Las operaciones aritméticas

Realiza las siguientes operaciones:

- $2 + 5$
- 49×100
- $1892 - 1472$
- *a*) $5 / 2$ *b*) $5432 / 277$ *c*) $1{,}1 + 2{,}2$
- Multiplicación con un número negativo. Paréntesis.
- *a*) $50 \times 100 - 4999$ *b*) $(50 \times 100) - 4999$
- $50 \times (100 - 4999)$
- *a*) 2^5 *b*) 32^{45}
- *a*) 12345678×87654321 *b*) $12345678 \times 87654321{,}0$
- Transforma 32^{45} en número real con **::Double**.
- *a*) $2^{0,5}$ *b*) $32{,}1^{32{,}6}$. (Operador **).
- Calcula el cociente y el resto de 2347 / 45 con los operadores **div** y **mod**. Comprueba la división.
- Realiza operaciones con la forma prefija.

Operaciones lógicas y comparaciones

Realiza las siguientes operaciones

- True $\wedge$ False. (Operador **&&**)
- True $\wedge$ True.
- False $\vee$ False (Operador **||**)
- $\neg$False. (Operador **not**)
- $\neg$(True $\wedge$ True)
- $5 = 5$ (Operador **==**)
- $5 = 12$
- $5 \neq 5$ (Operador **/=**)
- `"hola" = "hola"`
- $3 < 8$ (Operador **<**)
- $5 \leq 5$ (Operador **<=**)
- $6 <$ `"hola"` (Error)
- $34{,}8 \geq 2{,}3 \cdot 10^2$ (Operador **>=**)

Funciones predefinidas

- Aplica la función **succ** a números enteros, positivos y negativos, a números reales y a caracteres.
- Aplica la función **pred** a números enteros, positivos y negativos, a números reales y a caracteres.
- Aplica las funciones **min** y **max** a parejas de números.
- Aplica las funciones **min** y **max** a una terna de números.
- Calcula `succ 9 + max 5 4 +1`
- Componer varias funciones predefinidas. Uso de paréntesis.
- Calcula `succ 9 * 10`
- Opera con las siguientes funciones: **pi**, **sqrt**, **cos**, **cosh**, **asin**, **gcd**, **exp**, **log**.
- Opera con las siguientes funciones: **round**, **lcm** y **gcd**.

Definición de funciones

- Construye la función $f(x) = 2x$ en un archivo y cárgala en el intérprete. (Comando **:l**).
- Construye la función $g(x, y) = x^2 + y^2$ y recarga el archivo de texto. (Comando **:r**).
- Construye la función $h(x, y) = 6x + x^2 + y^2$ utilizando las funciones anteriores.
- Construye la función:

$$f(x) = \begin{cases} x & \text{si } x > 100 \\ x^2 & \text{si } x \leq 100 \end{cases}$$

  utilizando la construcción **if . . . then . . . else. . .** .
- Crea una constante llamada «respuesta» (en el archivo de texto) y asígnale el número 42.
- Define otra constante (en el intérprete) utilizando el comando **let**.
- Define una constante local utilizando la construcción **let . . . in . . .**

Listas 1

- Construye una lista numérica y asígnale un nombre.
- Una lista de caracteres y otra de booleanos.
- Las cadenas y las listas de caracteres son lo mismo.
- Intenta crear una lista con números y caracteres mezclados.
- Concatena dos listas númericas. (Operador **++**).
- Concatena dos cadenas.
- Añade un elemento al principio de una lista. Añade dos elementos. (Operador **:**).
- Añade un elemento al final de la lista.
- Accede mediante índices a los elementos de una lista. (Operador **!!**).
- Crea la lista `[[2,3,4],[5,7],[3,7,8]]` y accede a sus elementos.
- Haz actuar la función **succ** sobre cada elemento de la lista. (Comando **map**).

Listas 2

Construye la lista numérica `[4,8,15,16,23,42]` y la cadena `MazingerZ`.

- Aplica la función **head** a las listas.
- Aplica la función **tail** a las listas.
- Cambia el primer número de la lista por un 90.
- Aplica la función **last** a las listas.
- Aplica la función **init** a las listas.
- Aplica la función **length** a las listas.
- Aplica la función **reverse** a las listas.
- Aplica la función **take** a las listas con distintos valores numéricos.
- Aplica la función **drop** a las listas con distintos valores numéricos.
- Aplica las funciones **maximum**, **minimum**, **sum** y **product** a la lista numérica.
- Comprueba si distintos números o caracteres pertenecen a las listas. (Operador **elem**).

Rangos (Sucesiones aritméticas)

- La lista de los números pares empezando en 2 y terminando en 100.
- La lista de los 100 primeros números. Súmalos.
- La factorial de 25.
- La lista de los números del 100 al 1.
- La lista `[6,9..100]`.
- Listas con caracteres.
- La lista infinita de todos los números pares. (**Control + C** para detener).
- Repite eternamente la lista `[3,7,9]` (Comando **cycle**).
- Extrae los 10 primeros números de la lista anterior.
- Repite infinitamente el número 6. (Comando **repeat**).
- Repite 7 veces el número 10. (Comando **replicate**).

Listas intensionales

- Construye la lista (infinita) $\{2x | x \in \mathbb{N}\}$. Genera únicamente 20 elementos de esta lista utilizando **take**. Lo mismo pero variando el «generador».
- Construye la lista de los primeros 10 números cuadrados. Y la lista de las 10 primeras potencias de 2.
- Construye la lista de los números del 50 al 100 pero que además el resto de la división entre 7 sea 3. La lista anterior pero que además los números sean pares.
- Genera todos las parejas de números del 1 al 3 utilizando dos generadores. Haz lo mismo pero evitando repetir parejas.
- Construye una lista con todas las vocales de la frase “en un lugar de la marcha”.
- ⋆ Construye una lista con todos los divisores del número 60. Lo mismo pero con 59.
- ⋆ Construye una función que calcule todos los divisores de un número entero.
- ⋆ Construye una función para identificar números primos.
- ⋆ Construye una función para identificar números perfectos.

Tuplas

- Construye una tupla con elementos no homogeneos.
- Utiliza las funciones **fst** y **snd** sobre la dupla (“im”, “presionante”).
- Utiliza la función **zip** sobre las listas:

  [1, 2, 3, 4] y [“Uno”, “Dos”, “Tres”, “Cuatro”]

- Añada algún elemento más a alguna de las listas anteriores y ejecuta de nuevo **zip**.
- Utiliza **zip** sobre las listas:

  [1..] y [“Uno”, “Dos”, “Tres”, “Cuatro”]

- Encontrar todas las ternas pitagóricas cuyos números sean menores que 20.
- Todas las ternas pitagóricas pero que además el perímetro sea 24.
- Comprobar el teorema de Fermat para algún exponente.

Tipos de datos («Type»)

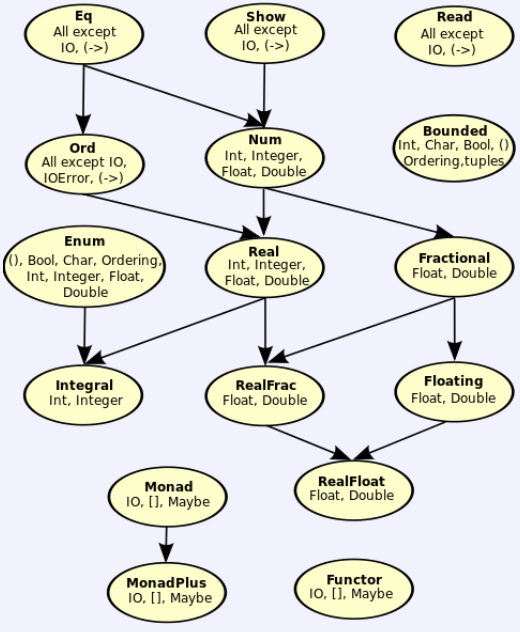
- Calcula el tipo (comando **:t**) de los siguientes elementos:

$$a)\,'\mathrm{n}' \qquad b)\,\mathrm{True} \qquad c)\,''\mathrm{Hola}'' \qquad d)\,4 == 5$$

- Calcula el tipo del número 27.
- Convierte 27 en tipo **Int**, **Integer**, **Float** y **Double**. Opera con dicho número y calcula su tipo.
- Convierte la lista `[4,5,6,7]` a tipo **Double** y pregunta por su tipo.
- Pregunta por el tipo de la función **doble**.
- Declara la función doble de tipo **Int -> Int** y vuelve a calcular su tipo.
- Declara la función doble de tipo **Double -> Double** y vuelve a calcular su tipo.
- Calcula el tipo de la función **media3**.
- Calcula el tipo de la función **quitaVocales**.
- Calcula el tipo de la función **head** e **init**.

Clases de tipo («Typeclass»)

- Calcula el tipo de la función ==. (Clase **Eq**).
- Calcula el tipo de la función >. (Clase **Ord**).
- Calcula el tipo de la función **succ**. (Clase **Enum**).
- Calcula el tipo de la función **+**. (Clase **Num**)



Ajuste de patrones («pattern matching»)

- Construye la función $f : \mathbb{N} \to \mathbb{N}$ dada por:

$$f(x) = \begin{cases} 1000 & \text{si } x = 7 \\ 0 & \text{si } x \neq 7 \end{cases}$$

- Construye una función que asocie a cada entero entre 1 y 4 su nombre en castellano.
- Elimina la excepción de la función anterior añadiendo un caso general.
- Pon el caso general en primer lugar en la función anterior.
- Define de forma recursiva la factorial.
- Construye la función que suma dos vectores bidimensionales. Otra con el producto escalar.
- Construye la función que extraiga la primera componente de una tripla. (Uso del comodín).
- Implementa con patrones la funciones **length**, **sum**, **product** y **head**.

Guardas

- Construye la función f dada por:

$$f(x) = \begin{cases} x & \text{si } \leq 0 \\ x^2 & \text{si } x > 0 \end{cases}$$

- Construye la función f dada por:

$$f(x) = \begin{cases} x & \text{si } x \leq 0 \\ 5 & \text{si } 0 < x < 5 \\ x^2 & \text{si } x \geq 5 \end{cases}$$

- Construye la función f dada por:

$$f(x) = \begin{cases} \texttt{"joven"} & \text{si } 0 \leq x \leq 18 \\ \texttt{"adulto"} & \text{si } 18 < x < 65 \\ \texttt{"viejo"} & \text{si } x \geq 65 \end{cases}$$

- Construye la función **abs** con guardas. Lo mismo con la función **signum**.

Definiciones locales («where»y «let»)

- Analiza la orden **where** en la función `preguntaIMC`.
- Define una constante global y utilízala en la función. La constante es accesible desde cualquier lugar.
- Define la constante como local y comprueba que no es accesible fuera de la función.
- Construye una función que devuelva una lista con todos los números pares hasta el argumento, utilizando una clausura **where**. La función creada no es accesible globalmente.
- Define una constante global utilizando **let**.
- Define una constante local utilizando **let . . . in**.
- Calcula el área total de un cilindro utilizando clausuras **let . . . in**.

Funciones recursivas

Crea recursivamente las siguientes funciones

- La función factorial.
- La función **sum**, **product** y **length**.
- La función **maximum**.
- La función **replicate**.
- La función **reverse**.
- La función **fibonacci**.
- La función **zip**.

Funciones «currificadas». Secciones

- Construye la función de dos argumentos **suma**. Analiza su tipo. «Currifica» dicha función. Analiza el nuevo tipo. Utiliza dicha función con **map**.
- Construye una función que multiplique tres argumentos y «currificala». Analiza los tipos de las funciones currificadas.
- «Currifica» el operador multiplicación. El resultado de esta operación se llama **sección**.
- Crea secciones con otro operador. Utiliza también la notación prefija.
- Crea secciones con operadores de comparación.
- Crea una función que devuelve *True* en el caso en que el caracter sea una letra mayúscula.

Funciones de orden superior

- La función **dosVeces** como función de orden superior. Análisis de su tipo. Definición con la composición de funciones.
- Analiza el tipo de la función **map**. Construye la función **map** usando un ajuste de patrones. Comprueba el tipo de la nueva función.
- Construye la función **map** utilizando listas intensivas.
- Analiza el tipo y el uso de la función **zipWith**. Construye una función que simule a dicha función.
- Analiza el tipo y el uso de la función **filter**. Construye una función que simule a dicha función usando patrones.
- Construye la función **filter** con listas intensivas.

Plegados

- Construye las funciones **sum**, **product** y **and** utilizando **foldl1** y **foldr1**.
- Simula la función **maximum** con pliegues.
- Convierte una lista de cifras en un número decimal utilizando **foldl1**.
- Define recursivamente las funciones **sum**, **product** y **and**.
- Define las funciones anteriores empleando **foldr**.
- Define las funciones anteriores empleando **flodl**.

Definir funciones que extraen elementos de una tupla. Utilizando el guion bajo.

crea tipos definidos por el usuario.

crear lista de booleanos

funcion length creada por recursionkjhgfs

extraer una sublista de una lista dada con drop y take

Escribir multilinea con : y :

definicion valor absoluto con if then else

la funcion signo con ifs anidados

definir lo mismo con guardas. Uso de otherwise

Definicion de not por patrones

Uso del comodin para patrones