

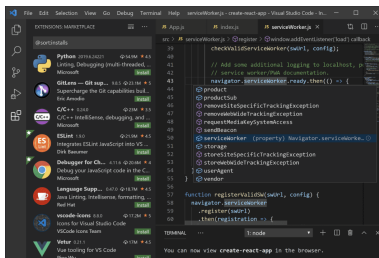
Introducción a la programación

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

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)

Configurando Vscode

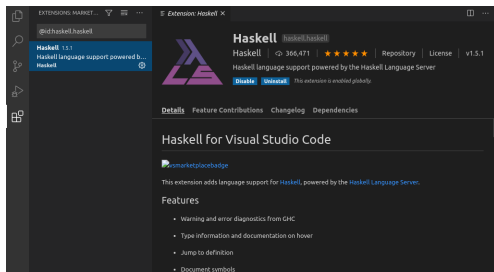
Vamos a instalar la extensión de Haskell:

- ▶ Abrir Visual Studio Code en sus computadoras
- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)
- ▶ Buscar `ext install haskell.haskell`

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: (si no funciona no es tan grave)

- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)

Configurando Vscode

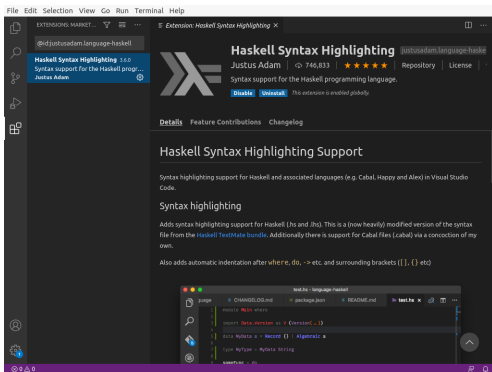
Ahora la extensión de Syntax Highlighting: (si no funciona no es tan grave)

- ▶ Abrir el buscador apretando ctrl+P (se abre una barra arriba)
- ▶ Buscar `ext install justusadam.language-haskell`

Configurando Vscode

Ahora la extensión de Syntax Highlighting: (si no funciona no es tan grave)

- ▶ 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`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función? EN LA MATERIA ES OBLIGATORIO DEFINIRLO!

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función? EN LA MATERIA ES OBLIGATORIO DEFINIRLO!
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en Escritorio/guia3/

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función? EN LA MATERIA ES OBLIGATORIO DEFINIRLO!
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en Escritorio/guia3/
- ▶ 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`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función? EN LA MATERIA ES OBLIGATORIO DEFINIRLO!
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función? EN LA MATERIA ES OBLIGATORIO DEFINIRLO!
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`
- ▶ Ahora vamos a abrir el intérprete interactivo de Haskell: `ghci`

Primeros pasos en Haskell

Hagamos nuestro primer programa:

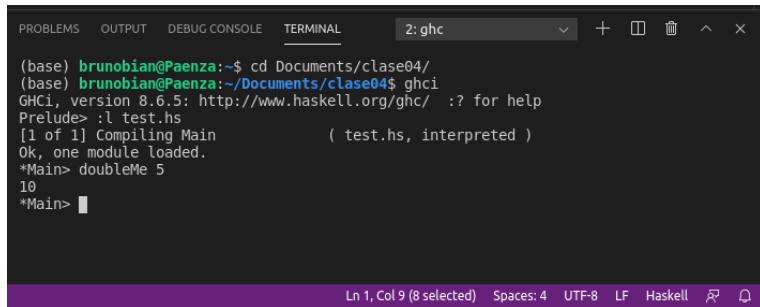
- ▶ Abrir un archivo nuevo File > New File
- ▶ Definir nuestra primera función: `doubleMe x = x + x`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función? EN LA MATERIA ES OBLIGATORIO DEFINIRLO!
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en `Escritorio/guia3/`
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`
- ▶ 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`
 - ▶ De qué tipo son los parámetros de entrada y salida de esta función? EN LA MATERIA ES OBLIGATORIO DEFINIRLO!
- ▶ Guardar el archivo como `test.hs`
 - ▶ Es importante recordar dónde lo guardamos
 - ▶ Vamos a guardarlo en Escritorio/guia3/
- ▶ Abrir una Terminal Terminal > New Terminal
- ▶ En la terminal asegurarse que estemos en el directorio donde guardamos el archivo
 - ▶ `cd ~/Escritorio/guia3/`
- ▶ 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`

Primeros pasos en Haskell



A screenshot of a terminal window with a dark background. The window has tabs at the top: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, and a dropdown menu shows '2: ghc'. The terminal text shows a user running 'cd Documents/clase04/' and 'ghci'. It then shows 'GHCi, version 8.6.5: http://www.haskell.org/ghc/ :? for help'. The user enters ':l test.hs', and the terminal shows '[1 of 1] Compiling Main (test.hs, interpreted)' and 'Ok, one module loaded.'. The user then enters '*Main> doubleMe 5', and the terminal shows '10'. The bottom status bar is purple and shows 'Ln 1, Col 9 (8 selected) Spaces: 4 UTF-8 LF Haskell' along with some icons.

```
(base) brunobian@Paenza:~$ cd Documents/clase04/
(base) 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> 
```

Ln 1, Col 9 (8 selected) Spaces: 4 UTF-8 LF Haskell

Algunos comando útiles

- ▶ `ghci` → abrir el intérprete de haskell
- ▶ `:l (:load) file.hs` → cargar el archivo `file.hs`
- ▶ `:r (:reload)` → recargar el último archivo (siempre hacerlo después de modificar el código!)
- ▶ `:q (:quit)` → salir del interprete `ghci`
- ▶ `:t (:type) E` → conocer el tipo de una expresión `E`. Ejemplo
`:t (+)`

Ya tenemos todo lo necesario para hacer la Guía 3
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 $k = g \circ f$

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

c) **maximo3**: devuelve el máximo entre tres números enteros.

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

c) **maximo3**: devuelve el máximo entre tres números enteros.

Una posible especificación

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { res es igual a  $x$ , o a  $y$  o a  $z$  }  
  asegura: { res es mayor o igual a  $x$ , y a  $y$ , y a  $z$  }  
}
```

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

c) **maximo3**: devuelve el máximo entre tres números enteros.

Una posible especificación

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { res es igual a  $x$ , o a  $y$  o a  $z$  }  
  asegura: { res es mayor o igual a  $x$ , y a  $y$ , y a  $z$  }  
}
```

Otra forma de especificar, usando lógica

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

c) **maximo3**: devuelve el máximo entre tres números enteros.

Una posible especificación

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { res es igual a  $x$ , o a  $y$  o a  $z$  }  
  asegura: { res es mayor o igual a  $x$ , y a  $y$ , y a  $z$  }  
}
```

Otra forma de especificar, usando lógica

```
problema maximo3 (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: {(res = x)  $\vee$  (res = y)  $\vee$  (res = z)}  
  asegura: {(res  $\geq$  x)  $\wedge$  (res  $\geq$  y)  $\wedge$  (res  $\geq$  z)}  
}
```

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Esto tiene (al menos) dos interpretaciones posibles:

- ▶ Cuando hay algún número repetido no lo sumo
 - ▶ $\text{sumaDistintos}(1,1,2) = 2$
- ▶ Cuando hay algún número repetido lo sumo una sola vez
 - ▶ $\text{sumaDistintos}(1,1,2) = 3$

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Esto tiene (al menos) dos interpretaciones posibles:

- ▶ Cuando hay algún número repetido no lo sumo
 - ▶ $\text{sumaDistintos}(1,1,2) = 2$
- ▶ Cuando hay algún número repetido lo sumo una sola vez
 - ▶ $\text{sumaDistintos}(1,1,2) = 3$

Una posible especificación de la primera opción

```
problema sumaDistintos (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: { - }  
  asegura: {si los 3 parámetros son distintos entonces  $res = x + y + z$ }  
  asegura: {si 2 parámetros son iguales, res es igual al no repetido}  
  asegura: {si los 3 parámetros son iguales,  $res = 0$ }  
}
```

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- g) **sumaDistintos**: que dados tres números enteros calcule la suma sin sumar repetidos (si los hubiera).

Esto tiene (al menos) dos interpretaciones posibles:

- ▶ Cuando hay algún número repetido no lo sumo
 - ▶ $\text{sumaDistintos}(1,1,2) = 2$
- ▶ Cuando hay algún número repetido lo sumo una sola vez
 - ▶ $\text{sumaDistintos}(1,1,2) = 3$

Otra especificación de la primera opción

```
problema sumaDistintos (x,y,z:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: {(  $x \neq y$ )  $\wedge$  ( $x \neq z$ )  $\wedge$  ( $y \neq z$ ) }  $\rightarrow$  res =  $x + y + z$ }  
  asegura: {(  $x = y$ )  $\wedge$  ( $x \neq z$ )  $\wedge$  ( $y \neq z$ ) }  $\rightarrow$  res =  $z$ }  
  asegura: {(  $x \neq y$ )  $\wedge$  ( $x = z$ )  $\wedge$  ( $y \neq z$ ) }  $\rightarrow$  res =  $y$ }  
  asegura: {(  $x \neq y$ )  $\wedge$  ( $x \neq z$ )  $\wedge$  ( $y = z$ ) }  $\rightarrow$  res =  $x$ }  
  asegura: {(  $x = y$ )  $\wedge$  ( $x = z$ )  $\wedge$  ( $y = z$ ) }  $\rightarrow$  res = 0}  
}
```

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- i) **digitoUnidades:** dado un número entero, extrae su dígito de las unidades.

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- i) **digitoUnidades**: dado un número entero, extrae su dígito de las unidades.

```
problema digitoUnidades (x:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { result es el último dígito de x }  
}
```

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- j) **digitoDecenas:** dado un número entero, extrae su dígito de las decenas.

Ejercicio 2: Especificar e implementar las siguientes funciones, incluyendo su signatura.

- j) **digitoDecenas**: dado un número entero, extrae su dígito de las decenas.

```
problema digitoDecenas (x:  $\mathbb{Z}$ ) :  $\mathbb{Z}$  {  
  requiere: {True}  
  asegura: { result es el dígito de x correspondiente a las  
             decenas}  
}
```

Ejercicio 4: Especificar e implementar las siguientes funciones utilizando tuplas para representar pares, ternas de números.

- b) **todoMenor**: dadas dos tuplas $\mathbb{R} \times \mathbb{R}$, decide si es cierto que cada coordenada de la primera tupla es menor a la coordenada correspondiente de la segunda tupla.

Ejercicio 4: Especificar e implementar las siguientes funciones utilizando tuplas para representar pares, ternas de números.

- b) **todoMenor**: dadas dos tuplas $\mathbb{R} \times \mathbb{R}$, decide si es cierto que cada coordenada de la primera tupla es menor a la coordenada correspondiente de la segunda tupla.

```
problema todoMenor (t1, t2:  $\mathbb{R} \times \mathbb{R}$ ) : Bool {  
  requiere: {True}  
  asegura: { result = true  $\leftrightarrow$  la primera componente de t1 es  
             menor que la primera componente de t2, y la segunda  
             componente de t1 es menor que la segunda componente  
             de t2 }  
}
```