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 №5 по курсу "Функциональное и логическое программирование"

Тема Управляющие структуры Lisp

Студент Пересторонин П.Г.

Группа ИУ7-63Б

Оценка _____

Преподаватель Толпинская Н. Б.

Москва — 2021 г.

Оглавление

1 Задания

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

```
(defun first-even-ge (arg)
  (if (evenp arg) arg (+ arg 1)))
```

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

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

1.3 Написать функцию, которая принимает 2 число и возвращает список из этих чисел, расположенный по возрастанию

```
(defun growing-lst (a b)
  (if (< a b) (list a b) (list b a)))
```

1.4 Написать функцию, которая принимает 3 числа и возвращает T только тогда, когда первое число расположенно между вторым и третьим

```
(defun pred (a b c)
  (and (> a b) (< a c)))
```

1.5 Каков результат вычисления следующих выражений

```
(and 'fee 'fie 'foe)
```

Результат: `foe`

```
(or 'fee 'fie 'foe)
```

Результат: `fee`

```
(and (equal 'abc 'abc) 'foe)
```

Результат: `foe`

```
(or Nil 'fie 'foe)
```

Результат: `fie`

```
(and Nil 'fie 'foe)
```

Результат: `Nil`

```
(or (equal 'abc 'abc) 'foe)
```

Результат: `T`

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

```
(defun predicate-2 (a b)
  (>= a b))
```

1.7 Какой из следующих двух вариантов ошибочен и почему?

1 вариант:

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

2 вариант:

```
(defun pred2 (x)
  (and (plusp x) (numberp x)))
```

Ошибочен второй вариант, потому что функция `plusp` принимает на вход один аргумент типа `number` и проверять, является ли аргумент числом, после выполнения функции `plusp` не имеет смысла, причем аргументы, не являющиеся числами, будут вызывать ошибку, в то время как 1 вариант будет работать с любым аргументом и возвращать `T` для положительных чисел.

1.8 Решить задачу 4, используя для ее решения конструкции `if`, `cond`, `and/or`

Используя `if`:

```
(defun pred (a b c)
  (if (> a b) (< a c) Nil))
```

Используя `cond`:

```
(defun pred (a b c)
  (cond
  ((> a b)
     (cond ((< a c) T) (T Nil))
  (T Nil)))
```

Используя `and/or`:

```
(defun pred (a b c)
  (and (> a b) (< a c)))
```

1.9 Переписать функцию how-alike приведенную в лекции и использующую `cond`, используя конструкции `if`, `and/or`

Используя `cond`:

```
(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 'diff)))
```

Используя `if`:

```
(defun how-alike-if (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
        'diff))))
```

Используя `and/or`:

```
(defun how-alike-and-or (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)
        'diff))
```

2 Ответы на вопросы к лабораторной работе

2.1 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`.

2.2 Работа функций `and`, `or`, `if`, `cond`

2.2.1 Функция `and`

Синтаксис:

```
(and expression-1 expression-2 ... expression-n)
```

Функция возвращает первое expression, результат вычисления которого = `Nil`. Если все не `Nil`, то возвращается результат вычисления последнего выражения.

Примеры:

```
(and 1 Nil 2)
```

Результат: `Nil`

```
(and 1 2 3)
```

Результат: 3

2.2.2 Функция `or`

Синтаксис:

```
(or expression-1 expression-2 ... expression-n)
```

Функция возвращает первое expression, результат вычисления которого не `Nil`. Если все `Nil`, то возвращается `Nil`.

Примеры:

```
(or Nil Nil 2)
```

Результат: 2

```
(or 1 2 3)
```

Результат: 1

2.2.3 Функция `if`

Синтаксис:

```
(if condition t-expression f-expression)
```

Если вычисленный предикат не `Nil`, то выполняется `t-expression`, иначе - `f-expression`.

Примеры:

```
(if Nil 2 3)
```

Результат: 3

```
(if 0 2 3)
```

Результат: 2

2.2.4 Функция `cond`

Синтаксис:

```
(cond
  (condition-1 expression-1)
  (condition-2 expression-2)
  ...
  (condition-n expression-n))
```

По порядку вычисляются и проверяются на равенство с `Nil` предикаты. Для первого предиката, который не равен `Nil`, вычисляется находящееся с ним в списке выражение и возвращается его значение. Если все предкаты вернут `Nil`, то и `cond` вернет `Nil`.

Примеры:

```
(cond (Nil 1) (2 3))
```

Результат: 3

```
(cond (Nil 1) (Nil 2))
```

Результат: `Nil`

2.3 Способы определения функций

2.3.1 Через `defun`

Синтаксис:

```
(defun func-name (list-of-argument) function-body)
```

Пример определения:

```
(defun sqr(x) (* x x))
```

Пример вызова:

```
(sqr 2)
```

Результат: 4

2.3.2 Через `lambda`

Синтаксис:

```
(lambda (list-of-arguments) function-body)
```

Пример использования:

```
((lambda (x) (* x x)) 2)
```

Результат: 4