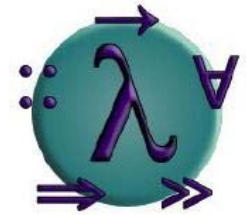


Practico N° 5 – Implementación de funciones en Haskell

Notas:

- Previo a la realización de este práctico es necesario leer el siguiente material sobre Haskell:
 - [Comenzando con funciones](#)
 - [Funciones – Segunda parte](#)



Comenzando con funciones

1. Definir una función `maximo::(Integer, Integer)->Integer` que devuelve el mayor de sus dos argumentos.
2. Definir una función `par::Integer->Bool` que indica si su argumento es par (Sugerencia: Utilizar el operador `'mod'`).
3. Definir una función `max3 ::(Integer, Integer, Integer)->Integer` que devuelve el máximo de sus argumentos.
4. Definir la función `signo:: Int->Int` que dado un número devuelve 1, 0 ó -1, en caso que el número sea positivo, cero o negativo respectivamente.
5. Definir la función `abso::Int->Int` que calcula el valor absoluto de un número.
6. Definir el predicado `bisiesto::Int->Bool` que determina si un año es bisiesto. Los años bisiestos son aquellos que son divisibles por 4 pero no por 100 a menos que también lo sean por 400. Por ejemplo, 1900 no es bisiesto pero 2000 sí lo es.
7. Tres números positivos pueden ser la medida de los lados de un triángulo si y sólo si el mayor de ellos es menor que la suma de los otros dos. Definir una función `lados_triangulo::(Float, Float, Float)-> Bool` que devuelva `True` si los tres números que se le pasan verifican esta condición, y `False` en caso contrario.
8. Definir una función `es_rectangulo::(Integer, Integer, Integer)->Bool` que devuelva `True` si los números que se le pasan pueden ser los lados de un triángulo rectángulo, y `False` en caso contrario. Sugerencia: una manera sería ordenar los tres números y verificar si el cuadrado del mayor de ellos es igual a la suma de los cuadrados de los otros dos. Sin embargo existe otra manera más fácil, utilizando (sólo una vez) la función `max_3 ::(Integer, Integer, Integer)->Integer` que devuelve el máximo de tres enteros. ¿Se te ocurre? Probar la función con las entradas (3,5,4), (5,13,12) y (7,3,5).
9. Definir una función para calcular el área de un círculo, dado su radio r . Usar "pi" que Haskell lo entiende !!!

Ejercicios complementarios – 2ª parte funciones

10. Considérese la siguiente función que calcula el número de raíces diferentes de una ecuación cuadrática de la forma $ax^2 + bx + c = 0$.
11. Considere la siguiente definición de la función `f::(Integer,Integer)->Integer`:
 $f(0,0)=1$, $f(0,1)=2$, $f(0,y)=y+2$ si $y>1$, $f(x+1,0)=1$, $f(x+1,y+1)=f(x,f(x+1,y))$
 - a) De una definición en Haskell para la función `f`.
 - b) Muestre la evaluación de la aplicación `f(3,2)`.

12. Define un operador (|||) que devuelva True si y solo si tan solo uno de sus argumentos es True. Por ejemplo:
13. Define el operador anterior con tan solo una ecuación. (AYUDA: Puedes usar la función predefinida not).
14. Define tres funciones (max2, max3 y max4) para calcular el máximo de dos, tres y cuatro enteros. (AYUDA: usa max2 para definir max3 y max4).
15. Define una función tresDiferentes que devuelva True si y solo si sus tres argumentos enteros son distintos.
16. Define una función cuatroIguales que devuelva True si y solo si sus cuatro argumentos enteros son todos iguales.
17. Define una función media3 que devuelva la media de los tres reales que toma como argumento. Usando esta función, define la función cuántos SobreMedia3 que tome tres reales y devuelva cuántos de estos son estrictamente mayores a su media.
18. Escribe una función divideA ::Integer->Integer->Bool , de modo que divideA x y sea True si y es un divisor de x.
19. Sabiendo que el máximo de dos números se puede calcular según la siguiente expresión:
$$\text{máximo}(x, y) = \frac{(x + y) + |x - y|}{2}$$
 - a) Escribe una función máximo en Haskell que tome dos enteros y devuelva el mayor.
 - b) ¿Qué expresión escribirías en Haskell para calcular el máximo de 10 y 20 usando la función anterior?
 - c) ¿Y el máximo de 10*3 y 40?
20. Usando la función que has definido en el apartado anterior, define una función en Haskell que devuelva el mayor de tres enteros. Escribe otra que devuelva el mayor de cuatro.
21. Escribe una función entre0y9 en Haskell que tome un entero y devuelva True si está entre 0 y 9 o False en caso contrario.
22. Escribe una función esMúltiploDe3 en Haskell que tome un entero y devuelva True si éste es múltiplo de 3 o False en caso contrario.
23. Escribe una función descomponer::Integer->(Integer, Integer, Integer) que a partir de una cantidad de segundos, devuelva las horas, minutos y segundos equivalentes. Por ejemplo: descomponer 7390.
Ya que 7390 segundos son dos horas, tres minutos y diez segundos. Da dos versiones, una usando where y otra usando let in.
24. Define una función incTupla3 que incremente todos los elementos de una tupla de tres enteros.
25. Escribe una función ordena3 que tome tres números enteros y devuelva una terna con los números ordenados en orden creciente.
26. Escribe una función esCapicúa que determine si un número positivo de exactamente cuatro cifras es capicúa o no. Tener en cuenta este ERROR : número de cifras incorrecto.
27. Escribe una función sumaCifras que calcule la suma de las cifras de un número natural, independientemente de su número de cifras.