アルゴリズムとデータ構造入門 第九回課題

1029-24-9540 山崎啓太郎

February 7, 2013

1 複素数システム

```
1  (define (square z) (* z z))
2  (define (real-part z) (car z))
3  (define (imag-part z) (cdr z))
4  (define (magnitude z)
5    (sqrt (+ (square (real-part z))
6             (square (imag-part z)) )))
7  (define (angle z)
8    (atan (imag-part z) (real-part z)))
9
10 (define (make-from-real-imag x y)
11   (cond
12     ((and (number? x) (number? y))  (cons x y))
13     ((number? x) (make-from-real-imag (- x (imag-part y)) (real-part y)))
14     ((number? y) (make-from-real-imag (real-part x) (+ (imag-part x) y)))
15     (else (make-from-real-imag (- (real-part x) (imag-part y)) (+ (imag-par
16
17 (define (add-complex z1 z2)
18   (make-from-real-imag
19     (+ (real-part z1) (real-part z2))
20     (+ (imag-part z1) (imag-part z2)) ))
21 (define (sub-complex z1 z2)
22   (make-from-real-imag
23     (- (real-part z1) (real-part z2))
24     (- (imag-part z1) (imag-part z2)) ))
25 (define (mul-complex z1 z2)
26   (make-from-real-imag
27     (- (* (real-part z1) (real-part z2)) (* (imag-part z1) (imag-part z2)))
28     (+ (* (imag-part z1) (real-part z2)) (* (real-part z1) (imag-part z2)))
29 (define (div-complex z1 z2)
```

```
30    (make-from-real-imag
31      (/ (+ (* (real-part z1) (real-part z2)) (* (imag-part z1) (imag-part z2
32      (/ (- (* (imag-part z1) (real-part z2)) (* (real-part z1) (imag-part z2
33
34  (define stringify-complex (lambda (z)
35    (cond
36      ((= (imag-part z) 0) (number->string (real-part z)))
37      ((= (real-part z) 0)
38        (cond
39          ((= (imag-part z) 1) "i")
40          ((= (imag-part z) -1) "-i")
41          (else (string-append (number->string (imag-part z)) "i")) ))
42      ((> (imag-part z) 0)
43        (if (= (imag-part z) 1)
44          (string-append (number->string (real-part z)) "+" "i")
45          (string-append (number->string (real-part z)) "+" (number->string (
46      (else
47        (if (= (imag-part z) -1)
48          (string-append (number->string (real-part z)) "-" "i")
49          (string-append (number->string (real-part z)) (number->string (imag-
```

2 実行例

(stringify-complex (make-from-real-imag 3 4)) => "3+4i"
(stringify-complex (make-from-real-imag 3 0)) => "3"
(stringify-complex (make-from-real-imag 0 3)) => "3i"
(stringify-complex (make-from-real-imag 0 1)) => "i"
(stringify-complex (make-from-real-imag 0 -1)) => "-i"
(stringify-complex (make-from-real-imag 3 1)) => "3+i"
(stringify-complex (make-from-real-imag 3 -2)) => "3-2i"
(stringify-complex (make-from-real-imag 3 -1)) => "3-i"
(stringify-complex (add-complex (make-from-real-imag 3 -1) (make-from-real-imag 2 4))) => "5+3i"
(stringify-complex (sub-complex (make-from-real-imag 3 -1) (make-from-real-imag 2 4))) => "1-5i"
(stringify-complex (mul-complex (make-from-real-imag 3 -1) (make-from-real-imag 2 4))) => "10+10i"
(stringify-complex (div-complex (make-from-real-imag 3 -1) (make-from-real-imag 2 4))) => "0.1-0.7i"
(stringify-complex (make-from-real-imag (make-from-real-imag 3 -1) (make-from-real-imag 2 4))) => "-1+i"

3　説明

make-from-real-imag 関数を使って、x+iy となる形の複素数を生成している。
add-complex、sub-complex、mul-complex、div-complex 関数にて四則演算が可能である。
また、make-from-real-imag 関数には複素数を渡すこともでき、その際には返り値が完結になるよう計算される。
簡略化は、make-from-real-imag 関数で生成された複素数を文字列化する stringify-complex 関数にて行なっている。
例として、簡略化されないと”0+1i” と表示されるものが、”i” と表示されるような処理がされている。（その他の例は実行例に有）