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 №3 по курсу "Функциональное и логическое программирование"

Студент Пересторонин П.Г.

Группа ИУ7-63Б

Оценка _____

Преподаватель Толпинская Н. Б.

Москва — 2021 г.

Оглавление

1 Задания

1.1 Составить диаграмму вычисления следующих выражений.

```
(equal 3 (abs -3))
```

```
(equal (+ 1 2) 3)
```

```
(equal (* 4 7) 21)
```

```
(equal (* 2 3) (+ 7 2))
```

```
(equal (- 7 3) (* 3 2))
```

```
(equal (abs (- 2 4)) 3)
```

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

```
1 (defun hypotenuse(a b) (sqrt (+ (* a a) (* b b))))
```

1.3 Написать функцию, вычисляющую объем параллелепипеда по 3-м его сторонам, и составить диаграмму ее вычисления.

```
1 (defun volume(a b h) (* a b h))
```

1.4 Каковы результаты вычисления следующих выражений?

```
(list 'a 'b c)
```

Результат: несвязанная переменная `c` (у переменной `c` не стоит блокирование вычисления и при попытке вычислить случается ошибка, потому что интерпретатор не может найти информацию о данном символе в `package`)

```
(cons 'a (b c))
```

Результат: несвязанная переменная c (конструкция `(b c)` рассматривается интерпретатором, как вычислимое выражение, для вычисления которого нужно вызвать функцию `b` с аргументом `c`, поэтому сначала интерпретатор вычисляет `c`, а у переменной `c` не стоит блокирование вычисления и при попытке вычислить случается ошибка, потому что интерпретатор не может найти информацию о данном символе в `package`)

```
(cons 'a '(b c))
```

Результат: `(a b c)`.

```
(caddr (1 2 3 4 5))
```

Результат: недопустимый вызов функции (ввиду отсутствия блокировки вычисления `(1 2 3 4 5)` рассматривается как список, на первом месте которого стоит символ, соответствующий имени функции, которую нужно вызвать с остальными элементами списка в качестве аргументов функции (`1` — не функция))

```
(cons 'a 'b 'c)
```

Результат: некорректное количество аргументов (функция `cons` ожидает 2 аргумента, а передано 3)

```
(list 'a (b c))
```

Результат: несвязанная переменная c (конструкция `(b c)` рассматривается интерпретатором, как вычислимое выражение, для вычисления которого нужно вызвать функцию `b` с аргументом `c`, поэтому сначала интерпретатор вычисляет `c`, а у переменной `c` не стоит блокирование вычисления и при попытке вычислить случается ошибка, потому что интерпретатор не может найти информацию о данном символе в `package`)

```
(list a '(b c))
```

Результат: несвязанная переменная (у переменной `a` не стоит блокирование вычисления и при попытке вычислить случается ошибка, потому что интерпретатор не может найти информацию о данном символе в `package`)

```
(list (+ 1 '(length '(1 2 3))))
```

Результат: ошибка вычисления (чистая математическая функция ожидает на вход `NUMBER`, а получена форма, вычисление которой заблокировано).

1.5 Написать функцию `longer-then` от двух списков-аргументов, которая возвращает T, если первый аргумент имеет большую длину.

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

1.6 Каковы результаты вычисления следующих выражений

```
(cons 3 (list 5 6))
```

Результат: `(3 5 6)`

```
(list 3 'from 9 'gives (- 9 3))
```

Результат: `(3 from 9 gives 6)`

```
(+ (length '(1 foo 2 too)) (car '(21 22 23)))
```

Результат: 25 (4 + 21)

```
(cdr '(cons is short for ans))
```

Результат: `(is short for ans)`

```
(car (list one two))
```

Результат: несвязанная переменная `one` (у переменной `one` не стоит блокирование вычисления и при попытке вычислить случается ошибка, потому что интерпретатор не может найти информацию о данном символе в `package`)

```
(cons 3 '(list 5 6))
```

Результат: `(3 list 5 6)`

```
(car (list 'one 'two))
```

Результат: `one`

1.7 Дана функция mystery, найти результаты выражений.

Функция:

```
(defun mystery (x) (list (second x) (first x)))
```

Выражения:

```
(mystery '(one two))
```

Результат: `(two one)`

```
(mystery 'free)
```

Результат: функция `second` применима только к значениям типа `list`, а `free` – не `list`.

```
(mystery (last 'one 'two))
```

Результат: ошибка: `one` – не `list` (а функция `принимает один аргумент типа list`)

```
(mystery 'one 'two)
```

Результат: ошибка: неверное количество аргументов.

2 Ответы на вопросы к лабораторной работе

2.1 Базис

Базис состоит из:

1. структуры, атомы;
2. встроенные (примитивные) функции (`atom`, `eq`, `cons`, `car`, `cdr`);
3. специальные функции, управляющие обработкой структур, представляющих вычислимые выражения (`quote`, `cond`, `lambda`, `label`, `eval`).

Таким образом, функции, входящие в базис:

- `atom` — функция определения, является ли объект атомом (возвращает `T`, если да, иначе – `Nil`);
- `eq` — функция проверки атомов на равенство (возвращает `T`, если равны, иначе – `Nil`)
- `cons` — функция создания точечной пары;
- `car` — функция получения первого элемента точечной пары;
- `cdr` — функция получения второго элемента точечной пары;
- `cond` — функция, позволяющая организовать ветвление (чаще всего используется в частичных функциях и имеет вид: `(cond (p1 e1) (p2 e2) ... (pn en))`, где `pi` – предикат, а `ei` – форма, соответствующая предикату `ei`)
- `quote` — функция блокировки вычислений, аргумент не вычисляется;
- `eval` — функция-интерпретатор, противоположность функции `quote`, пытается вычислить аргумент;
- `lambda`;
- `label`.

2.2 Классификация функций

Функции в Lisp классифицируют следующим образом:

- чистые математические функции;
- рекурсивные функции;
- специальные функции — формы (сегодня 2 аргумента, завтра - 5);
- псевдофункции (создают эффект на внешнем устройстве);
- функции с вариативными значениями, из которых выбирается 1;
- функции высших порядков — функционал: используется для синтаксического управления программ (абстракция языка).

По назначению функции разделяются следующим образом:

1. конструкторы — создают значение (`cons`, например);
2. селекторы — получают доступ по адресу (`car`, `cdr`);
3. предикаты — возвращают `Nil`, `T`.

2.3 Список, представление и интерпретация списков

Список — частный случай S-выражения, структура данных, может быть пустым или непустым, непустой список содержит голову (в случае Lisp хранящую произвольное S-выражение) и хвост, который также должен является списком.

Непустой список в Lisp представлен списковой ячейкой, точечная пара, первый элемент которой - произвольное S-выражение, второй - список; пустой - `Nil`.

`Список ::= <пустой список> | <непустой список>`, где
`<пустой список> ::= () | Nil`,

<непустой список> ::= (<S-выражение>. <список>),

В оперативной памяти списковая ячейка представляется бинарным узлом: 2 указателя, один из которых на голову списка, второй - на хвост.

2.4 Функции car и cdr

car — функция получения первого элемента точечной пары.

Примеры:

S-выражение	Результат выполнения car
(A . B)	A
((A . B) . C)	(A . B)
A	ошибка

cdr — функция получения второго элемента точечной пары.

S-выражение	Результат выполнения cdr
(A . B)	B
(A . (B . C))	(B . C)
A	ошибка

2.5 Назначение и отличие list от cons

cons — функция конструирования точечной пары, на вход получает 2 значения и делает из них точечную пару.

list — функция конструирования списка. На вход получает произвольное количество элементов и делает из них список.

Вызовы (list 1 2 3 4) и (cons 1 (cons 2 (cons 3 (cons 4 Nil)))) эквивалентны, то есть дают одинаковый результат.