Inmutabilidad y Estado

Hernan Quintana (hquintan@ulima.edu.pe)

Inmutabilidad

- Una variable se considera inmutable cuando no puede cambiar su valor.

```
int num = 10;
num = 20;
```

num es un variable mutable

En Haskell

```
x = 20
x = 10
```

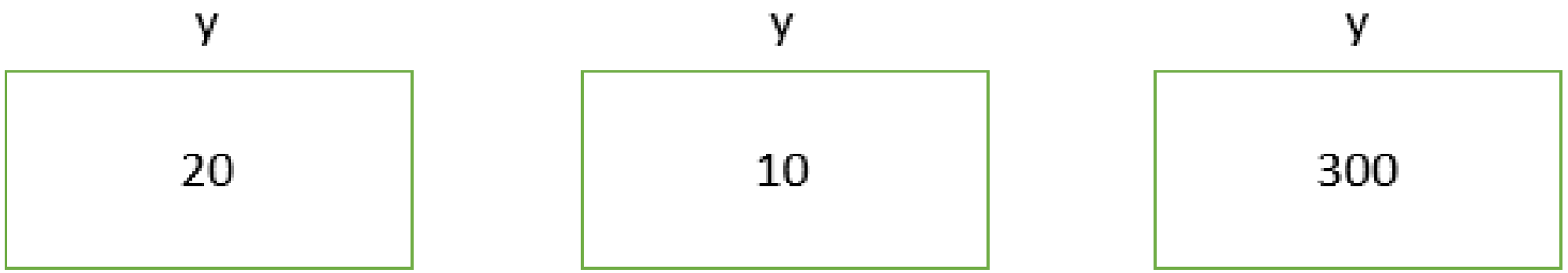
definiciones.hs

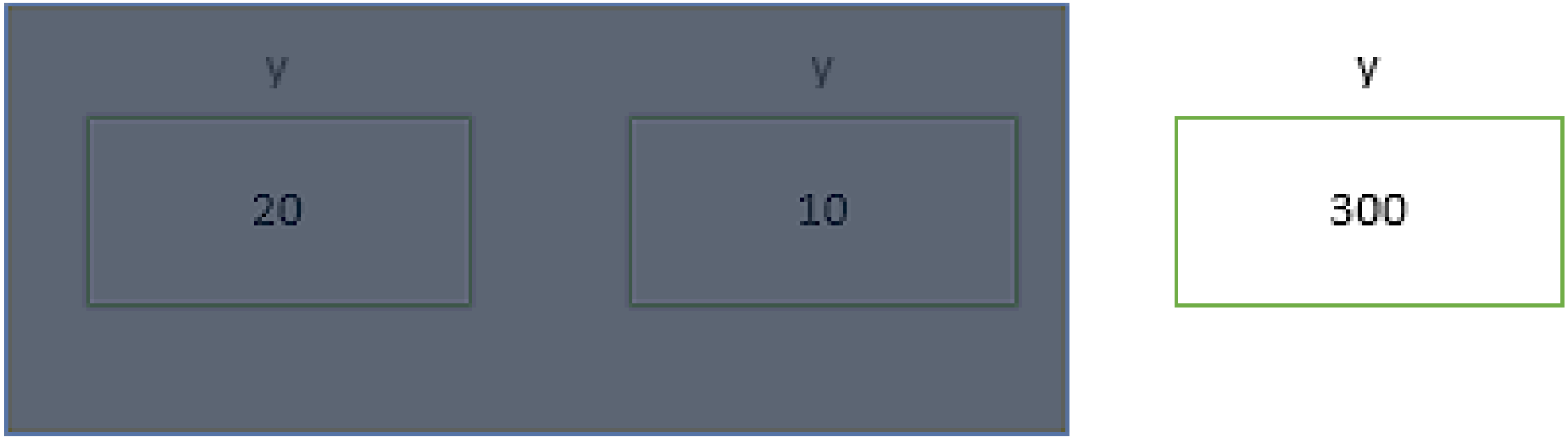
ERROR. En Haskell todas las variables son **inmutables**!

Pero si lo ponemos en el shell interactivo:

```
ghci> y = 20
ghci> y = 10
```

- **No** hay error... por qué?


y
20
y
10
y
300


y
20
y
10
y
300

- Se crean nuevas versiones de la variables, nunca se reusa la misma.
- La última siempre **oculta** a las otras, lo que no significa que no existan.

Beneficios

- Ausencia de **efectos secundarios** no deseados. Un efecto secundario es una consecuencia en el programa.
- Protección contra errores de nulos.
- Mayor desacoplamiento entre componentes.
- Ayuda a la programación distribuida (paralela).

Estado

- Son los valores de las propiedades de un componente, en un momento dado del tiempo.

```
PERSONA:
- Edad: 20
- Peso: 70 kg
- Estatura: 1.70 m
```

```
PERSONA:
- Edad: 21
- Peso: 70 kg
- Estatura: 1.70 m
```

Estado de un Programa / Componente

- Las variables declaradas en un programa son sus propiedades.
- En un determinado momento, la variables tienen determinado valor. Si alguna cambia, se dice que el **estado del programa/componente ha cambiado**.

```python
def main():
    x = 10
    nombre = "Pedro"

main()
```

```haskell
distancia x1 y1 x2 y2 =
    sqrt (
        cuadrado (x2 - x1)
        +
        cuadrado (y2 - y1)
    )
```

Cuál es el estado de estos programas?

Es importante que el estado de un programa se mantenga consistente y predecible. Caso contrario, la probabilidad de errores se incrementará,

Funciones impuras

- **Función impura:** Función que durante su ejecución *impacta* en el estado general del programa.

```
num = 0

def sq(x):
    global num
    num = x
    return x * x
```

- **Función impura:** Función que *no* devuelve un valor.

```
public void factorial(int a){
    int i;
    f=1;
    for(i=1;i<=a;i++) {
        f=f*i;
    }
    System.out.println(f);
}
```

En resumen, una función impura es un componente que tiene **efectos secundarios** fuera del contexto del mismo componente.

En Haskell, todas las funciones son **puras**.

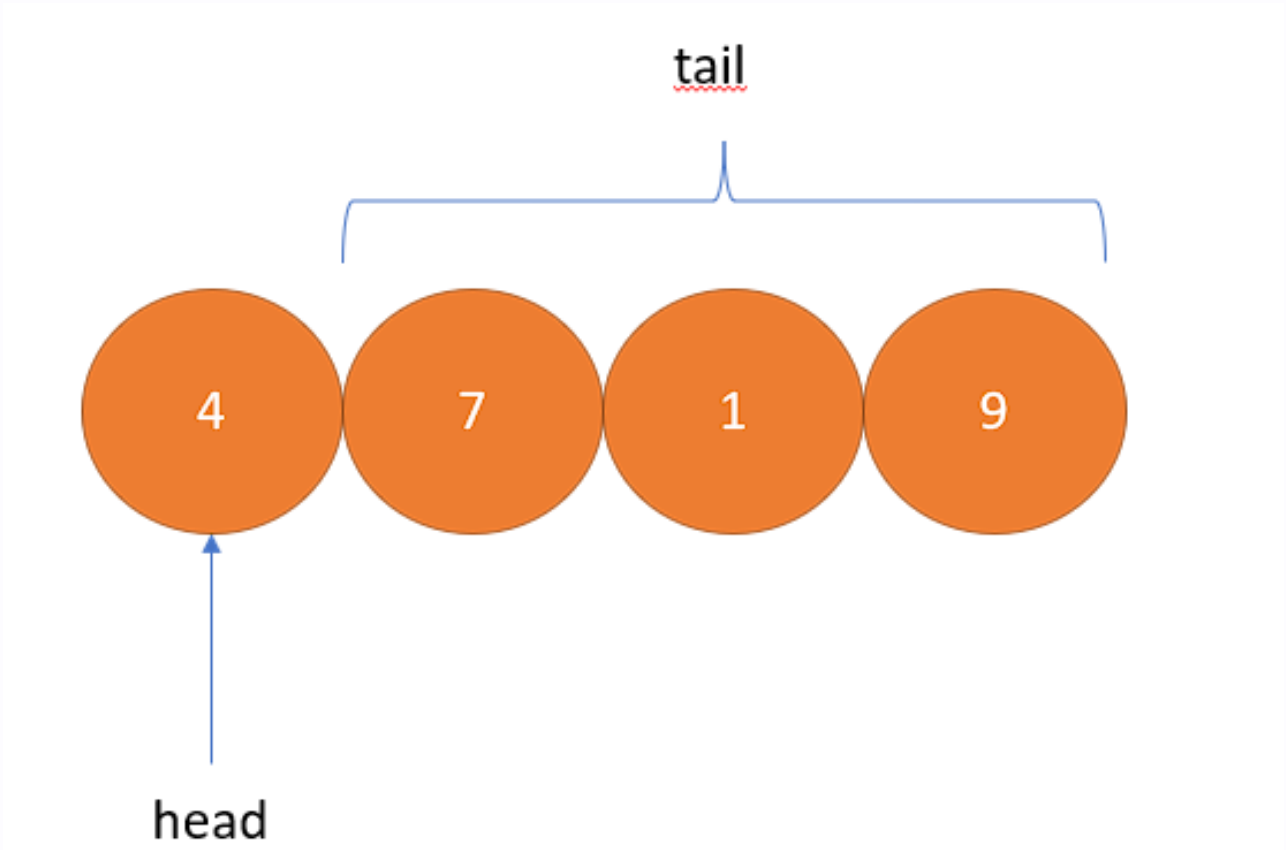
```
distancia x1 y1 x2 y2 =
    sqrt (
        cuadrado (x2 - x1)
        +
        cuadrado (y2 - y1)
    )
```

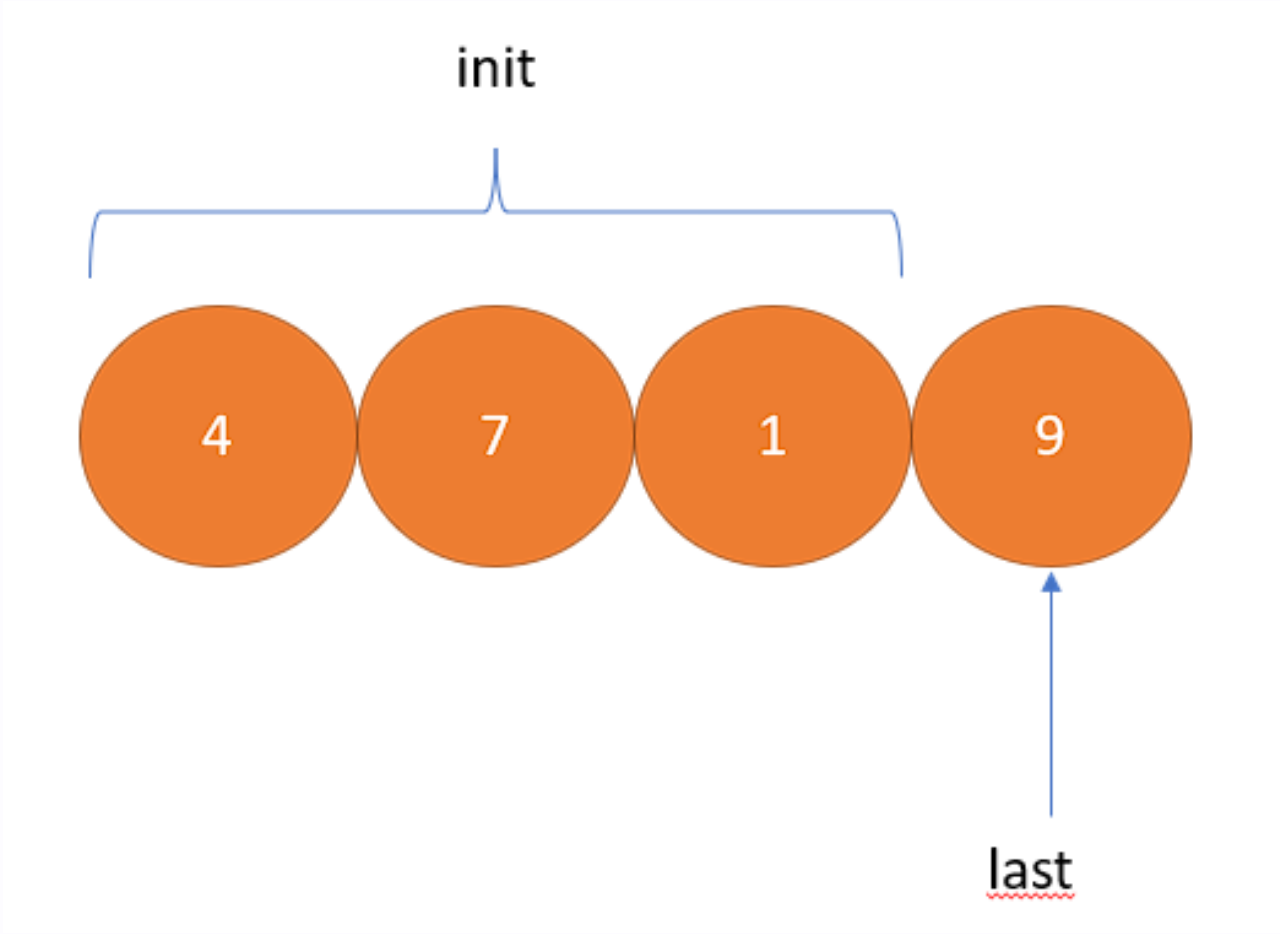
Haskell al manejar variables **inmutables**, x1, x2, y1 e y2 son copias y no las variables originales, por lo que no se tienen efectos secudnarios.

Listas

- Es una estructura de datos que nos permite almacenar varios valores del **mismo tipo**.

```
lista :: [Int] = [1, 2]
```


tail
4
7
1
9
head


init
4
7
1
9
last

Funciones básicas para trabajo con listas

Nombre	Descripción
length	longitud de una lista
null	determina si una lista contiene elemento o está vacía.
head	Devuelve el primer elemento de una lista.
tail	Devuelve una nueva lista sin el primer elemento.

continuación

Nombre	Descripción
last	Devuelve el último elemento de una lista.
init	Devuelve la lista sin el último elemento.
++	Concatena dos listas.
reverse	Devuelve los elementos de la lista en orden inverso.

Listas de más dimensiones.

- Se podrían manejar listas de más de una dimensión.
- Ejm: Definir una estructura de datos que nos permita almacenar 3 jugadas de la tinka (lotería).
  - 14, 34, 32, 40, 6, 24
  - 40, 27, 15, 12, 1,22
  - 33, 20, 11, 2, 36, 27

```
jugadasTinka :: [[Int]] = [
    [14, 34, 32, 40, 6, 24],
    [40, 27, 15, 12, 1,22],
    [33, 20, 11, 2, 36, 27]
]
```

Más funciones de listas

Nombre	Descripción
concat	Recibe una lista de listas y las concatena en una sola.