Iniciando Haskell

1 Instalación

El principal compilador de Haskell es el GHC (Glorious Haskell Compiler).
Para instalarlo, ver `https://www.haskell.org/ghc/download.html`
GHC tiene 2 componentes principales:

- **ghc:** es un compilador de Haskell, mientras que
- **ghci:** es un intérprete de Haskell.

Aconsejamos usar el intérprete para comenzar a probar los programas en Haskell, ya que éste les permitirá experimentar y evaluar expresiones directamente.

2 ghci

El intérprete de Haskell **ghci** se ejecuta tecleando ghci desde una terminal.

```
$ ghci
GHCi, version 8.y.z: https://www.haskell.org/ghc/  :? for help
Prelude>
```

Las funciones definidas en el módulo `Prelude.hs` y algunas librerías estándar son cargadas al ejecutar el intérprete. Con lo cual podríamos usarlo sin definir un programa Haskell, para probar las funciones definidas en allí. Por ejemplo:

```
Prelude> 1+2
3
Prelude> let x = 42 in x / 9
4.666666666666667
Prelude> map succ [1,2,3]
[2,3,4]
Prelude> product [1..7]
5040
Prelude>
```

Podemos pasar nombres de archivos a cargar por la línea de comandos:

```
$ ghci Test.hs
GHCi, version 8.y.z: https://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Test          ( Test.hs, interpreted )
Ok, one module loaded.
*Test>
```

Para ver la lista de comandos que podemos usar en el intérprete, junto con una descripción de lo que hace cada uno, escribimos `:help` (o `:h`). A continuación mostramos sólo los comandos más usados.

Para cargar un programa en Haskell en ghci usamos `:load` (o `:l`), por ejemplo:

```
Prelude> :l prog.hs
```

Para cargar el último archivo `:reload` (o `:r`). Esto es especialmente útil luego de editar el último archivo que cargamos.

El comando `:type` (o `:t`) es utilizado para ver el tipo de una expresión. Por ejemplo:

```
Prelude> :t map
map :: (a -> b) -> [a] -> [b]
Prelude> :t map (+1)
map (+1) :: Num b => [b] -> [b]
```

El comando `:info` (o `:i`) nos da información, tanto de funciones como de tipos o clases de tipo:

```
Prelude> :i map
map :: (a -> b) -> [a] -> [b]   -- Defined in ‘GHC.Base’

Prelude> :i Bool
data Bool = False | True  -- Defined in ‘GHC.Types’
instance Eq Bool -- Defined in ‘GHC.Classes’
instance Ord Bool -- Defined in ‘GHC.Classes’
instance Enum Bool -- Defined in ‘GHC.Enum’
instance Show Bool -- Defined in ‘GHC.Show’
instance Read Bool -- Defined in ‘GHC.Read’
instance Bounded Bool -- Defined in ‘GHC.Enum’

Prelude> :i Num
class Num a where
  (+) :: a -> a -> a
  (-) :: a -> a -> a
  (*) :: a -> a -> a
  negate :: a -> a
  abs :: a -> a
  signum :: a -> a
  fromInteger :: Integer -> a
  {-# MINIMAL (+), (*), abs, signum, fromInteger,
      (negate | (-)) #-}
   -- Defined in ‘GHC.Num’
instance Num Word -- Defined in ‘GHC.Num’
instance Num Integer -- Defined in ‘GHC.Num’
instance Num Int -- Defined in ‘GHC.Num’
instance Num Float -- Defined in ‘GHC.Float’
instance Num Double -- Defined in ‘GHC.Float’
```