Recuperatorio del segundo trabajo práctico
Números complejos

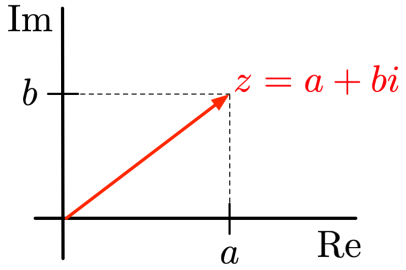
Taller de Álgebra 1 - Segundo cuatrimestre de 2022

Llamamos *conjunto de los números complejos* a

$$\mathbb{C} = \{a + bi : a, b \in \mathbb{R}\}.$$

- Si $z = a + bi \in \mathbb{C}$, a es la *parte real* y b es la *parte imaginaria* de z.
- La suma y el producto de números complejos se dan de manera natural:
  - $(a + bi) + (c + di) = (a + c) + (b + d)i$
  - $(a + bi) \cdot (c + di) = (ac - bd) + (ad + bc)i$

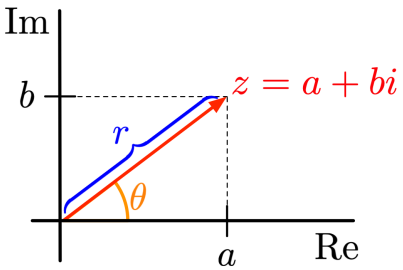
Podemos representar todos los números complejos en un plano: el *plano complejo*.



Si $z = a + bi \in \mathbb{C}$, lo representamos en el punto (a, b) del plano. El *módulo* de z es su distancia al origen, esto es: $|z| = \sqrt{a^2 + b^2}$. El *conjugado* de z es su reflexión con respecto al eje horizontal, esto es: $\overline{z} = a - bi$. Todo número complejo no nulo z tiene un inverso, y viene dado por $z^{-1} = \frac{\overline{z}}{|z|^2}$.

Coordenadas polares

Podemos representar un numero complejo usando coordenadas polares:



Si $z \in \mathbb{C}$ es como en el dibujo, se tiene que r es su módulo; llamamos a $\theta \in [0, 2\pi)$ su *argumento*. Más precisamente, despejando:

- $a = r \cos\theta$
- $b = r \sin\theta$

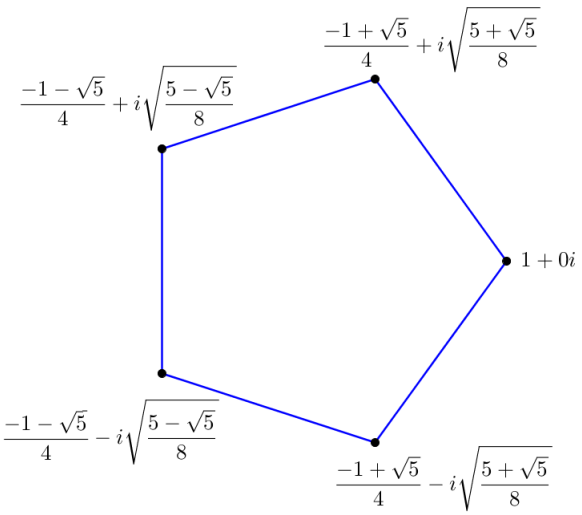
Luego, $z = r(\cos\theta + i \sin\theta)$; más aún, si $a, b > 0$ se tiene que $\theta = \arctan(b/a)$.

Raíces n-ésimas de la unidad

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

$$\zeta_k = \cos\left(\frac{2k\pi}{n}\right) + i \sin\left(\frac{2k\pi}{n}\right), \qquad 0 \leq k < n. \quad k \in \mathbb{Z}$$

Estas son las soluciones de $\{z \in \mathbb{C} : z^n = 1\}$.



Raíces quintas de la unidad

Trabajaremos con el siguiente renombre de tipos para complejos:

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

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

Los polinomios con coeficientes complejos se representan con el tipo

```
type Polinomio = [Complejo]
```

El polinomio $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ se representa con la lista

$$[a_n, a_{n-1}, \ldots, a_1, a_0].$$

Por ejemplo, el polinomio $(1+i)x^3 - (1-2i)x + 4i$ se representa con la lista

$$[(1,1), (0,0), (-1,2), (0,4)].$$

Invariante del tipo. La lista de coeficientes no empieza con el complejo 0. Se puede suponer que el invariante se cumple en los polinomios que reciben nuestras funciones.

Ejercicio 1

Implementar las funciones

1.1. `re :: Complejo -> Float`

que devuelve la parte real de un complejo.

1.2. `im :: Complejo -> Float`

que devuelve la parte imaginaria de un complejo.

1.3. `suma :: Complejo -> Complejo -> Complejo`

que devuelve la suma de dos complejos.

1.4. `producto :: Complejo -> Complejo -> Complejo`

que devuelve el producto de dos complejos.

1.5. `conjugado :: Complejo -> Complejo`

que devuelve el conjugado de un complejo.

1.6. `inverso :: Complejo -> Complejo`

que devuelve el inverso de un complejo no nulo.

1.7. `cociente :: Complejo -> Complejo -> Complejo`

que dados $z, w \in \mathbb{C}$ con $w \neq 0$, devuelve z/w.

1.8. `potencia :: Complejo -> Integer -> Complejo`

que dados $z \in \mathbb{C}$ y $k \in \mathbb{N}$, devuelve z^k.

1.9. `raicesCuadratica :: Float -> Float -> Float -> (Complejo, Complejo)`

que, dada una función cuadrática ax^2+bx+c con $a, b, c \in \mathbb{R}$, $a \neq 0$, devuelva sus dos raíces complejas. En caso de haber una sola, devolverla dos veces.

Ejemplos

```
*Main> re (3,4)
3.0
*Main> im (3,4)
4.0
*Main> suma (3,4) (-1,1)
(2.0, 5.0)
*Main> producto (3,4) (-1,1)
(-7.0, -1.0)
*Main> inverso (3,4)
```

```
(0.12, -0.16)
*Main> cociente (3,4) (-1,1)
(0.5, -3.5)
*Main> potencia (3,4) 3
(-117.0, 44.0)
*Main> raicesCuadratica 1 2 5
((-1.0, -2.0), (-1.0, 2.0))
```

Nota 1: el orden de los pares de `raicesCuadratica` es indistinto, así que `raicesCuadratica 1 2 5` también podría devolver `((-1.0, 2.0), (-1.0, -2.0))`

Ejercicio 2

Implementar las funciones

2.1. `modulo :: Complejo -> Float`

que devuelve el módulo de un número complejo.

2.2. `distancia:: Complejo -> Complejo -> Float`

que, dados $z, w \in \mathbb{C}$, devuelve $|z - w|$.

2.3. `argumento :: Complejo -> Float`

que devuelve el argumento de un número complejo.

2.4. `pasarACartesianas :: Float -> Float -> Complejo`

que toma $r \geq 0$ y $\theta \in [0, 2\pi)$ y devuelve el complejo z que tiene módulo r y argumento θ.

2.5. `raizCuadrada :: Complejo -> (Complejo,Complejo)`

que, dado $z \in \mathbb{C}$, devuelve los dos complejos w que satisfacen $w^2 = z$.

2.6. `raicesCuadraticaCompleja :: Complejo -> Complejo -> Complejo -> (Complejo,Complejo)`

que, dado un polinomio $az^2 + bz + c$, con $a, b, c \in \mathbb{C}$, $a \neq 0$, devuelva sus dos raíces complejas.

Ejemplos

```
*Main> modulo (3,4)
5.0
*Main> argumento (-1,-1)
3.926991
*Main> distancia (pasarACartesianas (sqrt 2) (pi)/4)) (1,1) < 1/10^4
```

```
True
*Main> distancia (potencia (fst (raizCuadrada (0,1))) 2) (0,1) < 1/10^4
True
*Main> distancia (potencia (snd (raizCuadrada (0,1))) 2) (0,1) < 1/10^4
True
*Main> raizCuadrada (1,2)
((1.2720196,0.78615147),(-1.2720196,-0.78615147))
*Main> raicesCuadraticaCompleja (1,0) (0,1) (2,0)
((0,-2.0), (0,1.0))
```

Nota 2: Al igual que con `raicesCuadratica`, el orden de las raices que devuelve `raicesCuadraticaCompleja` es indistinto.

Nota 3: Las funciones pueden devolver números con algunos errores de precisión. Por ejemplo, podría suceder

```
*Main> raizCuadrada (1,0)
((1.0,0.0),(-1.0,-8.742278e-8))
```

cuando el resultado correcto sería

```
*Main> raizCuadrada (1,0)
((1.0,0.0),(-1.0,0))
```

Estos errores son inevitables; dependen del modo de calcular las funciones y del mecanismo interno que tiene Haskell para representar los números.

Ejercicio 3

Implementar las funciones

3.1. `raicesNEsimas :: Integer -> [Complejo]`

que, dado n natural, devuelve la lista con las raíces n-ésimas de la unidad.

3.2. `sonRaicesNEsimas :: Integer -> [Complejo] -> Float -> Bool`

que, dados n natural, una lista de complejos $[z_1, \ldots, z_m]$ y un número positivo ϵ devuelve `True` si $|z_k^n - 1| < \epsilon$ para todo $1 \leq k \leq m$, y `False` en otro caso.

Ejemplos

```
*Main> raicesNEsimas 5
[(1.0,0.0),(0.30901697,0.95105654),(-0.80901706,0.5877852),
(-0.80901694,-0.58778536),(0.30901712,-0.9510565)]
*Main> sonRaicesNEsimas 5 (raicesNEsimas 5) 10**(-4)
True
*Main> sonRaicesNEsimas 10 (raicesNEsimas 5) 10**(-4)
```

```
True
*Main> sonRaicesNEsimas 3 (raicesNEsimas 5) 10**(-4)
False
```

Nota 4: Nuevamente, podrían aparecer errores de precisión, por ejemplo:

```
*Main> raicesNEsimas 4
[(1.0,0.0),(-4.371139e-8,1.0),(-1.0,-8.742278e-8),(1.1924881e-8,-1.0)]
```

Ejercicio 4

Implementar las funciones

4.1. `esRaizDe :: Polinomio -> Complejo -> Float -> Bool`

que decida si un número complejo z es raíz (con cierta precisión ϵ) de un polinomio complejo $P(x)$; es decir, si $|P(z)| < \epsilon$.

Nota: para pasar de `Int` a `Integer`, usar `toInteger`

4.2. `cuentaRaices :: Polinomio -> [Complejo] -> Float -> Integer`

que dado un polinomio $P(x)$, una lista de números complejos $[z_1, \ldots, z_m]$ y un número positivo ϵ, devuelva la cantidad de i tal que $|P(z_i)| < \epsilon$ para $i = 1, \ldots, m$.

Ejemplos

```
*Main> esRaizDe [(1,2),(-1,3),(0,15)] (-1,2) 0.1
True
*Main> esRaizDe [(1,2),(-1,3),(0,15)] (-1,2.1) 0.1
False
*Main> esRaizDe [(1,2),(-1,3),(0,15)] (-1,2.1) 1.5
True

*Main> cuentaRaices [(1,2),(-1,3),(0,15)] [(-1,2),(0,-3.01),(0,-3),(1,1)] 1
3
*Main> cuentaRaices [(1,2),(-1,3),(0,15)] [(-1,2),(0,-3.01),(0,-3),(1,1)] 0.1
2
*Main> cuentaRaices [(1,2),(-1,3),(0,15)] [] 0.1
0
```

Condiciones de entrega

El trabajo práctico se realiza en grupos de 3. Leer la solapa *Evaluación* en el campus virtual del Taller.

Está prohibido subir el código que implementen a repositorios públicos, así como también usar total o parcialmente soluciones implementadas por otros grupos. No está permitida la interacción con otros grupos para discutir las soluciones de los ejercicios propuestos.

No evaluaremos la eficiencia de los algoritmos que propongan para resolver los ejercicios (y por ende, el tiempo en que tardan en ejecutarse las funciones) pero sí la correctitud del código. Algunas funciones que implementen pueden demorar minutos en arrojar el resultado, no se preocupen por el tiempo sino porque devuelvan el valor correcto.

La entrega consiste en un único archivo `.hs` con las funciones de los ejercicios implementadas, junto con todas las funciones auxiliares que sean necesarias para ejecutarlas. Las funciones deben respetar la signatura (nombres y parámetros) especificados en cada ejercicio, dado que serán testeadas automáticamente. El archivo que entregan tiene que poder compilarse solo, y sin llamar a otro módulo. Les pedimos que utilicen como base para escribir el código el archivo RTP2.hs que se encuentra en el campus virtual del Taller.

El archivo entregado debe tener la forma "`apellido1-apellido2-apellido3.hs`", respetando el orden alfabético de los apellidos. Por ejemplo, el grupo formado por los estudiantes María López, Ariel Gómez, y Juan Pérez debe entregar un archivo llamado "`Gomez-Lopez-Perez.hs`".

Además, en las primeras líneas del archivo deben ir los nombres completos comentados, empezando por el apellido y siguiendo por el o los nombres sin comas y la dirección de email. En el ejemplo:

```
-- López María lopezmaria@xxx.com
-- Gómez Ariel gomezariel@yyy.com
-- Pérez Juan perezjuan@zzz.com
```

La entrega se debe llevar a cabo a través del campus virtual, subiendo el archivo con el código con el mecanismo disponible dentro de la solapa *Evaluación* en el campus virtual del taller. Uno y solo un integrante será el encargado de subir el Trabajo Práctico al campus en representación de todo el grupo.

Se evaluará la corrección de las funciones implementadas, la declaratividad y claridad del código, y que las funciones auxiliares (si las hay) tengan nombres apropiados.

Si tienen dudas o consultas respecto del trabajo práctico, pueden enviar un mail a la

lista de docentes `algebra1-doc (arroba) dc.uba.ar`. No hacer consultas a través de la lista de mails de alumnos.

Además del código, deberán rendir un coloquio, en el que se conversará **individualmente con cada integrante del grupo** sobre el trabajo realizado, debiendo responder diversas preguntas sobre lo que entregaron. El día y horario del coloquio lo acordarán por email con el docente que haya corregido su trabajo. Ver sección *Evaluación* en el campus virtual.

Fecha de entrega: Hasta el domingo 4 de diciembre a las 23:55 hs por el campus virtual del Taller en la solapa *Evaluación*.