

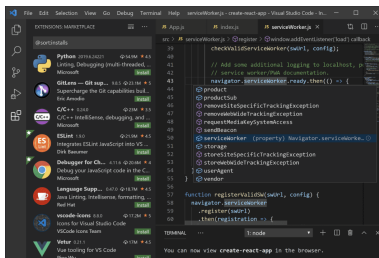
Introducción a la programación

Práctica 3: Introducción a Haskell

Configurando Vscode

VS code es una IDE (Integrated Development Environment), existen MUCHAS:

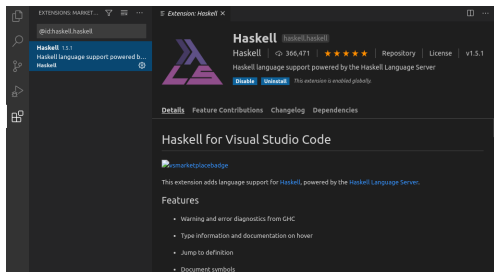
- ▶ Visual Studio (<https://visualstudio.microsoft.com/es/>)
- ▶ Eclipse (<https://www.eclipse.org/>)
- ▶ IntelliJ IDEA (<https://www.jetbrains.com/es-es/idea/>)
- ▶ **Visual Code o Visual Studio Code** (<https://code.visualstudio.com/>)
 - ▶ Es un editor de textos que se “convierte” en IDE mediante *extensions*.
 - ▶ Lo utilizaremos para programar en Haskell y Python.



Configurando Vscode

Vamos a instalar la extensión de Haskell:

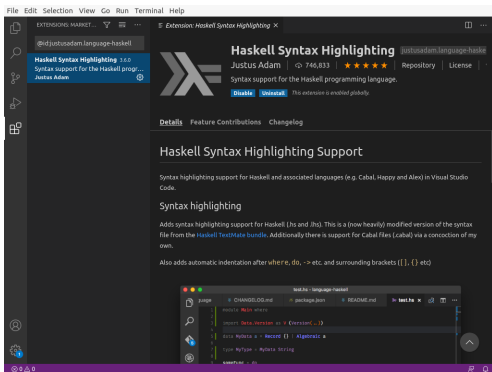
- ▶ Abrir Visual Studio Code en sus computadoras
- ▶ Abrir el buscador apretando `ctrl+P` (se abre una barra arriba)
- ▶ Buscar `ext install haskell.haskell`
- ▶ En la barra de la izquierda se abre el buscador de extensiones con una sola opción encontrada. Hacemos click y la instalamos (si no lo está).



Configurando Vscode

Ahora la extensión de Syntax Highlighting:

- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)
- ▶ Buscar `ext install justusadam.language-haskell`
- ▶ En la barra de la izquierda se abre el buscador de extensiones con una sola opción encontrada. Hacemos click y la instalamos (si no lo está).



Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/clase04/`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/clase04/`
- ▶ Abrir una Terminal Terminal > New Terminal

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/clase04/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd /Escritorio/clase04/`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/clase04/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd /Escritorio/clase04/`
- ▶ Ahora vamos a abrir el intérprete interactivo de Haskell: `ghci`

Primeros pasos en Haskell

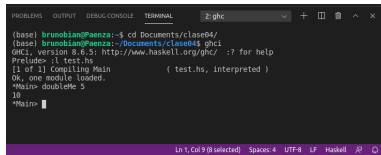
Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/clase04/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd /Escritorio/clase04/`
- ▶ Ahora vamos a abrir el intérprete interactivo de Haskell: `ghci`
- ▶ Dentro del intérprete tenemos que pedirle que cargue nuestro archivo: `:l test.hs`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/clase04/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd /Escritorio/clase04/`
- ▶ Ahora vamos a abrir el intérprete interactivo de Haskell: `ghci`
- ▶ Dentro del intérprete tenemos que pedirle que cargue nuestro archivo: `:l test.hs`
- ▶ Ahora nuestra función ya existe y podemos usarla `doubleMe 5`



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL z: ghc
(base) bruno@brunobian@paenza:~$ cd Documents/clase04/
(base) bruno@brunobian@paenza:~/Documents/clase04$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/ :? for help
Prelude> :l test.hs
[1 of 1] Compiling Main             ( test.hs, interpreted )
Ok, one module loaded.
*Main> doubleMe 5
10
*Main> 
```

Ya tenemos todo lo necesario para la Guía 2
Ahora a programar!!

Ejercicio 1

- a) Implementar la función parcial $f :: \text{Integer} \rightarrow \text{Integer}$ definida por extensión de la siguiente manera:

$$f(1) = 8, f(4) = 131, f(16) = 16$$

cuya especificación es la siguiente:

```
problema f (n:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere:  $\{n = 1 \vee n = 4 \vee n = 16\}$   
  asegura:  $\{(n = 1 \rightarrow result = 8) \wedge (n = 4 \rightarrow result = 131) \wedge (n =$   
     $16 \rightarrow result = 16)\}$   
}
```

- b) Análogamente, especificar e implementar la función parcial $g :: \text{Integer} \rightarrow \text{Integer}$

$$g(8) = 16, g(16) = 4, g(131) = 1$$

- c) A partir de las funciones definidas en los ítems 1 y 2, implementar las funciones parciales $h = f \circ g$ y $k = g \circ f$