|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

**Лабораторная работа № 6**

|  |  |
| --- | --- |
| **По дисциплине**: Функциональное и логическое программирование  **Тема: «**Использование управляющих структур, работа со списками**»**  **Студент** Унтилова А.О.  **Группа** ИУ7-66  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Толпинская Н.Б., Строганов Ю.В. |  |

Москва.

2021 г.

1. **Практическая часть**

**Задание 1.**

Пусть (setf lst ‘(a b)) (setf ls2 ‘(c d)) Каковы результаты вычисления следующих выражений?

1. (cons ls1 ls2)

Результат: ((A B) C D)

1. (list ls1 ls2)

Результат: ((A B) (C D))

1. (append ls1 ls2)

Результат (A B C D)

**Задание 2**. Каковы результаты вычисления следующих выражений?

1. (reverse ())

Результат: Nil

1. (last ())

Результат: Nil

1. (reverse ‘(a))

Результат: (A)

1. (last ‘(a))

Результат: (A)

1. (reverse ‘((a b c)))

Результат: ((a b c))

1. (last ‘((a b c)))

Результат: ((a b c))

**Задание 3.** Написать, по крайней мере, два варианта функции, которая возвращает последний элемент своего списка-аргумента.

1. (defun last\_1(lst)(car (last lst)))
2. (defun last\_2(lst)(car (reverse lst)))
3. (defun last\_3(lst) (first (reverse lst)))

**Задание 4**. Написать, по крайней мере, два варианта функции, которая возвращает свой список-аргумент без последнего элемента.

1. (defun lst\_1(lst)

(reverse (cdr (reverse lst))))

1. (defun myf1 (lst)

(if (cdr lst)

(cons (car lst) (myf1 (cdr lst)))

nil) )

**Задание 5**. 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.

|  |
| --- |
| (defun roll\_dice ()  (list (+ 1 (random 6)) (+ 1 (random 6)))  )  (defun check\_abs\_win (dice)  (or (equal 7 (+ (first dice) (second dice)))  (equal 11 (+ (first dice) (second dice)))  )  )  (defun check\_abs\_sum (sum\_dice)  (or (equal 7 sum\_dice)  (equal 11 sum\_dice))  )  (defun check\_reroll (dice)  (or (equal dice '(1 1)) (equal dice '(6 6)))  )  (defun turn\_first()  (let ((dice\_first (roll\_dice)))  (print "first throw")  (print dice\_first)  (if (cond ((check\_abs\_win dice\_first) (print "first win"))  )  nil  (cond ((check\_reroll dice\_first) (turn\_first))  (T (+ (first dice\_first) (second dice\_first)))  )  )  )  )  (defun turn\_second()  (let ((dice\_second (roll\_dice)))  (print "second throw")  (print dice\_second)  (if (cond ((check\_abs\_win dice\_second) (print "second win"))  )  nil  (+ (first dice\_second) (second dice\_second))  )  )  )    (defun game ()  (let ((first\_sum (turn\_first)))  (if first\_sum  (let ((second\_sum (turn\_second)))  (if (and first\_sum second\_sum)  (cond ((> first\_sum second\_sum) (print "first win"))  ((< first\_sum second\_sum) (print "second win"))  (T (print "dead heat"))  )  )  )  )  )  nil  ) |

**Теоретические вопросы:**

1. **Структуроразрушающие и не разрушающие структуру списка функции**

Функции делятся на:

* Функции, разрушающие структуру списка (после работы функции теряется возможность работать с теми данными, которые были в списках)
* Не разрушающие структуру списка функции (работают медленнее).
* **nconc** – объединяет спсики, не создает копии (теряется возможность работать со списками отдельно)
* **reverse** - обращает список (одноаргументна) (не может обрабатывать атом) (работает с копией) (разворачивают указатели только верхнего уровня списка)
* **nreverse** - не создает копию (работает по верхнему уровню спусковых ячеек)
* **last** - возвращает последнюю списковую ячейку верхнего уровня, может работать и с точечной парой
* **nth** - голову n-й списковой ячейки верхнего уровня (нумерация с 0)
* **nthcdr** - остаток списка начиная n-го списковой ячейки верхнего уровня (нумерация с 0) все стандартные функции работают только по верхнему уровню списка
* **lengs** - количество списковых ячеек верхнего уровня списка (может работать с любой структурой, которая в памяти представляется как списковая ячейка)
* **remove** – удаление из списка, не разрушает структуру; удалить = найти путем сравнения и потом удалить.

(remove el lst)

(remove ‘a ‘(bac)) -> (bc)

(remove ‘(a b) ‘((a b) c d)) -> ((a b) c d)

если надо удалить исходный структурированный список, то система должна использовать в качестве сравнения не eq, а equal:

(remove ‘(a b) ‘((a b) c d): test #’equal) -> (c d)

* **rplaca** - замена по указателю car (replaca lst el), car указатель исходного списка переставляется на второй аргумент.
* **rplacd** - замена указателя cdr (replacd lst el)
* subst – заменяет в исходном списке старые элементы на новые., не разрушает структуру

(subst new old lst)

* **nsubst** – разрушает структуру

1. **Отличие в работе функций cons, list, append и в их результате**

**append** – многоаргументная функция, объединяет спсики.. Не разрушает структуру. Является формой.

append создает копии всех аргументов кроме последнего. Копии создаются в списковых ячейках.

append(lst1 lst2)

(setf lst3(append lst1 lst2))

lst1 – символьный атом, у которого мб установлено значение.

В результате появится возможность работать с результатом работы функции, но при этом есть возможность работать с исходными данными списков lst1, lst2.

**list и cons** являются функциями создания списков.

Функция **cons** создает списочную ячейку и устанавливает два указателя на аргументы. Если значением первого аргумента является атом, а второго - список, то результатом функции **cons** будет **список**, голова которого есть значение первого аргумента, а хвост - значение второго.

Функция **list** принимает переменное число аргументов и возвращает список, элементы которого – переданные в функцию аргументы. Функция list не является базовой функцией. Относится к функциям-конструкторам. Может иметь произвольное количество аргументов, создает столько списочных ячеек, сколько аргументов.