Лабораторная работа № 5. Интерпретатор стекового языка программирования

9 декабря 2024 г.

Булат Валиуллин ИУ9-11Б

Цель работы

Познакомиться с интерпретацией низкоуровневого кода на примере FORTH-подобного языка программирования.

Реализация

```
(define (interpret program stack)
  (define (process-word word ind stack dict)
    (cond
      ((number? word) (cons word stack))

      ;; Встроенные операции:
      ((eq? word '+)
       (let ((n1 (car stack))
             (n2 (cadr stack)))
         (cons (+ n1 n2) (cddr stack))))

      ((eq? word '-)
       (let ((n1 (car stack))
             (n2 (cadr stack)))
         (cons (- n2 n1) (cddr stack))))

      ((eq? word '*)
       (let ((n1 (car stack))
             (n2 (cadr stack)))
         (cons (* n1 n2) (cddr stack))))

      ((eq? word '/)
       (let ((n1 (car stack))
             (n2 (cadr stack)))
```

```
   (cons (/ n2 n1) (cddr stack))))

((eq? word 'mod)
 (let ((n1 (car stack))
       (n2 (cadr stack)))
   (cons (remainder n2 n1) (cddr stack))))

((eq? word 'neg)
 (cons (-(car stack)) (cdr stack)))

((eq? word '=)
 (let ((n1 (car stack))
       (n2 (cadr stack)))
   (if (equal? n1 n2)
       (cons -1 (cddr stack))
       (cons 0 (cddr stack)))))

((eq? word '<)
 (let ((n1 (car stack))
       (n2 (cadr stack)))
   (if (> n1 n2)
       (cons -1 (cddr stack))
       (cons 0 (cddr stack)))))

((eq? word '>)
 (let ((n1 (car stack))
       (n2 (cadr stack)))
   (if (< n1 n2)
       (cons -1 (cddr stack))
       (cons 0 (cddr stack)))))

((eq? word 'drop)
 (cdr stack))

((eq? word 'swap)
 (let ((n1 (car stack))
       (n2 (cadr stack)))
   (cons n2 (cons n1 (cddr stack)))))

((eq? word 'dup)
 (cons (car stack) stack))

((eq? word 'over)
 (cons (cadr stack) stack))

((eq? word 'rot)
```

```
     (cons (caddr stack) (cons (cadr stack) (cons (car stack)
                                                  (cdddr stack)))))

    ((eq? word 'depth)
     (cons (length stack) stack))

    ;; Невстроенное слово
    (else #f)
    ))

(define (lookup dict w)
  (if (eq? (caar dict) w)
      (cdar dict)
      (lookup (cdr dict) w)))

(define (skip-to-endif program ind)
  (let ((w (vector-ref program ind)))
    (if (eq? w 'endif)
        (+ ind 1)
        (skip-to-endif program (+ ind 1)))))

(define (read-next-word
         program ind stack return-stack dict defining? current-word)
  (if (>= ind (vector-length program))
      stack
      (let ((word (vector-ref program ind)))
        (cond
          (defining?
            (if (eq? word 'end)
                (let ((word-name (car current-word))
                      (start (cdr current-word)))
                  (read-next-word program (+ ind 1) stack return-stack
                                  (cons (cons word-name start) dict) #f #f))
                (read-next-word program (+ ind 1) stack return-stack
                                dict #t current-word)))

          ;; Управляющие конструкции:
          ((eq? word 'define)
           (let* ((word-name (vector-ref program (+ ind 1)))
                  (start (+ ind 2)))
             (read-next-word program (+ ind 2) stack return-stack
                             dict #t (cons word-name start))))

          ((or (eq? word 'exit) (eq? word 'end))
           (if (null? return-stack)
               stack
```

```
                (let ((return-ind (car return-stack)))
                  (read-next-word program return-ind stack
                                  (cdr return-stack) dict #f #f))))

            ((eq? word 'if)
             (let ((flag (car stack)))
               (if (= flag 0)
                   (read-next-word program (skip-to-endif program (+ ind 1))
                                   (cdr stack) return-stack dict #f #f)
                   (read-next-word program (+ ind 1) (cdr stack)
                                   return-stack dict #f #f))))

            ((eq? word 'endif)
             (read-next-word program (+ ind 1) stack return-stack dict #f #f))

            ;; Не управляющие конструкции
            ((let ((new-stack (process-word word ind stack dict)))
               (if new-stack
                   (read-next-word program (+ ind 1)
                                   new-stack return-stack dict #f #f)
                   (let ((start (lookup dict word)))
                     (if start
                         (read-next-word
                          program start stack (cons (+ ind 1) return-stack)
                          dict #f #f)
                         )))))

            (else
             (read-next-word program (+ ind 1)
                             (process-word word ind stack dict)
                             return-stack dict #f #f))))))

  (read-next-word program 0 stack '() '() #f #f))
```

Тестирование

```
Welcome to DrRacket, version 8.15 [cs].
Language: R5RS; memory limit: 128 MB.
(interpret #(2 3 * 4 5 * +) '()) ok
(interpret #(define -- 1 - end 5 -- --) '()) ok
(interpret #(define abs dup 0 < if neg endif end 9 abs -9 abs) '()) ok
(interpret #(define =0? dup 0 = end define <0? dup 0 < end define signum =0?
if exit endif <0? if drop -1 exit endif drop 1 end 0
signum -5 signum 10 signum) '()) ok
(interpret #(define -- 1 - end define =0? dup 0 = end define =1? dup 1 = end
```

```
 define factorial =0? if drop 1 exit endif =1? if drop 1 exit endif dup -
-
  factorial * end 0 factorial 1 factorial 2 factorial 3 factorial
   4 factorial) '()) ok
(interpret #(define =0? dup 0 = end define =1? dup 1 = end define -
- 1 - end
 define fib =0? if drop 0 exit endif =1? if drop 1 exit endif --
dup -- fib
  swap fib + end define make-fib dup 0 < if drop exit endif dup fib swap
   -- make-fib end 10 make-fib) '()) ok
(interpret #(define =0? dup 0 = end define gcd =0? if drop exit endif swap
over mod gcd end 90 99 gcd 234 8100 gcd) '()) ok
#t
```

Вывод

Позади ещё одно незабываемое приключение сквозь недооценённый мной ранее мир программирования, во время которого я, кажется, окончательно понял, что такое этот ваш стек, и как же работает интерпретатор (ещё бы, свой писать пришлось, хоть и для низкоуровневого кода на примере FORTH-подобного языка, но я бы ни за что не поверил в это полгода назад, так что однозначно можно гордиться).