



- 1)  $20^{*33}$
- 2)  $2^{^18}$
- 3)  $24^{*(67-111)}$
- 4)  $2^{^(6+3)}$
- 5)  $73/5$
- 6) `div 73 5`
- 7) `mod 73 5`
- 8) `mod (-20) 3`
- 9) `div (-20) 3`
- 10) Deduzca el comportamiento de las funciones `div` y `mod`, **predefinidas\*** en Haskell
- 11) Pruebe sus propios ejemplos

Esto significa que Hugs interpreta expresiones aritméticas y computa su resultado.

Se denominan **\*predefinidos** los elementos del lenguaje que están disponibles para su uso cada vez que ejecutamos Haskell. Funciones, operadores, tipos, colecciones de objetos, palabras reservadas. Estos elementos están definidos en un archivo de Haskell especial que se denomina PRELUDE

En Haskell hay muchas funciones (u operadores) predefinidas, ejemplos `div`, `mod`. Veamos algunas otras:

Escriba y ejecute las siguientes expresiones en Hugs.

- 1) `max 7 9`
- 2) `min 2 (-8)`

En Hugs no se pueden realizar definiciones en línea de comandos, para ello se deberá utilizar un archivo `.hs`.

Pruebe ejecutar la siguiente línea en Hugs

```
y = x+1
```

El error que devuelve Hugs es debido a que se están intentando definir las variables `x` y `y` en línea de comandos.

Probemos la solución correcta: Cree un archivo en la carpeta Mis Documentos de su disco y denomínelo `primero.hs`

Copie las siguientes líneas en su archivo y guarde.

```
y = x+1  
x = 2*3
```

Ejecutar en Hugs File/open/ y luego busque primero.hs y selecciónelo.  
Escriba x en la línea de comandos y dé enter, lo mismo con y.

### Tipos simples predefinidos en Haskell

A continuación estudiaremos los principales **tipos de datos** predefinidos y las funciones asociadas a éstos.

Si bien es un concepto que no tiene una definición formal como para introducir en este nivel, nosotros manejaremos el término **tipo de datos** como el equivalente a un conjunto de elementos, que puede ser finito o infinito.

#### El tipo Bool

Los valores de este tipo representan expresiones lógicas, cuyo resultado de evaluación puede ser verdadero o falso.

En Haskell sólo hay dos constantes para este tipo {True, False} y deben escribirse exactamente así. Estas son las dos constantes que representan los dos resultados posibles de evaluar una expresión que tenga (o que pertenezca a ) el tipo Bool.

#### Funciones y operadores del tipo Bool

- ❖ && - Conjunción lógica  
    &&: BoolxBool → Bool

**Notación currificada:**  
    &&: Bool → Bool → Bool

- ❖ || - Disyunción lógica  
    ||: BoolXBool → Bool

**Notación currificada:**  
    ||: Bool → Bool → Bool

Los operadores && y || utilizan notación infija

Más adelante estudiaremos lo que significa **notación currificada** para funciones y operadores

- ❖ not - Negación lógica  
    not : Bool → Bool

El operador not utiliza notación prefija

- ❖ otherwise - Función constante que devuelve el valor True  
otherwise : Bool

Calcular en Haskell:

- 1)  $(8 > 2) \ || \ (4 > 1)$
- 2)  $(8 > 2) \ \&\& \ (4 > 1)$
- 3)  $(8 > 2) \ \&\& \ (4 < 1)$
- 4) Escribir y probar en Hugs sus propias expresiones utilizando los operadores lógicos introducidos.

### Operadores de igualdad y orden

Para todos los tipos básicos (aún los que todavía no hemos visto) están definidos los siguientes operadores binarios, las expresiones que utilizan estos operadores evalúan a algún elemento del conjunto Bool, por lo tanto se denominan expresiones booleanas.

>	Mayor que	>=	Mayor o igual que
<	Menor que	<=	Menor o igual que
==	Igual a	/=	Distinto de

Observación: el tipo de los dos argumentos (u operandos) para cualquier aplicación de los operadores anteriores debe ser el mismo. No se pueden comparar valores de tipos distintos)

Escriba y evalúe las siguientes expresiones en Hugs.

- 1)  $7 < 0$
- 2)  $8 > 2$
- 3)  $\text{div } 8 \ 4 == 2$
- 4)  $\text{div } 4 \ 3 == 0$
- 5)  $\text{mod } 8 \ 4 == 0$
- 6)  $\text{not } (\text{mod } 8 \ 4 == 7)$
- 7)  $4 /= 4$
- 8)  $\text{div } 8 \ 9 /= 20$
- 9)  $\text{mod } 50 \ 10 <= 0$
- 10)  $\text{div } 50 \ 10 >= 6$
- 11) Agregue sus propias expresiones booleanas en Hugs y evalúelas