



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**Отчет по лабораторной работе №6
по дисциплине «Функциональное и логическое
программирование»**

Тема Использование управляющих структур, работа со списками

Студент Романов А.В.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватель Толпинская Н.Б., Строганов Ю. В.

Задание 1

Постановка задачи

Чем принципиально отличаются функции `cons`, `list`, `append`?

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

Каковы результаты следующих выражений?

```
1 (cons lst1 lst2)
2 (list lst1 lst2)
3 (append lst1 lst2)
```

Решение

1. `((A B) C D)`
2. `((A B) (C D))`
3. `(A B C D)`

Задание №2

Постановка задачи

Каковы результаты вычисления следующих выражений?

```
1 (reverse ())
2 (last ())
3 (reverse '(a))
4 (last '(a))
5 (reverse '((a b c)))
6 (last '((a b c)))
```

Решение

1. `Nil`
2. `Nil`
3. `(a)`
4. `(a)`
5. `((a b c))`
6. `((a b c))`

Задание №3

Постановка задачи

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

Решение

Листинг 1: Решение задания №3

```
1 (defun last (lst)
2   (if (cdr lst)
3       (last-elem (cdr lst))
4       (car lst))
5 )
6
7 (defun last-reduce (lst)
8   (reduce #'(lambda (acc current) current) lst)
9 )
```

Задание №4

Постановка задачи

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

Решение

Листинг 2: Решение задания №3

```
1 (defun init (lst)
2   (defun wrapper (lst acc)
3     (if (cdr lst)
4         (wrapper (cdr lst) (append acc (cons (car lst) Nil)))
5         acc)
6   )
7   (wrapper lst ())
8 )
9
10 (defun init-reverse (lst)
11   (reverse (cdr (reverse lst)))
12 )
```

Задание №5

Постановка задачи

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 получает 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`.

Решение

Листинг 3: Решение задания №5

```
1 (defun random-cube-value ()
2   (list (random 7) (random 7)))
3
4 (defun dices-sum (pair)
5   (+ (car pair) (car (cdr pair))))
6
7 (defun check-absolute-win (sum)
8   (or (= sum 7) (= sum 11)))
9
10 (defun check-rerun (pair)
11   (let* ((fst (car pair))
12          (snd (car (cdr pair))))
13     (or (= fst snd 1) (= fst snd 6)))
14 )
15
16 (defun dices ()
17   (let* ((fst-dices-pair (random-cube-value))
18          (fst-dices-sum (dices-sum fst-dices-pair))
19          (format T "Player one dices: ~a.~%" fst-dices-pair)
20          (cond ((check-absolute-win fst-dices-sum)
21                 (format T "Player one win!~%"))
22                ((check-rerun fst-dices-pair)
23                 (format T "Rerun!~%") (dices))
24                (t (let* ((snd-dices-pair (random-cube-value))
25                           (snd-dices-sum (dices-sum snd-dices-pair))
26                           (format T "Player two dices: ~a.~%" snd-dices-pair)
27                           (cond ((check-absolute-win snd-dices-sum)
28                                  (format T "Player two win!~%"))
29                                  ((> fst-dices-sum snd-dices-sum)
30                                   (format T "Player one win!~%"))
31                                  (t
32                                   (format T "Player two win!~%"))))))))
32   )
33 )
```

Контрольные вопросы

Вопрос 1. Структуроразрушающие и не разрушающие структуру списка функции

Ответ. Не разрушающие структуру списка функции. Данные функции не меняют сам объект-аргумент, а создают копию. К таким функциям относятся: `append`, `reverse`, `last`, `nth`, `nthcdr`, `length`, `remove`, `subst` и прочие.

Структуроразрушающие функции. Данные функции меняют сам объект-аргумент, невозможно вернуться к исходному списку. Чаще всего такие функции начинаются с префикса `n-`. К таким функциям относятся: `nreverse`, `nconc`, `nsubst` и прочие.

Вопрос 2. Отличие в работе функций `cons`, `list`, `append` и в их результате.

Ответ. Функция `cons` — чисто математическая, конструирует списковую ячейку, которая может вовсе и не быть списком. Является списком только в том случае, если вторым аргументом передан список.

Функция `list` — форма, принимает произвольное количество аргументов и конструирует из них список. Результат — всегда список. При нуле аргументов возвращает пустой список.

Функция `append` — форма, принимает на вход произвольное количество аргументов и для всех аргументов, кроме последнего, создает копию, ссылая при этом последний элемент каждого списка-аргумента на первый элемент следующего по порядку списка-аргумента. Копирование для последнего не делается в целях эффективности.