Complejos

Taller de Álgebra I

Primer cuatrimestre 2018

Complejos

type

Definamos un renombre de tipos para Complejos

```haskell
type Complejo = (Float, Float)
```

donde el primer elemento es la parte real y el segundo la parte imaginaria.

Ejercicio

Implementar las funciones

1. `suma :: Complejo -> Complejo -> Complejo`
2. `producto :: Complejo -> Complejo -> Complejo`
3. `re :: Complejo -> Float,  im :: Complejo -> Float`
4. `modulo :: Complejo -> Float`
5. `argumento :: Complejo -> Float`

```
Ejemplo> suma (2,3) (-1,4)
(1.0,7.0)
```

```
Ejemplo> producto (2,3) (-1,4)
(-14.0,5.0)
```

Más ejercicios

Implementar

1. `conjugado :: Complejo -> Complejo`
2. `inverso :: Complejo -> Complejo`
3. `cociente :: Complejo -> Complejo -> Complejo`
4. `potencia :: Complejo -> Integer -> Complejo`

```
Ejemplo> potencia (1, 3) 8
(-8432.0,-5376.0)
```

```
Ejemplo> potencia (1, 3) (-8)
(-8.432e-5,5.376e-5)
```

```
Ejemplo> potencia (0, 1) 4
(1.0,0.0)
```

Ejercicio 6

Agregar al código

```
numerador = (1, sqrt 3) :: Complejo
denominador = (1, -1) :: Complejo
a = (-1, sqrt 3) :: Complejo
```

Ejercicio

1. El ejercicio 6 i) de la práctica 6 pide encontrar la forma binomial de $\left(\frac{1+\sqrt{3}i}{1-i}\right)^{17}$. Encontrarla como una expresión usando las constantes anteriores[a].
2. El ejercicio 6 ii) pide encontrar la forma binomial de $(-1+\sqrt{3}i)^n$ para cada $n \in \mathbb{N}$. Hacer una función `primerasPotencias :: Integer -> [Complejo]` que dado un natural n devuelva la lista de las primeras n potencias de a desde 1 hasta n.

[a] Cuidado: lo que encontramos es una aproximación

```
Ejemplo> primerasPotencias 3
[(-1.0,1.7320508),(-2.0,-3.4641016),(8.0,0.0)]
```

Raíces n-ésimas

Dado un número natural n, las raíces n-ésimas de la unidad están dadas por

$$\cos\left(\frac{2k\pi}{n}\right) + i\sin\left(\frac{2k\pi}{n}\right), \qquad 0 \leq k < n.$$

Las primitivas son, de acuerdo a esta notación, aquellas con $(k, n) = 1$.

Ejercicios

1. Implementar la función `raicesNEsimas:: Integer -> Set Complejo` que, dado un número natural n, devuelve el conjunto de las raíces n-ésimas de la unidad.
2. (Ejercicio 11, práctica 6) Implementar la función `moebius :: Integer -> Complejo` que, dado un número natural n, devuelve la suma de las raíces n-ésimas *primitivas* de la unidad.