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»

Отчёт по лабораторной работе №3 по курсу «Функциональное и логическое программирование»

Тема Работа интерпретатора Lisp

Студент Богаченко А. Е.

Группа ИУ7-66Б

Оценка (баллы) _____

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

Москва — 2021 г.

Задание 1

Составить диаграмму вычисления следующих выражений:

Листинг 1 – Задание 1

```
(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. (equal 3 (abs -3))
     3 вычисляется как 3
     (abs -3)
       -3 вычисляется как -3
       abs применяется к -3
       3
     equal применяется к 3 и 3
     T

2. (equal (+ 1 2) 3)
     (+ 1 2)
       1 вычисляется как 1
       2 вычисляется как 2
       + применяется к 1 и 2
       3
     3 вычисляется как 3
     equal применяется к 3 и 3
     T
```

```
3. (equal (* 4 7) 21)
     (* 4 7)
       4 вычисляется как 4
       7 вычисляется как 7
       * применяется к 4 и 7
       28
     21 вычисляется как 21
     equal применяется к 28 и 21
     Nil

4. (equal (* 2 3) (+ 7 2))
     (* 2 3)
       2 вычисляется как 2
       3 вычисляется как 3
       * применяется к 2 и 3
       6
     (+ 7 2)
       7 вычисляется как 7
       2 вычисляется как 2
       + применяется к 7 и 2
       9
     equal применяется к 6 и 9
     Nil
```

```
5. (equal (- 7 3) (* 3 2))
     (- 7 3)
       7 вычисляется как 7
       3 вычисляется как 3
       - применяется к 7 и 3
       4
     (* 3 2)
       3 вычисляется как 3
       2 вычисляется как 2
       * применяется к 3 и 2
       6
     equal применяется к 4 и 6
     Nil

6. (equal (abs (- 2 4)) 3)
     (abs (- 2 4))
       (- 2 4)
         2 вычисляется как 2
         4 вычисляется как 4
         - применяется к 2 и 4
         -2
       abs применяется к -2
       2
     3 вычисляется как 3
     equal применяется к 2 и 3
     Nil
```

Задание 2

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

Листинг 2 – Задание 2

```
1 (defun calc-triangle-hyp (a b)
2   "Task 2"
```

```
3   (let* ((a-square (* a a))
4         (b-square (* b b))
5         (square-sum (+ a-square b-square)))
6     (sqrt square-sum)))
```

```
(calc-triangle-hyp 3 4)

  (a-square (* 3 3))
    (* 3 3)
      3 вычисляется как 3
      3 вычисляется как 3
      * применяется к 3 и 3
      9
    привязка 9 к a-square
  (b-square (* 4 4))
    (* 4 4)
      4 вычисляется как 4
      4 вычисляется как 4
      * применяется к 4 и 4
      16
    привязка 16 к b-square
  (square-sum (+ a-square b-square)))
    (+ a-square b-square)
      + применяется к a-square и b-square
      25
    привязка 25 к square-sum
  (sqrt square-sum)
    sqrt применяется к square-sum
    5
```

Задание 3

Написать функцию, вычисляющую объём параллелепипеда по 3-м его сторонам и составить диаграмму её вычисления.

Листинг 3 – Задание 3

```
(defun calc-par-vol (a b h)
  "Task 3"
  (* a b h))
```

```
(calc-par-vol 1 2 3)

  (* 1 2 3)
    1 вычисляется как 1
    2 вычисляется как 2
    3 вычисляется как 3
    * приминяется к 1, 2, 3
    6
```

Задание 4

Каковы результаты вычисления следующих выражений?

Листинг 4 – Задание 4

```
(list 'a 'b c) ; -> the variable C is unbound
(cons 'a (b c)) ; -> the variable C is unbound
(cons 'a '(b c)) ; -> (a b c)
(caddr (1 2 3 4 5)) ; -> illegal function call
(cons 'a 'b 'c) ; -> invalid number of arguments: 3
(list 'a (b c)) ; -> the variable C is unbound
(list a '(b c)) ; -> the variable A is unbound
(list (+ 1 '(length '(1 2 3)))) ; -> the value (LENGTH '(1 2 3)) is not of type NUMBER
```

Задание 5

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

Листинг 5 – Задание 5

```
(defun longer-than (a b)
  "Task 5"
  (let ((len-a (length a))
```

```
4       (len-b (length b)))
5   (> len-a len-b)))
```

Задание 6

Каковы результаты вычисления следующих выражений?

Листинг 6 – Задание 6

```
1  (cons 3 (list 5 6)) ; -> (3 5 6)
2  (list 3 'from 9 'gives (- 9 3)) ; -> (3 from 9 gives 6)
3  (+ (length '(1 foo 2 too)) (car '(21 22 23))) ; -> 25
4  (cdr '(cons is short for ans)) ; -> (is short for ans)
5  (car (list one two)) ; -> variable ONE is unbound
6  (cons 3 '(list 5 6)) ; -> (3 list 5 6)
7  (car (list 'one 'two)) ; -> one
```

Задание 7

Дана функция

Листинг 7 – mystery

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

Каковы результаты вычисления следующих выражений?

Листинг 8 – Задание 7

```
1  (mystery '(one two)) ; -> (two one)
2  (mystery 'free) ; -> the value FREE is not of type LIST
3  (mystery (last 'one 'two)) ; -> the value ONE is not of type LIST when binding LIST
4  (mystery 'one 'two) ; -> invalid number of arguments: 2
```

Задание 8

Написать функцию, которая переводит температуру в системе Фаренгейта в температуру по Цельсию. $f = \frac{9}{5} \cdot c + 32$

Листинг 9 – Задание 8

```
(defun f-to-c (temp)
  "Task 7"
  (float (* (- temp 32) (/ 5 9))))
```

Как бы назывался роман Р. Бредбери «+451 по Фаренгейту» в системе по Цельсию?
– «232.78 по Цельсию»

Задание 9

Что получится при вычислении каждого из выражений?

Листинг 10 – Задание 9

```
  (list 'cons T Nil) ; -> (cons T Nil)
  (eval (eval (list 'cons T Nil))) ; -> undefined function
  (apply #'cons '(T Nil)) ; -> (T)
  (list 'eval Nil) ; -> (eval Nil)
  (eval (list 'cons T Nil)) ; -> (T)
  (eval Nil) ; -> Nil
  (eval (list 'eval Nil)) ; -> Nil
```

Дополнительное задание 1

Написать функцию, вычисляющую катет по заданной гипотенузе и другому катету прямоугольного треугольника и составить диаграмму её вычисления.

Листинг 11 – Дополнительное задание 1

```
(defun find-triangle-side (hyp side)
  "Extra 1"
  (let* ((hyp-square (* hyp hyp))
         (side-square (* side side))
         (square-sub (- hyp-square side-square)))
    (sqrt square-sub)))
```

```
(find-triangle-side 5 4)

  (hyp-square (* 5 5))
    (* 5 5)
      5 вычисляется как 5
      5 вычисляется как 5
      * применяется к 5 и 5
      25
    привязка 25 к hyp-square
  (side-square (* 4 4))
    (* 4 4)
      4 вычисляется как 4
      4 вычисляется как 4
      * применяется к 4 и 4
      16
    привязка 16 к side-square
  (square-sub (- hyp-square side-square)))
    (- hyp-square side-square)
      - применяется к hyp-square и side-square
      9
    привязка 9 к square-sub
  (sqrt square-sub)
    sqrt применяется к square-sub
    3
```

Контрольные вопросы

1. Базис языка Lisp

Базис языка представлен:

- структурами, атомами;
- функциями:

atom, eq, cons, car, cdr,
cond, quote, lambda, eval, label.

2. Классификация функций языка Lisp

Функции в языке Lisp:

- чистые (с фиксированным количеством аргументов) – математические функции;
- рекурсивные функции;
- специальные функции – формы (принимают произвольное количество аргументов или по разному обрабатывают аргументы);
- псевдофункции (создающие «эффект» - отображающие на экране процесс обработки данных и т. п.);
- функции с вариативными значениями, выбирающие одно значение;
- функции высших порядков – функционалы (используются для построения синтаксически управляемых программ).

3. Синтаксис элементов языка и их представление в памяти

Точечные пары ::= (<атом>, <атом>) |
(<атом>, <точечная пара>) |
(<точечная пара>, <атом>) |
(<точечная пара>, <точечная пара>)
Список ::= <пустой список> | <непустой список>), где
<пустой список> ::= () | Nil,
<непустой список> ::= (<первый элемент>, <хвост>) ,
<первый элемент> ::= (S-выражение),
<хвост> ::= <список>
Список – частный случай S-выражения.

Синтаксически любая структура (точечная пара или список) заключается в `()`:
`(A . B)` – точечная пара
`(A)` – список из одного элемента
Пустой список изображается как `Nil` или `()`
Непустой список может быть изображён: `(A. (B . (C ()))))` или `(A B C)`
Элементы списка могут являться списками: `((A) (B) (C))`

Любая непустая структура `Lisp` в памяти представлена списковой ячейкой, хранящей два указателя: на голову (первый элемент) и хвост (всё остальное).

4. Функции `car, cdr`

Являются базовыми функциями доступа к данным. `car` принимает точечную пару или список в качестве аргумента и возвращает первый элемент или `Nil`, `cdr` – возвращает все элементы, кроме первого или `Nil`.

5. Функции `list, cons`

Являются функциями создания списков (`cons` – базовая, `list` – нет). `cons` создаёт списочную ячейку и устанавливает два указателя на аргументы. `list` принимает переменное число аргументов и возвращает список, элементами которого являются аргументы, переданные в функцию.