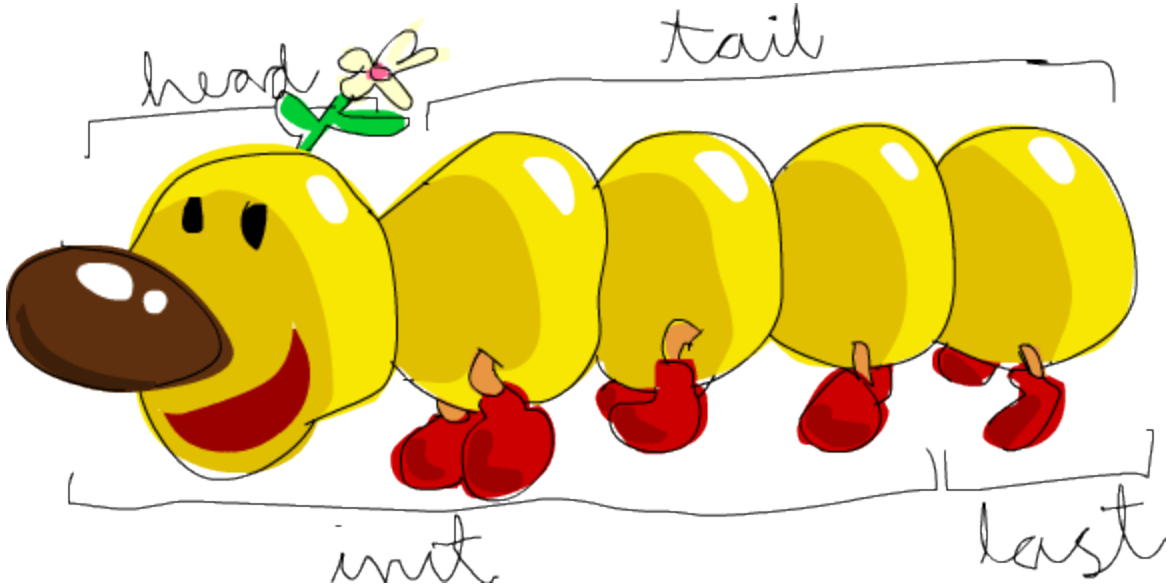
[1, 2, 3, 4, 5]

[‘a’, ’e’, ’i’, ’o’, ’u’]

[2.5, 5.0, 7.5, 10.0]

[True , True, False]

[ ]



Cabeza : primer elemento (head).

Cola : la lista sin el primer elemento (tail)

DEFINICIÓN

La lista vacía es una lista

- [ ]

Un elemento concatenado con una lista es una lista (**cons**)

- (x:xs)

¿SON LISTAS?

[ ]

Si

[1 2 3 4 5 6 7]

No

(1,2,3,4,5,6,7)

No

[1,2,3,4,5,6,7]

Si

1: 2: 3: 4: 5: 6: 7:[ ]

Si, **consing**

[1, '2', 3, 4, 5, 6, 7]

No

[[1],[2,3],[4,5,6],[7]]

Si

"1234567"

Si

STRING

Las cadenas son listas de caracteres

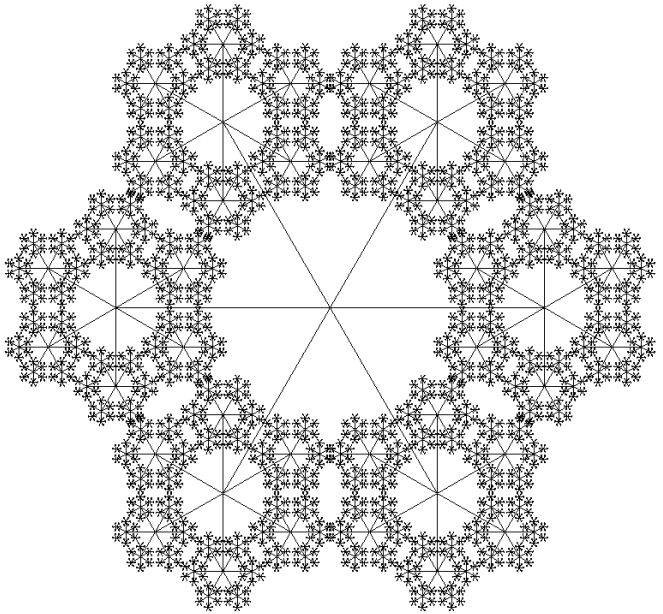
Se pueden usar todas las funciones definidas para listas sobre cadenas

TUPLAS VS. LISTAS

Las tuplas tienen un número fijo de elementos, tamaño **estático**, mientras que en las listas puedes seguir agregando elementos, tamaño **dinámico**.

Los elementos de una lista tienen que ser todos del mismo tipo, **homogéneas**, mientras que en las tuplas puedes almacenar elementos de diferentes tipos, **heterogéneas**.

FUNCIONES RECURSIVAS

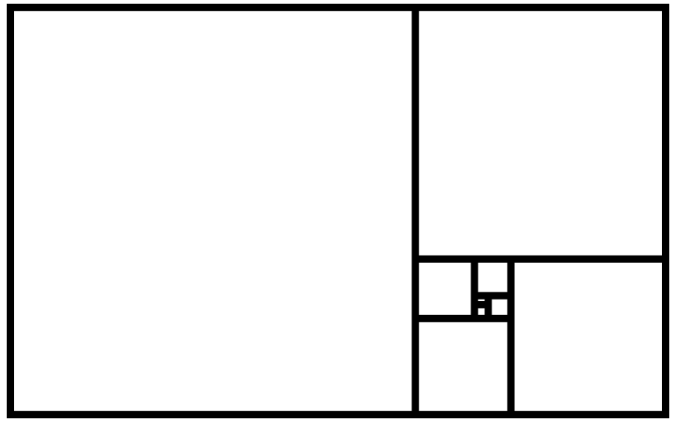


Son aquellas funciones que se definen a partir de ellas mismas.

Funciones que en su cuerpo se llaman a si mismas.

Funciones que necesitan de si mismas para ser definidas.

$$n!$$

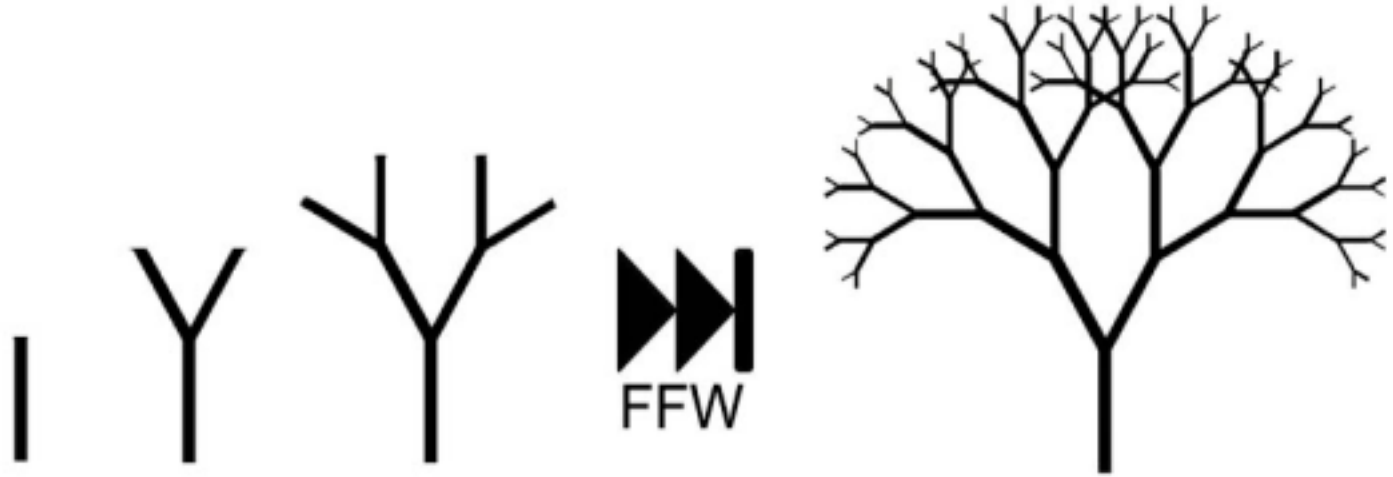


FUNCIONES RECURSIVAS

Se busca simplificar un problema hasta un caso que sea sencillo de resolver (caso base).

A partir de éste resolver los casos mas complejos.

Con cada llamada recursiva se va reduciendo el problema hasta llegar al caso base.



CASO BASE

Es el caso mas sencillo del problema.

Se le conoce también como cláusula de escape.

Este caso es el que va a hacer que ejecución termine y no continúe infinitamente.

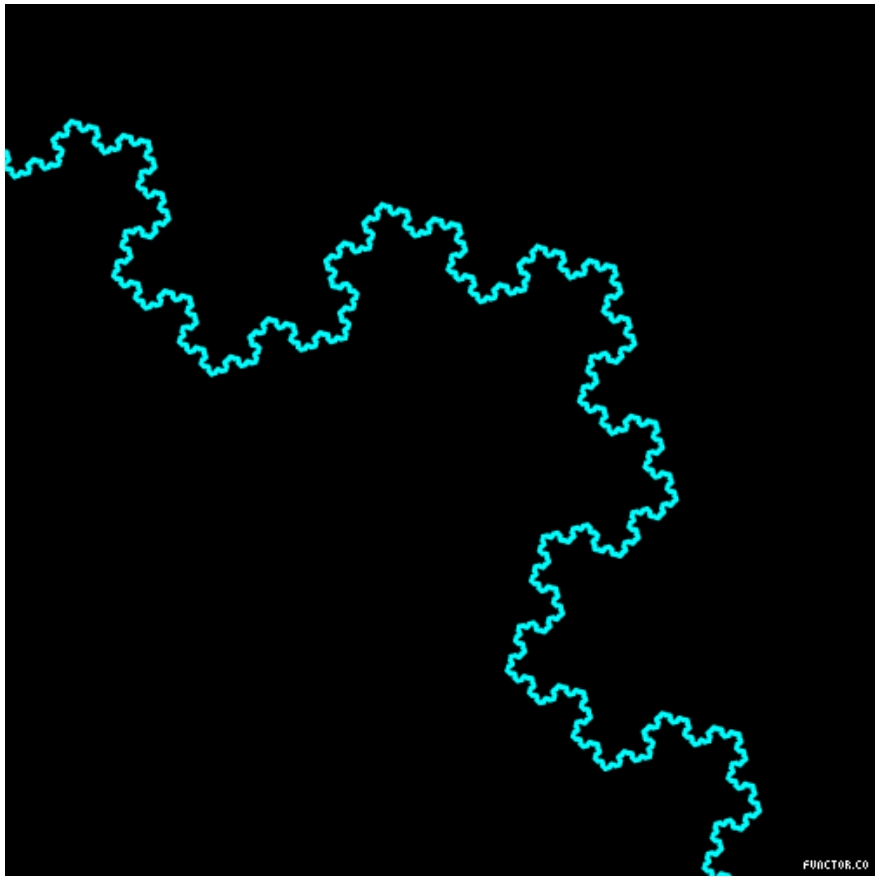
No es necesariamente único.

Se trata de reducir todos los demás casos a este.

LLAMADA RECURSIVA

Es cuando en el cuerpo de una función se llama a si misma.

Con cada llamada se tiene que reducir el problema para que se llegue al caso base y la ejecución termine.



EJEMPLOS

Factorial

Par

Impar

Car

Cdr

Ultimo

Toma

Deja

Concatenar 2 listas

Merge

Suma