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 №1 по курсу "Функциональное и логическое программирование"

Студент Тэмуужин Янжинлхам

Группа ИУ7-63Б (ИУ7и-67б)

Оценка _____

Преподаватель Толпинская Н. Б.

Москва — 2022 г.

Оглавление

1 Ответы на вопросы к лабораторной работе 2

1.1 Элементы языка 2

1.2 Синтаксис элементов языка 3

1.3 Представление в памяти 3

1.4 Символ апостроф 3

1.5 Базис языка Lisp 4

1.6 Ядро языка 4

2 Задания 5

2.1 Представить следующие списки в виде списочных ячеек (№1) . 5

2.2 Используя только функции `CAR` и `CDR`, написать выражения, возвращающие: (№2) 5

2.2.1 Второй элемент 5

2.2.2 Третий элемент 5

2.2.3 Четвертый элемент 5

2.3 Что будет в результате вычисления выражений? (№3) 5

2.4 Что будет в результате вычисления выражений? (№4) 6

2.5 Написать функцию, возвращающую список... (№5) 8

1 Ответы на вопросы к лабораторной работе

1.1 Элементы языка

Элементы языка — атомы и точечные пары (структуры, которые строятся с помощью унифицированных структур - блоков памяти - бинарных узлов). Атомы бывают:

- **символы** (идентификаторы) — синтаксически представляют собой набор литер (последовательность букв и цифр, начинающаяся с буквы; могут быть связанные и несвязанные);
- **специальные символы** — используются для обозначения «логических» констант (`T`, `Nil`);
- **самоопределимые атомы** — числа, строки - последовательность символов в кавычках (`"abc"`).

Список - динамическая структура данных, которая может быть пустой или непустой. Если она не пустая, то состоит из двух элементов:

1. Головы - S-выражение.
2. Хвоста - список.

Список представляет из себя заключенную в скобки последовательность из атомов, разделенных пробелами, или списков. Любой список является программой - его нужно вычислять.

Примеры списков:

(A B C)
(1 2 3)
((A B) (C D))

1.2 Синтаксис элементов языка

`Точечная пара ::= (<атом> . <атом>) | (<точечная пара> . <атом>) | (<атом> . <точечная пара>) | (<точечная пара> . <точечная пара>)`

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

`<пустой список> ::= () | Nil`,

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

Синтаксически любая структура (точечная пара или список) заключается в круглые скобки: `(A . B)` — точечная пара. `(A)` — список из одного элемента. Непустой список — `(A . (B . (C . (D . Nil))))` или `(A B C D)` Пустой список — `Nil` или `()`.

Элементы списка могут быть списками, например — ((A (B C) (D (E)))). Таким образом, синтаксически наличие скобок является признаком структуры — списка или точечной пары.

Любая непустая структура Lisp в памяти представляется списковой ячейкой, хранящий два указателя: на голову (первый элемент) и хвост (все остальное).

1.3 Представление в памяти

1. **(A . B)**

2. **(A B)**

1.4 Символ апостроф

Особая функция quote, которая в качестве своего значения выдаёт сам аргумент, не вычисляя его, то есть блокирует вычисление своего аргумента:

`(quote e) => e`

Функция quote используется часто, поэтому допускается упрощённый способ обращения к ней с помощью апострофа

1.5 Базис языка Lisp

Базис состоит из:

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

1.6 Ядро языка

Ядро – основные действия, которые наиболее часто используются. Ядро ши ре, чем базис.

2 Задания

2.1 Представить следующие списки в виде списочных ячеек (№1)

Решение приложено к отчету на отдельном листе.

2.2 Используя только функции CAR и CDR, написать выражения, возвращающие: (№2)

2.2.1 Второй элемент

```
(car (cdr '(1 2 3 4)))
```

2.2.2 Третий элемент

```
(car (cdr (cdr '(1 2 3 4))))
```

2.2.3 Четвертый элемент

```
(car (cdr (cdr (cdr '(1 2 3 4)))))
```

2.3 Что будет в результате вычисления выражений? (№3)

```
(CAADR '((blue cube) (red pyramid)))
(car (car (cdr ((blue cube) (red pyramid)))))
```

Результат: `red`.

```
(CDAR '((abc) (def) (ghi)))
(cdr (car ((abc) (def) (ghi))))
```

Результат: `Nil`

```
(CADR '((abc) (def) (ghi)))
(car (cdr ((abc) (def) (ghi))))
```

Результат: `(def)`

```
(CADDR '((abc) (def) (ghi)))
(car (cdr (cdr ((abc) (def) (ghi)))))
```

Результат: `(ghi)`

2.4 Что будет в результате вычисления выражений? (№4)

```
(list 'Fred 'and Wilma)
```

Результат: ошибка (несвязная переменная)

```
(list 'Fred '(and Wilma))
```

Результат: `(Fred (and Wilma))`

```
(cons Nil Nil)
```

Результат: `(Nil)`

```
(cons T Nil)
```

Результат: `(T)`

```
(cons Nil T)
```

Результат: `(Nil . T)`

```
(list Nil)
```

Результат: `(Nil)`

```
(list Nil)
```

Результат: `(Nil)`

```
(cons (T) Nil)
```

Результат: ошибка (функция T не определена)

```
(list '(one two) '(free temp))
```

Результат: `((one two) (free temp))`

```
(cons 'Fred '(and Wilma))
```

Результат: `(Fred and Wilma)`

```
(cons 'Fred '(Wilma))
```

Результат: `(Fred Wilma)`

```
(list Nil Nil)
```

Результат: `(Nil Nil)`

```
(list T Nil)
```

Результат: `(T Nil)`

```
(list Nil T)
```

Результат: `(Nil T)`

```
(cons T (list Nil))
```

Результат: `(T Nil)`

```
(list (T) Nil)
```

Результат: ошибка (функция T не определена)

```
(cons '(one two) '(free temp))
```

Результат: `((one two) free temp)`

2.5 Написать функцию, возвращающую список... (№5)

Функция `(f ar1 ar2 ar3 ar4)`**, возвращающая** `((ar1 ar2) (ar3 ar4))`**:**

С помощью функции `list`:

```
(defun fl1(ar1 ar2 ar3 ar4)
(list (list ar1 ar2) (list ar3 ar4)))
```

С помощью функции `cons`:

```
(defun fc1(ar1 ar2 ar3 ar4) (cons
(cons ar1 (cons ar2 Nil))
(cons
(cons ar3 (cons ar4 Nil)
) Nil)
))
```

Функция `(f ar1 ar2)`, **возвращающая** `((ar1) (ar2))`:

С помощью функции `list`:

```
(defun fl2(ar1 ar2) (list (list ar1) (list ar2)))
```

С помощью функции `cons`:

```
(defun fc2(ar1 ar2) (cons
(cons ar1 Nil)
(cons (cons ar2 Nil) Nil)
))
```

Функция `(f ar1)`, **возвращающая** `(((ar1)))`:

С помощью функции `list`:

```
(defun fl3(ar1) (list (list (list ar1))))
```

С помощью функции `cons`:

```
(defun fc3(ar1) (cons (cons (cons ar1 Nil) Nil) Nil))
```