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 №3 по курсу "Функциональное и логическое программирование"

Тема Работа интерпретатора Lisp

Студент Ковалец К. Э.

Группа ИУ7-63Б

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

Москва — 2022 г.

1 Практические задания

1.1 Задание 1

Написать функцию, которая принимает целое число и возвращает первое четное число, не меньшее аргумента.

Листинг 1.1 – Решение задания 1

```
(defun first-even (num)
    (if (evenp num) num (+ num 1)))

;; (FIRST-EVEN 1) -> 2
;; (FIRST-EVEN 2) -> 2
```

1.2 Задание 2

Написать функцию, которая принимает число и возвращает число того же знака, но с модулем на 1 больше модуля аргумента.

Листинг 1.2 – Решение задания 2

```
(defun module-plus (num)
    (+ num (if (> num 0) 1 -1)))

;; (MODULE-PLUS 1) -> 2
;; (MODULE-PLUS -1) -> -2
```

1.3 Задание 3

Написать функцию, которая принимает два числа и возвращает список из этих чисел, расположенный по возрастанию.

Листинг 1.3 – Решение задания 3

```
(defun ascending-list (num1 num2)
    (if (> num1 num2)
        (list num2 num1) (list num1 num2)))

;; (ASCENDING-LIST 1 2) -> (1 2)
;; (ASCENDING-LIST 2 1) -> (1 2)
```

1.4 Задание 4

Написать функцию, которая принимает три числа и возвращает T только тогда, когда первое число расположено между вторым и третьим.

Листинг 1.4 – Решение задания 4

```
(defun between (num1 num2 num3)
    (or (and (> num1 num2) (< num1 num3))
        (and (< num1 num2) (> num1 num3))))

;; (BETWEEN 2 1 3) -> T
;; (BETWEEN 2 3 1) -> T
;; (BETWEEN 1 2 3) -> NIL
;; (BETWEEN 4 3 2) -> NIL
```

1.5 Задание 5

Каков результат вычисления следующих выражений?

Листинг 1.5 – Решение задания 5

```
(and 'fee 'fie 'foe)          ;; FOE
(or nil 'fie 'foe)            ;; FIE
(and (equal 'abc 'abc) 'yes) ;; YES
(or 'fee 'fie 'foe)           ;; FEE
(and nil 'fie 'foe)           ;; NIL
(or (equal 'abc 'abc) 'yes)   ;; T
```

1.6 Задание 6

Написать предикат, который принимает два числа-аргумента и возвращает T, если первое число не меньше второго.

Листинг 1.6 – Решение задания 6

```
(defun predicate (num1 num2)
    (>= num1 num2))

;; (PREDICATE 1 2) -> NIL
;; (PREDICATE 2 1) -> T
```

1.7 Задание 7

Какой из следующих двух вариантов предиката ошибочен и почему?

- **numberp** — возвращает `T`, если значение аргумента — числовой атом, `Nil` иначе;
- **plusp** — возвращает `T`, если аргумент больше нуля, `Nil` иначе.

Листинг 1.7 – Решение задания 7

```
(defun pred1 (x)
    (and (numberp x) (plusp x))) ;; OK

(defun pred2 (x)
    (and (plusp x) (numberp x)))  ;; ERROR

;; (PRED1 1) -> T
;; (PRED2 1) -> T

;; (PRED1 'Hello) -> NIL
;; (PRED2 'Hello) -> The value HELLO is not of type NUMBER
```

Во втором случае при попытке проверить, является ли аргумент больше нуля, может возникнуть ошибка, если аргумент не является числовым атомом, так как при вызове функции `and` аргументы вычисляются слева направо до первого вернувшего `Nil`

1.8 Задание 8

Решить задачу 4, используя для ее решения конструкции IF, COND, AND/OR. Написать функцию, которая принимает три числа и возвращает T только тогда, когда первое число расположено между вторым и третьим.

Листинг 1.8 – Решение задания 8

```
;; С использованием IF

(defun between (num1 num2 num3)
    (if (> num1 num2)
        (< num1 num3)

        (if (< num1 num2)
            (> num1 num3))))

;; С использованием COND

(defun between (num1 num2 num3)
    (cond ((> num1 num2) (< num1 num3))
          ((< num1 num2) (> num1 num3))))

;; С использованием AND/OR

(defun between (num1 num2 num3)
    (or (and (> num1 num2) (< num1 num3))
        (and (< num1 num2) (> num1 num3))))

;; (BETWEEN 2 1 3) -> T
;; (BETWEEN 2 3 1) -> T
;; (BETWEEN 1 2 3) -> NIL
;; (BETWEEN 4 3 2) -> NIL
```

1.9 Задание 9

Переписать функцию how-alike, приведенную в лекции и использующую COND, используя только конструкции IF, AND/OR.

Листинг 1.9 – Функция how-alike из лекции

```
(defun how_alike(x y)
    (cond ((or  (= x y) (equal x y)) 'the_same)
          ((and (oddp x) (oddp y))   'both_odd)
          ((and (evenp x) (evenp y)) 'both_even)
          (t 'difference)))
```

Листинг 1.10 – Решение задания 9

```
;; С использованием IF

(defun how_alike(x y)
    (if (or  (= x y) (equal x y)) 'the_same
    (if (and (oddp x) (oddp y))   'both_odd
    (if (and (evenp x) (evenp y)) 'both_even 'difference))))

;; С использованием AND/OR

(defun how_alike(x y)
    (or (and (or  (= x y) (equal x y)) 'the_same)
        (and (and (oddp x) (oddp y))   'both_odd)
        (and (and (evenp x) (evenp y)) 'both_even)
        'difference))

;; (HOW_ALIKE 4 4) -> THE_SAME
;; (HOW_ALIKE 4 6) -> BOTH_EVEN
;; (HOW_ALIKE 1 3) -> BOTH_ODD
;; (HOW_ALIKE 4 5) -> DIFFERENCE
```

2 Ответы на теоретические вопросы к лабораторной работе

2.1 Базис Lisp

Базис языка – минимальный набор конструкций языка и структур данных, с помощью которых можно решить любую задачу.

Базис языка Lisp содержит:

- атомы и структуры (представляющиеся бинарными узлами);
- базовые функции и функционалы:
  - встроенные – примитивные функции (atom, eq, cons, car, cdr);
  - специальные функции и функционалы (quote, cond, lambda, eval, apply, funcall).

2.2 Классификация функций

Функции в Lisp классифицируют следующим образом:

- чистые математические функции (имеют фиксированное кол-во аргументов и один результат);
- рекурсивные функции;
- специальные функции – формы (прнимают произвольное число аргументов или по разному обрабатывают аргументы);
- псевдофункции (создают эффект на внешнем устройстве);
- функции с вариативными значениями, из которых выбирается одно;
- функции высших порядков – функционалы (используются для создания синтаксически управляемых программ).

По назначению функции разделяются следующим образом:

- конструкторы — создают значение (cons, list);
- селекторы — получают доступ по адресу (car, cdr);
- предикаты — возвращают Nil, T.

2.3 Способы создания функций

Функцией называется правило, по которому каждому значению одного или нескольких аргументов ставится в соответствие конкретное значение результата.

- В Lisp можно определить функцию без имени с помощью λ**-выражений**. Lambda-определение безымянной функции:

  `(lambda <lambda-список> <форма>)`

  Lambda-вызов функции:

  `(<lambda-выражение> <формальные параметры>)`

- Также в Lisp можно определить функцию с именем с помощью **defun**. В таких функциях defun связывает символьный атом с Lambda-определением:

  `(defun f <lambda-выражение>)`

  Упрощенное определение:

  `(defun f(arg1, ..., argN) <формы>)`

2.4 Работа функций Cond, if, and/or

- **cond**

```
(cond (test1 body1)
      (test2 body2)
       ...
      (testN bodyN)
     [(T else-body)])
```

Список аргументов обрабатывается последовательно: вычисляется выражение `test_i`, и если не `Nil`, то вычисляется `body_i`, и работа функции завершается, если ни один тест не выполнился, то возвращается `Nil`, можно организовать ветку «`else`», явно указав в качестве `test - T`.

- **if**

```
(if test T-body F-body)
```

Работа функции `if` очевидна, с учетом, что всё что не nil, то T. Результат теста может быть как атомом (не обязательно Nil) так и списком. В зависимости от test, будет вычислен либо один либо другой аргумент.

- **and**

```
(and arg1 arg2 ... argN)
```

Функция `and` вычисляет аргументы, пока не станет очевидным результат (появится первый `nil`). Как только станет очевиден результат – возвращается последнее вычисленное значение.

- **or**

```
(or arg1 arg2 ... argN)
```

Функция `or` вычисляет аргументы, пока не станет очевидным результат (появится первый не `nil`). Как только станет очевиден результат – возвращается последнее вычисленное значение.