第三章作业

2019年9月8日 0:47

```
#lang racket
(define (make-accumulator x)
  (define (add amount)
    (begin (set! x (+ x amount))
    x
    )
    )
  add
  )

(define A (make-accumulator 5))
(A 10)
(A 10)

(define (make-monitored f)
  (define (helper f times)
    (define (run)
      (begin (set! times (+ times 1))
                  (lambda (t) (f t))))
    (define (how-many-calls?)
      times)
    (define (reset)
      (set! times 0))
    (define (dispatch m)
      (cond ((eq? m 'how-many-calls?) (how-many-calls?))
            ((eq? m 'reset) (reset))
            (else ((run) m))))
    dispatch
  )

(helper f 0)
)
(define s (make-monitored sqrt))
(s 100)
(s 'how-many-calls?)
(s 25)
(s 'how-many-calls?)
(s 'reset)
(s 'how-many-calls?)
(s 'how-many-calls?)
```