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 №2 по дисциплине «Функциональное и логическое программирование»

Тема Определение функций пользователя

Студент Завойских Е.В.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Москва — 2023 г.

1 Теоретические вопросы

1.1 Базис Lisp

Базис — это минимальный набор конструкций языка, на основе которого могут быть построены остальные.

Базис Lisp:

- атомы;
- структуры;
- базовые функции и базовые функционалы.

1.2 Классификация функций

- базисные функции;
- функции ядра;
- пользовательские функции.

Классификация функций по аргументам и поведению:

- чистые функции (фиксированное количество аргументов, для определенного набора аргументов один фиксированный результат);
- формы (переменное количество аргументов или аргументы обрабатываются по-разному);
- функционалы (принимают функцию в качестве аргумента или возвращают функцию).

Классификация функций по назначению:

- селекторы (car, cdr);
- конструкторы (cons, list);
- предикаты (atom, numberp);
- функции сравнения (eq, eql, equal).

1.3 Способы создания функций

- с использованием λ-нотации (функции без имени)

  λ-выражение: (lambda λ-список тело_функции), где λ-список — формальные параметры функции.

  Вызов такой функции осуществляется следующим способом: (λ-выражение фактические параметры).

  Вычисление функций без имени может быть выполнено с использованием функционала apply: (apply λ-выражение список_фактических_параметров); или с использованием функционала funcall: (funcall λ-выражение фактические_параметры).

- с использованием макро-определения defun:

  (defun имя_функции λ-выражение),

  или в облегченной форме:

  (defun имя_функции $(x_1, x_2, ..., x_k)$ тело_функции), где $(x_1, x_2, ..., x_k)$ — список аргументов.

  В качестве имени функции выступает символьный атом. Вызов именованной функции осуществляется следующим образом: (имя_функции фактические параметры).

1.4 Функции car и cdr, eq, eql, equal, equalp

Функции car и cdr принимают точечную пару или список в качестве аргумента. Функция car возвращает голову (значение по первому указателю списковой ячейки). Функция cdr возвращает хвост (значение по второму указателю списковой ячейки).

eq, eql, equal, equalp — функции сравнения:

- eq сравнивает два символьных атома. Возвращает T, когда значением одного из аргументов является атом, и одновременно значения аргументов равны. В ином случае возвращает Nil;

- eql сравнивает символьные атомы и числа одного типа. Например, (eql 3 3) -> T, (eql 3 3.0) -> Nil;

- equal работает идентично eql + сравнивает списки (считая списки эквивалентными, если они рекурсивно, согласно тому же equal, имеют одинаковую структуру и содержимое);

- equalp сравнивает символьные атомы, числа разных типов (и (equalp 1 1), и (equalp 1 1.0) вернет T) и списки.

1.5 Назначение и отличие в работе cons и list

cons принимает 2 аргумента, создает списковую ячейку и ставит указатели на 2 аргумента, таким образом объединяя их в точечную пару.

Функция list, составляющая список из значений своих аргументов, создает столько списковых ячеек, сколько аргументов ей было передано. Эта функция относится к особым, поскольку у неё может быть произвольное число аргументов.

Основные отличия:

- cons принимает фиксированное количество аргументов, list — произвольное.

- cons создает одну списковую ячейку, list — список.

2 Практические задания

2.1 Составить диаграмму вычисления следующих выражений:

1. (equal 3 (abs -3));
2. (equal (+ 1 2) 3);
3. (equal (* 4 7) 21);
4. (equal (* 2 3) (+ 7 2));
5. (equal (- 7 3) (* 3 2));
6. (equal (abs (- 2 4)) 3).

Решение приложено к отчету на отдельном листе.

2.2 Написать функцию, вычисляющую гипотенузу прямоугольного треугольника по заданным катетам и составить диаграмму ее вычисления

```
1 (defun hyp (a b) (sqrt (+ (* a a) (* b b))))
2 (hyp 3 4) -> 5
```

Диаграмма приложена к отчету на отдельном листе.

2.3 Каковы результаты вычисления следующих выражений? (объяснить возможную ошибку и варианты ее устранения)

1. (list 'a c)

   ошибка: variable C has no value

   решение: добавление ' перед c

2. (cons 'a (b c))
   ошибка: undefined function B
   решение: добавление ' перед (b c)
3. (cons 'a '(b c))
   (A B C)
4. (caddr (1 2 3 4 5))
   ошибка: 1 is not a function name; try using a symbol instead
   решение: добавление ' перед (1 2 3 4 5)
5. (cons 'a 'b 'c)
   ошибка: too many arguments given to CONS: (CONS 'A 'B 'C)
   решение: замена 'b 'c на '(b c)
6. (list 'a (b c))
   ошибка: undefined function B
   решение: добавление ' перед (b c)
7. (list a '(b c))
   ошибка: variable A has no value
   решение: добавление ' перед a
8. (list (+ 1 '(length '(1 2 3))))
   ошибка: (LENGTH '(1 2 3)) is not a number
   решение: убрать ' перед (length '(1 2 3))

2.4 Написать функцию *longer_then* от двух списков-аргументов, которая возвращает T, если первый аргумент имеет большую длину

```
1 (defun longer_then (l1 l2) (> (length l1) (length l2)))
```

2.5 Каковы результаты вычисления следующих выражений?

1. (cons 3 (list 5 6)) -> (3 5 6)
2. (cons 3 '(list 5 6)) -> (3 list 5 6)
3. (list 3 'from 9 'lives (- 9 3)) -> (3 from 9 lives 6)
4. (+ (length for 2 too) (car '(21 22 23))) -> variable FOR has no value
5. (cdr '(cons is short for ans)) -> (is short for ans)
6. (car (list one two)) -> variable ONE has no value
7. (car (list 'one 'two)) -> one

2.6 Дана функция (defun mystery (x) (list (second x) (first x))). Какие результаты вычисления следующих выражений?

1. (mystery (one two)) -> undefined function ONE
2. (mystery (last one two)) -> variable ONE has no value
3. (mystery free) -> variable FREE has no value
4. (mystery one 'two) -> variable ONE has no value

2.7 Написать функцию, которая переводит температуру в системе Фаренгейта в температуру по Цельсию (defun f_to_c (temp)...)

```
1 (defun f-to-c (temp) (* (/ 5 9) (- temp 32.0)))
2 (f-to-c 451) -> 232.77779
```

2.8 Что получится при вычислении каждого из выражений?

1. (list 'cons t NIL) -> (cons t NIL)
2. (eval (list 'cons t NIL)) -> (t)
3. (eval (eval (list 'cons t NIL))) -> undefined function T
4. (apply cons "(t NIL)) -> bad syntax for complex number: CONS
5. (eval NIL) -> NIL
6. (list 'eval NIL) -> (eval NIL)
7. (eval (list 'eval NIL)) -> NIL