Scheme

Кевролетин В.В. 236гр.

24 мая 2011 г.

Задание27

Условие

make-accumulator

Решение

```
(define (make-accumulator num)
  (lambda (a)
    (set! num (+ num a))
    num))

;;usage
(define A (make-accumulator 5))
(A 10) ;; > 15
(A 25) ;; > 50
```

Задание28

Условие

make-monitored

Решение

```
(define (make-monitored funct)
  (let ((counter 0))
    (lambda (command)
      (cond ((eq? command 'how-many) counter)
            ((eq? command 'reset) (set! counter 0)
             0)
            (else (set! counter (+ counter 1))
                  (funct command))))))

;;usage
(define sm (make-monitored (lambda (x) (* x x))))
(sm 'how-many) ;; > 0
(sm 2)         ;; > 4
(sm 'how-many) ;; > 1
(sm 3)         ;; > 9
```

```
(sm 'how-many)  ;; > 2
(sm 'reset)     ;; > 0
(sm 'how-many)  ;; > 0
```

Задание29

Условие

make-account-pass

Решение

```
(define (make-account-pass balance password)
  (define (withdraw a)
    (if (>= balance a)
        (begin
          (set! balance (- balance a))
          balance)
        "Insufficient funds"))
  (define (deposit a)
    (set! balance (+ balance))
    balance)

  (let ((err-cnt 0)
        (access-denided 0))
    (define (dispatch m p)
      (if (eq? access-denided 1) (lambda (a) "Access denided!")
          (if (eq? p password)
              (begin
                (set! err-cnt 0)
                (cond ((eq? m 'withdraw) withdraw)
                      ((eq? m 'deposit) deposit)))
              (begin
                (set! err-cnt (+ err-cnt 1))
                (if (> err-cnt 3) (begin
                                    (set! access-denided 1)
                                    (dispatch m p))
                    (lambda (a) "Wrond password")))))))
    dispatch))

;;usage
(define acc (make-account-pass 100 'good-passw))
((acc 'withdraw 'good-passw) 20) ;; > 80
((acc 'deposit 'good-passw) 10)  ;; > 90
((acc 'deposit 'bad-passw) 10)   ;; > "Wrond password"
((acc 'deposit 'bad-passw) 10)   ;; > "Wrond password"
((acc 'deposit 'bad-passw) 10)   ;; > "Access denided!"
((acc 'withdraw 'good-passw) 20) ;; > "Access denided!"
```

Задание29

Условие

make-account-pass

Решение

```
(define (call-the-cops)
  (print "Hey, you! Put your hands up!!!"))

(define (make-account-pass balance password)
  (define (withdraw a)
    (if (>= balance a)
        (begin
          (set! balance (- balance a))
          balance)
        "Insufficient funds"))
  (define (deposit a)
    (set! balance (+ balance))
    balance)

  (let ((err-cnt 0)
        (access-denided 0))
    (define (dispatch m p)
      (if (eq? access-denided 1) (lambda (a) (call-the-cops))
          (if (eq? p password)
              (begin
                (set! err-cnt 0)
                (cond ((eq? m 'withdraw) withdraw)
                      ((eq? m 'deposit) deposit)))
              (begin
                (set! err-cnt (+ err-cnt 1))
                (if (> err-cnt 3) (begin
                                    (set! access-denided 1)
                                    (call-the-cops))
                    (lambda (a) "Wrond password"))))))
    dispatch))

;;usage
(define acc (make-account-pass 100 'good-passw))
((acc 'withdraw 'good-passw) 20) ;; > 80
((acc 'deposit 'good-passw) 10)  ;; > 90
((acc 'deposit 'bad-passw) 10)   ;; > "Wrond password"
((acc 'deposit 'bad-passw) 10)   ;; > "Wrond password"
((acc 'deposit 'bad-passw) 10)   ;; > "Hey, you! Put your hands up!!!"
((acc 'withdraw 'good-passw) 20) ;; > "Hey, you! Put your hands up!!!"
```