

# Introducción a los Algoritmos - 1er. cuatrimestre 2012

## Guía 2 (Primera parte): Definiciones y funciones

Docentes: Araceli Acosta, Javier Blanco, Paula Estrella, Pedro Sánchez Terraf

El objetivo de los siguientes ejercicios es introducirnos en la **programación funcional**, es decir, al desarrollo de programas como funciones.

---

### Funciones

En el práctico anterior trabajamos con expresiones aritméticas. Para completar el lenguaje de manera que nos permita escribir programas utilizaremos definiciones de funciones.

1. **Para realizar en grupo:** Leé la información sobre *función matemática* que se encuentra en **Wikipedia**. También consultá en la bibliografía de la materia, Cálculo de Programas, el concepto de función (pags. 12 y 13) para responder a las preguntas.

- a) ¿Qué es una función?
- b) ¿Qué ejemplos de función conocés?
- c) ¿Qué son el *dominio* y *codominio* de una función?
- d) ¿Para qué servirán las funciones en programación?
- e) ¿Qué es la aplicación de una función?

2. En las siguientes definiciones identificá las variables, las constantes y el nombre de la función

- a)  $f.x \doteq 5 * x$
- b)  $duplica.a \doteq a + a$
- c)  $por2.y \doteq 2 * y$
- d)  $multiplicar.zz.tt \doteq zz * tt$

3. Escribí una función que dados dos valores, calcule su promedio.

4. Tomando las definiciones del punto 2 evaluá las siguientes expresiones. Justificá cada paso utilizando la notación aprendida. Luego, controlá los resultados en **Haskell**.

- a)  $multiplicar.(f.5).2 + 1$
- b)  $por2.(duplica.3 + 5)$

5. Tomando las definiciones en el punto 2 demostrará que *duplica* y *por2* si son aplicadas al mismo valor dan siempre el mismo resultado. En otras palabras, la expresión  $duplica.x = por2.x$  es **válida**.

6. Definí las funciones simples que describimos a continuación, luego implementalas en **haskell**. Por ejemplo:

**Enunciado:**  $signo : Int \rightarrow Int$ , que dado un entero retorna su signo, de la siguiente forma: retorna 1 si  $x$  es positivo, -1 si es negativo y 0 en cualquier otro caso.

**Solución:**

$sgn : Num \rightarrow Num$	$sgn :: Int \rightarrow Int$
$sgn.x \doteq ( 0 < x \rightarrow 1$	$sgn\ x \mid 0 < x = 1$
$\quad \square x < 0 \rightarrow -1$	$\quad \mid x < 0 = -1$
$\quad \square x = 0 \rightarrow 0$	$\quad \mid x == 0 = 0$
)	

En **haskell** los conectivos para las condiciones se pueden escribir así:

$\wedge$	<b>&amp;&amp;</b>
$\vee$	<b>  </b>
$\neg$	<b>not</b>

- a)  $entre0y9 : Int \rightarrow Bool$ , que dado un entero devuelve *True* si el entero se encuentra entre 0 y 9.
- b)  $segundo3 : (Int, Int, Int) \rightarrow Int$ , que dada una terna de enteros devuelve su segundo elemento.

- c)  $ordena : (Int, Int) \rightarrow (Int, Int)$ , que dados dos enteros los ordena de menor a mayor.
- d)  $rangoPrecio : Int \rightarrow String$ , que dado un número que representa el precio de una computadora, retorne “muy barato” si el precio es menor a 2000, “demasiado caro” si el precio es mayor que 5000, “hay que verlo bien” si el precio está entre 2000 y 5000, y “esto no puede ser!” si el precio es negativo.
- e)  $absoluto : Int \rightarrow Int$ , que dado un entero retorne su valor absoluto.
- f)  $esMultiplo2 : Int \rightarrow Bool$ , que dado un entero  $n$  devuelve  $True$  si  $n$  es múltiplo de 2.

**Ayuda:** usar `mod`, el operador que devuelve el resto de la división.

- g)  $rangoPrecioParametrizado : Int \rightarrow (Int, Int) \rightarrow String$  que dado un número  $x$ , que representa el precio de un producto, y un par  $(menor, mayor)$  que represente el rango de precios que uno espera encontrar, retorne “muy barato” si  $x$  está por debajo del rango, “demasiado caro” si está por arriba del rango, “hay que verlo bien” si el precio está en el rango, y “esto no puede ser!” si  $x$  es negativo.
- h)  $mayor3 : (Int, Int, Int) \rightarrow (Bool, Bool, Bool)$ , que dada una terna de enteros devuelve una terna de valores booleanos que indica si cada uno de los enteros es mayor que 3.

Por ejemplo:  $mayor3.(1, 4, 3) = (False, True, False)$  ;  $mayor3.(5, 1984, 6) = (True, True, True)$