# *Programación Orientada a Objetos*

*Clase 7 - Introducción al paradigma funcional*

*PARADIGMAS DE PROGRAMACION* *UTN - La Plata*

## Temario

1) Características del Paradigma
2) Definición de funciones
3) Evaluación de funciones
4) Ejemplos
5) Actividad 1

## 1) Características del Paradigma

- Un programa, en este paradigma, es un conjunto de definiciones de ***funciones o ecuaciones matemáticas*****,** cuya evaluación permite obtener el resultado de una consulta.

- El sistema operativo ***evalúa las funciones y devuelve un resultado***, a modo de ‘calculadora’.

- La programación funcional se encuadra dentro del ***paradigma declarativo***: dice qué hacer pero no cómo hacerlo, ni en qué orden se ejecutan las acciones.

## 1) Características del Paradigma

- Se basa en el ***concepto algebraico de función:*** para un argumento dado existe solo un resultado posible, sin importar el orden en que se evalúen las expresiones matemáticas que compongan el programa, siempre se obtendrá el mismo resultado.

- Las variables no se usan como celdas de memoria cuyo contenido puede variar durante la ejecución del programa, sino como ***variables matemáticas*** cuyo valor se liga al argumento de la función y no cambia.

## 1) Características del Paradigma

- Estas dos características son la base de una propiedad especial: ***la transparencia referencial***, la cual asegura que no habrá efectos laterales que modifiquen el valor final resultante.

- Siempre que se evalúe la función en un argumento dado, el resultado será idéntico.

- En el Paradigma Imperativo se pueden tener efectos laterales.

## 2) Definición de funciones

- Se debe definir **un *script o ambiente*** en el que se detalla:
  - Tipo de la función (dominio y codominio)
  - La ecuación que la describe

cuad :: num -> num  tipo de la función

cuad(x):= x*x  definición o ecuación

nombre  argumento  valor

La definición introduce un binding entre su nombre y su valor

## 2) Definición de funciones

- Tipos de datos
  - Standard: num, bool, char, α
  - Derivado
    - Par ordenado: (num, char), (num, num), etc.
    - Lista:  [ ] , [X], [X:Xs], [Xs: Xss]
    - Función
- No existen las estructuras de control.
  - Se pueden incluir condiciones en las funciones
  - Para repetir se usa recursión.

## 2) Definición de funciones: formas de definir funciones

- Por ecuaciones simples

```
doble(x):= 2* x
```

- Por análisis de casos o Guardas

```
min(x,y):= x, if x < y
        := y, otherwise
```

- Con definiciones locales

```
potCuarta(x):= cuad(x)*cuad(x)
          where cuad(x):=x*x
```

## 2) Definición de funciones: formas de definir funciones

- Por patrones

  and(x,true):=x

  and(x,false):=false

- Recursivas

  fact(0):=1

  fact(x):= x* fact(x-1)

- De alto orden(composición de funciones)

usando map, filter, take, take while (se ven con listas)

## 2) Definición de funciones: formas de definir funciones

- Se reduce a la forma más ***simple o forma normal (canónica)***
- La reducción consiste en sustituir la función por su definición y simplificar.
- No importa el orden de reducción, siempre debe dar el mismo resultado:
  - Normal: primero se evalúa la función y después se simplifican los argumentos
  - Aplicativo: primero se simplifican los argumentos y luego se evalúa la función

## 3) Evaluación de funciones

Orden normal

?cuad(5 + 4)
(5+4)* (5+4)
9 * (5+4)
9 * 9
81

Orden aplicativo

?cuad(5 + 4)
cuad(9)
9 * 9
81

## 4) Ejemplos

- Definir y dar el tipo de una función que:

  a) Eleve a la cuarta un número

cuarta::

cuarta(x):=

O se puede definir así:

cuarta(x):=

## 4) Ejemplos

- Definir y dar el tipo de una función que:

  a) Eleve a la cuarta un número

```
cuarta:: num → num
cuarta(x):= x*x*x*x
```

O se puede definir así:

```
cuarta(x):=cuad(x)*cuad(x)
where cuad(x):=x*x
```

## 4) Ejemplos

- Definir y dar el tipo de una función que:

  b) Calcule el máximo valor entre dos números y retorne 0 si son iguales.

  max::

  max(x,y):=

## 4) Ejemplos

- Definir y dar el tipo de una función que:

  b) Calcule el máximo valor entre dos números y retorne 0 si son iguales.

  max::(num,num) → num

  max(x,y):=x, if x>y

  := y, if y<x

  := 0, otherwise

## 4) Ejemplos

- Definir y dar el tipo de la función:

  c) Cinco, que dado cualquier valor retorna 5

  cinco::

  cinco(x):=

## 4) Ejemplos

- Definir y dar el tipo de la función:

  c) Cinco, que dado cualquier valor retorna 5

  cinco::α → num

  cinco(x):=5

## 4) Ejemplos

- Definir y dar el tipo de la función:

  d) Signo, que indica el signo de su argumento o 0

signo::

signo(x):=

## 4) Ejemplos

- Definir y dar el tipo de la función:

  d) Signo, que indica el signo de su argumento o 0

  signo::num → num

  signo(x):=1, if x>0

  := 0, if x=0

  := -1, otherwise

## 4) Ejemplos

- Definir y dar el tipo de la función:

  e) xor, que devuelve el or exclusivo de sus argumentos

```
xor::

xor(x,y):=
```

## 4) Ejemplos

- Definir y dar el tipo de la función:

  e) xor, que devuelve el or exclusivo de sus argumentos

  xor::bool,bool → bool

  xor(x,y):=False,  if x=y

  :=True,  otherwise

## 5) Actividad 1

- Definir y dar el tipo de una función que:

  1) Retorne el primer valor de un par ordenado

  2) Retorne el valor absoluto de un número

  3) Retorne la suma de 2 números si ambos son positivos o un cero en cualquier otro caso