

Tema 2: Introducción a la programación con Haskell

Programación declarativa (2009–10)

José A. Alonso Jiménez

Grupo de Lógica Computacional
Departamento de Ciencias de la Computación e I.A.
Universidad de Sevilla

IM Tema 2: Introducción a la programación con Haskell

Tema 2: Introducción a la programación con Haskell

1. El sistema GHC
2. Iniciación a GHC
 - Inicio de sesión con GHCi
 - Cálculo aritmético
 - Cálculo con listas
 - Cálculos con errores
3. Aplicación de funciones
4. Guiones Haskell
 - El primer guión Haskell
 - Nombres de funciones
 - La regla del sangrado
 - Comentarios en Haskell

Tema 2: Introducción a la programación con Haskell

1. El sistema GHC
2. Iniciación a GHC
3. Aplicación de funciones
4. Guiones Haskell

El sistema GHC

- ▶ Los programas funcionales pueden evaluarse manualmente (como en el tema anterior).
- ▶ Los **lenguajes funcionales** evalúan automáticamente los programas funcionales.
- ▶ **Haskell** es un lenguaje funcional.
- ▶ **GHC** (Glasgow Haskell Compiler) es el intérprete de Haskell que usaremos en el curso.

Tema 2: Introducción a la programación con Haskell

1. El sistema GHC

2. Iniciación a GHC

Inicio de sesión con GHCi

Cálculo aritmético

Cálculo con listas

Cálculos con errores

3. Aplicación de funciones

4. Guiones Haskell

Inicio de sesión

► Inicio mediante ghci

```
I1M> ghci
GHCi, version 6.10.3: http://www.haskell.org/ghc/  :? for help
Prelude>
```

► La llamada es Prelude>

- Indica que ha cargado las definiciones básicas que forman el prelude y el sistema está listo para leer una expresión, evaluarla y escribir su resultado.

Cálculo aritmético: Operaciones aritméticas

► Operaciones aritméticas en Haskell:

```
Prelude> 2+3
5
Prelude> 2-3
-1
Prelude> 2*3
6
Prelude> 7 'div' 2
3
Prelude> 2^3
8
```

Cálculo aritmético: Precedencia y asociatividad

► Precedencia:

```
Prelude> 2*10^3
2000
Prelude> 2+3*4
14
```

► Asociatividad:

```
Prelude> 2^3^4
2417851639229258349412352
Prelude> 2^(3^4)
2417851639229258349412352
Prelude> 2-3-4
-5
Prelude> (2-3)-4
-5
```

Cálculo con listas: Seleccionar y eliminar

- ▶ Seleccionar el primer elemento de una lista no vacía:
`| head [1,2,3,4,5] ~> 1`
- ▶ Eliminar el primer elemento de una lista no vacía:
`| tail [1,2,3,4,5] ~> [2,3,4,5]`
- ▶ Seleccionar el n -ésimo elemento de una lista (empezando en 0):
`| [1,2,3,4,5] !! 2 ~> 3`
- ▶ Seleccionar los n primeros elementos de una lista:
`| take 3 [1,2,3,4,5] ~> [1,2,3]`
- ▶ Eliminar los n primeros elementos de una lista:
`| drop 3 [1,2,3,4,5] ~> [4,5]`

Cálculo con listas

- ▶ Calcular la longitud de una lista:
`| length [1,2,3,4,5] ~> 5`
- ▶ Calcular la suma de una lista de números:
`| sum [1,2,3,4,5] ~> 15`
- ▶ Calcular el producto de una lista de números:
`| product [1,2,3,4,5] ~> 120`
- ▶ Concatenar dos listas:
`| [1,2,3] ++ [4,5] ~> [1,2,3,4,5]`
- ▶ Invertir una lista:
`| reverse [1,2,3,4,5] ~> [5,4,3,2,1]`

Ejemplos de cálculos con errores

```
Prelude> 1 `div` 0
*** Exception: divide by zero
Prelude> head []
*** Exception: Prelude.head: empty list
Prelude> tail []
*** Exception: Prelude.tail: empty list
Prelude> [2,3] !! 5
*** Exception: Prelude.(!!): index too large
```

Tema 2: Introducción a la programación con Haskell

1. El sistema GHC
2. Iniciación a GHC
3. Aplicación de funciones
4. Guiones Haskell

Aplicación de funciones en matemáticas y en Haskell

- ▶ Notación para funciones en matemáticas:
 - ▶ En matemáticas, la aplicación de funciones se representa usando paréntesis y la multiplicación usando yuxtaposición o espacios
 - ▶ Ejemplo:

$$f(a, b) + cd$$
 representa la suma del valor de f aplicado a a y b más el producto de c por d .
- ▶ Notación para funciones en Haskell:
 - ▶ En Haskell, la aplicación de funciones se representa usando espacios y la multiplicación usando $*$.
 - ▶ Ejemplo:

$$f\ a\ b + c*d$$
 representa la suma del valor de f aplicado a a y b más el producto de c por d .

Prioridad de la aplicación de funciones

- ▶ En Haskell, la aplicación de funciones tiene mayor prioridad que los restantes operadores. Por ejemplo, la expresión Haskell $f\ a + b$ representa la expresión matemática $f(a) + b$.
- ▶ Ejemplos de expresiones Haskell y matemáticas:

| Matemáticas | Haskell |
|--------------|------------------|
| $f(x)$ | $f\ x$ |
| $f(x, y)$ | $f\ x\ y$ |
| $f(g(x))$ | $f\ (g\ x)$ |
| $f(x, g(y))$ | $f\ x\ (g\ y)$ |
| $f(x)g(y)$ | $f\ x\ * \ g\ y$ |

Tema 2: Introducción a la programación con Haskell

1. El sistema GHC

2. Iniciación a GHC

3. Aplicación de funciones

4. Guiones Haskell

El primer guión Haskell

Nombres de funciones

La regla del sangrado

Comentarios en Haskell

Guiones Haskell

- ▶ En Haskell los usuarios pueden definir funciones.
- ▶ Las nuevas definiciones se definen en guiones, que son ficheros de textos compuestos por una sucesión de definiciones.
- ▶ Se acostumbra a identificar los guiones de Haskell mediante el sufijo `.hs`

El primer guión Haskell

- ▶ Iniciar emacs y abrir dos ventanas: C-x 2
- ▶ En la primera ventana ejecutar Haskell: M-x run-haskell
- ▶ Cambiar a la otra ventana: C-x o
- ▶ Iniciar el guión: C-x C-f ejemplo.hs
- ▶ Escribir en el guión las siguientes definiciones

```
doble x      = x+x
cuadruple x = doble (doble x)
```

- ▶ Grabar el guión: C-x C-s
- ▶ Cargar el guión en Haskell: C-c C-l
- ▶ Evaluar ejemplos:

```
*Main> cuadruple 10
40
*Main> take (doble 2) [1,2,3,4,5,6]
[1,2,3,4]
```

17 / 22

El primer guión Haskell

- ▶ Volver al guión: C-x o
- ▶ Añadir al guión las siguientes definiciones:

```
factorial n = product [1..n]
media ns    = sum ns `div` length ns
```

- ▶ Grabar el guión: C-x s
- ▶ Cargar el guión en Haskell: C-c C-l
- ▶ Evaluar ejemplos:

```
*Main> factorial (doble 2)
24
*Main> doble (media [1,5,3])
6
```

18 / 22

Nombres de funciones

- ▶ Los nombres de funciones tienen que empezar por una letra en minúscula. Por ejemplo,
 - ▶ `sumaCuadrado`, `suma_cuadrado`, `suma'`
- ▶ Las palabras reservadas de Haskell no pueden usarse en los nombres de funciones. Algunas palabras reservadas son

| | | | | | | |
|-------------------|---------------------|----------------------|----------------------|-----------------------|---------------------|-----------------------|
| <code>case</code> | <code>class</code> | <code>data</code> | <code>default</code> | <code>deriving</code> | <code>do</code> | <code>else</code> |
| <code>if</code> | <code>import</code> | <code>in</code> | <code>infix</code> | <code>infixl</code> | <code>infixr</code> | <code>instance</code> |
| <code>let</code> | <code>module</code> | <code>newtype</code> | <code>of</code> | <code>then</code> | <code>type</code> | <code>where</code> |
- ▶ Se acostumbra escribir los argumentos que son listas usando `s` como sufijo de su nombre. Por ejemplo,
 - ▶ `ns` representa una lista de números,
 - ▶ `xs` representa una lista de elementos,
 - ▶ `css` representa una lista de listas de caracteres.

La regla del sangrado

- ▶ En Haskell la disposición del texto del programa (el **sangrado**) delimita las definiciones mediante la siguiente regla:

Una definición acaba con el primer trozo de código con un margen izquierdo menor o igual que el del comienzo de la definición actual.

- ▶ Ejemplo:

```
a = b + c
  where
    b = 1
    c = 2
d = a * 2
```

- ▶ Consejos:
 - ▶ Comenzar las definiciones de las funciones en la primera columna.
 - ▶ Usar el tabulador en emacs para determinar el sangrado en las definiciones.

Comentarios en Haskell

- ▶ En los guiones Haskell pueden incluirse comentarios.
- ▶ Un **comentario simple** comienza con `--` y se extiende hasta el final de la línea.
- ▶ Ejemplo de comentario simple:

```
-- (factorial n) es el factorial del número n.  
factorial n = product [1..n]
```

- ▶ Un **comentario anidado** comienza con `{-` y termina en `-}`
- ▶ Ejemplo de comentario anidado:

```
{- (factorial n) es el factorial del número n.  
   Por ejemplo, factorial 3 == 6 -}  
factorial n = product [1..n]
```

Bibliografía

1. R. Bird. *Introducción a la programación funcional con Haskell*. Prentice Hall, 2000.
 - ▶ Cap. 1: Conceptos fundamentales.
2. G. Hutton *Programming in Haskell*. Cambridge University Press, 2007.
 - ▶ Cap. 2: First steps.
3. B. O'Sullivan, D. Stewart y J. Goerzen *Real World Haskell*. O'Reilly, 2008.
 - ▶ Cap. 1: Getting Started.
4. B. Pope y A. van IJzendoorn *A Tour of the Haskell Prelude (basic functions)*
5. B.C. Ruiz, F. Gutiérrez, P. Guerrero y J.E. Gallardo. *Razonando con Haskell*. Thompson, 2004.
 - ▶ Cap. 2: Introducción a Haskell.
6. S. Thompson. *Haskell: The Craft of Functional Programming*, Second Edition. Addison-Wesley, 1999.
 - ▶ Cap. 2: Getting started with Haskell and Hugs.