1 Лабораторная работа №4

Смирнов Пётр, ИУ7-65Б

1.1 Номер 1

Чем принципиально отличаются функции cons, list, append?

cons создает списковую ячейку и расставляет указатели на голову и хвост, передается 2 S-выражения;

list создает столько списковых ячеек, сколько аргументов;

append выстраивает в одну цепочку элементы всех списков, поставляемых в качестве аргументов. append объединяет элементы списков, расположенные лишь на самом верхнем уровне. append создаёт копии все списков, кроме последнего.

Каковы результаты вычисления следующих выражений?

```
(setf lst1 '(a b c) lst2 '(d e))

(cons lst1 lst2)    ; ((A B C) D E)
(list lst1 lst2)    ; ((A B C) (D E))
(append lst1 lst2) ; (A B C D E)
```

1.2 Номер 2

Каковы результаты вычисления следующих выражений и почему?

LAST возвращает последние N ячеек списка. Если N не указано, возвращается последняя списковая ячейка.

REVERSE переставляет элементы в обратном порядке.

```
(reverse '(a b c))          ; (C B A)
(reverse ())                ; NIL
(reverse '(a b (c (d))))    ; ((C (D)) B A)
(reverse '((a b c)))        ; ((A B C))
(reverse '(a))              ; (A)
(last '(a b c))             ; (C)
(last '(a b (c)))           ; ((C))
(last '(a))                 ; (A)
(last ())                   ; NIL
(last '((a b c)))           ; ((A B C))
```

1.3 Номер 3

Написать, по крайней мере, два варианта функции, которая возвращает последний элемент своего списка-аргумента.

```
(defun var1(arr) (car (reverse arr)))

(defun var2(arr) (car (last arr)))

(defun var3(arr)
      (cond ((car arr)
                  (nth (- (length arr) 1)
                       arr))))

(var3 '(1 2 3 4 5)) ; 5
(var3 ())           ; NIL
(var3 '(1))         ; 1
(var3 '(1 2))       ; 2
```

1.4 Номер 4

Написать, по крайней мере, два варианта функции, которая возвращает свой список-аргумент без последнего элемента.

```
(defun var1(arr)
    (nreverse (cdr (nreverse arr))))

(defun var2(arr)
    (cond ((cdr arr) (rplacd (last arr 2) nil) arr)))


(var1 '(1 2 3 4)) ; (1 2 3)
(var1 '(1 2))     ; (1)
(var1 '(1))       ; NIL
(var1 '())        ; NIL
(var2 '(1 2 3 4)) ; (1 2 3)
(var2 '(1 2))     ; (1)
(var2 '(1))       ; NIL
(var2 '())        ; NIL
```

1.5 Номер 5

Напишите функцию swap-first-last, которая переставляет в списке-аргументе первый и последний элементы.

```
; Данная реализация переставляет указатели
(defun swap-first-last-1 (arr)
    (cond ((not (cdr arr))
           arr)
          ((not (cddr arr))
             (let ((lastt (last arr)))
                  (rplacd (cdr arr) arr)
                  (rplacd arr nil)
                  lastt))
          (t (let ((ln (length arr))
                   (lastt (last arr)))
                  (rplacd lastt
                          (cdr arr))
                  (rplacd (nthcdr (- ln 2) arr)
                          arr)
                  (rplacd arr nil)
                  lastt))))

; Данная реализация просто меняет значения по car
(defun swap-first-last-2 (arr)
    (cond ((not (cdr arr)) arr)
          (t (let ((firstval (car arr)))
                (rplaca arr
                        (car (last arr)))
                (rplaca (last arr)
                        firstval)
                arr)))))



(swap-first-last-1 '(1 2 3 4 5)) ; (5 2 3 4 1)
(swap-first-last-1 '(1 2 3))     ; (3 2 1)
(swap-first-last-1 '(1 2))       ; (2 1)
(swap-first-last-1 '(1))         ; (1)
(swap-first-last-1 '())          ; NIL
```

1.6 Номер 6

Написать простой вариант игры в кости, в котором бросаются две правильные кости. Если сумма выпавших очков равна 7 или 11 – выигрыш, если выпало (1, 1) или (6, 6) – игрок имеет право снова бросить кости, во всех остальных случаях ход переходит ко второму игроку, но запоминается сумма выпавших очков. Если второй игрок не выигрывает абсолютно, то выигрывает тот игрок, у которого больше очков. Результат игры и значения выпавших костей выводить на экран с помощью функции print.

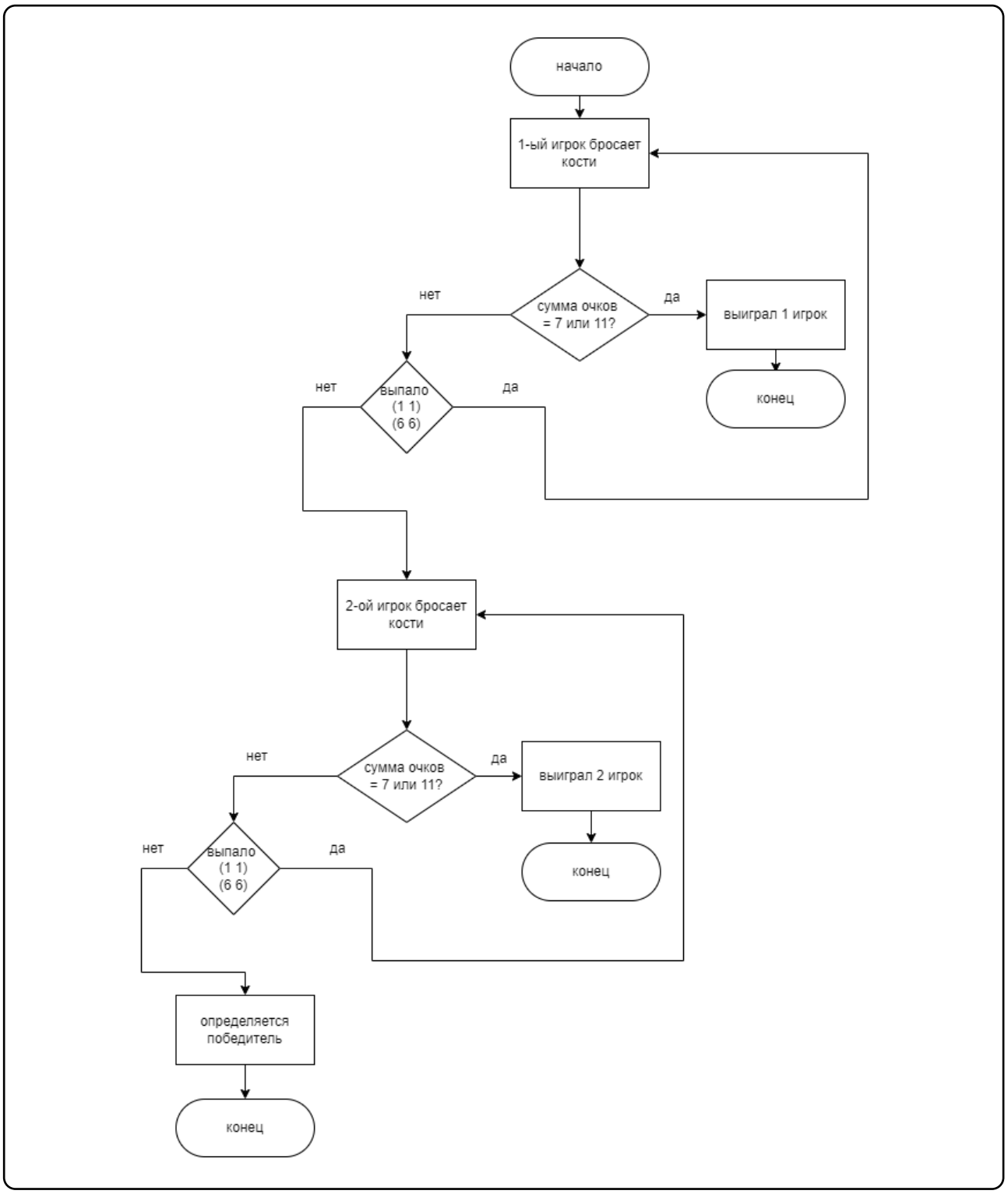

Рисунок 1.1 – Схема алгоритма игры в кости.

```
(setf *random-state* (make-random-state t))

(defun random-1-6 () (+ 1 (random 6)))
(defun ksum(arr) (+ (caddr arr) (cadddr arr)))

(defun format-throw (player)
    (print (format nil "~d-ый игрок бросил кости. Значения костей:
    ~d ~d~%"
            (car player) (cadr player) (caddr player))))

(defun format-win (player)
    (print (format nil "~d-ый игрок выиграл. Значения костей: ~d ~
   d~%"
            (car player) (cadr player) (caddr player))))

(defun play-cons(id)
  (cons id (cons (random-1-6)
        (cons (random-1-6) nil))))

(defun cond-after-throw(player)
  (let  ((k1 (cadr player))
         (k2 (caddr player)))

        (cond ((or (eql (+ k1 k2) 7)
                   (eql (+ k1 k2) 11))
               (format-win player) (cons 0 player))

              ((or (and (eql k1 1) (eql k2 1))
                   (and (eql k1 6) (eql k2 6)))
                (play (play-cons (car player))))

              (t (cons -1 player)))))

(defun play(player)
  (format-throw player)
  (cond-after-throw player))

(defun kosti()
    (let ((pl-1 (play (play-cons 1))))
        (cond ((eql (car pl-1) -1)
              (let* ((pl-2 (play (play-cons 2)))
                     (sum1 (ksum pl-1))
                     (sum2 (ksum pl-2)))
                (cond ((eql (car pl-2) -1)
                       (cond ((< sum1 sum2)
                              (format-win (cdr pl-2)))
                             ((> sum1 sum2)
                              (format-win (cdr pl-1)))
                             (t (print "Победила дружба")))))))))
   )

(kosti)
```

1.7 Номер 7

Написать функцию, которая по своему списку-аргументу lst определяет, является ли он палиндромом.

```
(defun check(lst) (equal lst (reverse lst)))

(check '(1 2 3 2 1)) ; T
(check '(1 2 3 2 3)) ; NIL
(check '())          ; T
(check '(1))         ; T
(check '(1 2))       ; NIL
```

1.8 Номер 8

Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна.столица), и возвращают по стране – столицу, а по столице – страну.

```
(defun my_assoc_1(key table)
    (cond ((equal (caar table) key) (cdar table))
          ((equal (caadr table) key) (cdadr table))
          ((equal (caaddr table) key) (cdaddr table))
          ((equal (car (cadddr table)) key) (cdr (cadddr table))))
   )

(defun my_assoc_2(key table)
    (cdr (assoc key
                table
                :test #'equal)))

(defun my_rassoc_1(key table)
    (cond ((equal (cdar table) key) (caar table))
          ((equal (cdadr table) key) (caadr table))
          ((equal (cdaddr table) key) (caaddr table))
          ((equal (cdr (cadddr table)) key) (car (cadddr table))))
   )

(defun my_rassoc_2(key table)
    (car (rassoc key
                 table
                 :test #'equal)))

(setf table '((Россия . Москва)
              (США . Вашингтон)
              (Китай . Пекин)
              (Германия . Берлин)))

(my_assoc_1 'Германия table) ; БЕРЛИН
(my_assoc_2 'Германия table) ; БЕРЛИН
(my_assoc_1 'Франция table)  ; NIL
(my_assoc_2 'Франция table)  ; NIL

(my_rassoc_1 'Москва table)  ; РОССИЯ
(my_rassoc_2 'Москва table)  ; РОССИЯ
(my_rassoc_1 'Париж table)   ; NIL
(my_rassoc_2 'Париж table)   ; NIL
```

1.9 Номер 9

Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда а) все элементы списка – числа, б) элементы списка – любые объекты.

```
; все элементы - числа
(defun allnum (arr number)
(rplaca arr
        (* number
           (car arr))))

; элементы произвольные
(defun rand(arr number)
  (cond ((numberp (car arr))
           (rplaca arr
                   (* number (car arr))))
        ((numberp (cadr arr))
           (rplaca (cdr arr)
                   (* number (cadr arr))))
        ((numberp (caddr arr))
           (rplaca (cddr arr)
                   (* number (caddr arr))))
        (t arr)))

(setf l1 '(1 2 3)
      l2 '((a) (2) 3)
      l3 '())

(allnum l1 12)
(rand l2 12)
(rand l3 12)

(print l1)  ; (12 2 3)
(print l2)  ; (12 2 3)
(print l3)  ; NIL
```