

Министерство науки и высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №1
«Исследование базовых функций языка Лисп»
по дисциплине
«МЕТОДЫ И СРЕДСТВА ИСКУССТВЕННОГО ИНТЕЛЛЕКТА»

Выполнил студент группы ИС/б-17-2-о
Горбенко К. Н.
Проверил
Сметанина Т.И.

Севастополь
2020

1 ЦЕЛЬ РАБОТЫ

Изучение технологии подготовки и выполнения Лисп-программ в выбранной интегрированной среде, исследование свойств базовых функций обработки списков, а также способов описания и вызова нерекурсивных функций в языке программирования Лисп.

2 ЗАДАНИЕ НА РАБОТУ

1. Изучить среду программирования на языке Лисп по методическим указаниям.

2. Определить на языке Лисп функцию в соответствии с вариантом задания. Определение выполнить различными способами, применяя базовые функции обработки списков (см. п. 1.2.2) и дополнительные функции (см. п.1.2.3).

3. Создать в среде программирования Лисп-проект в соответствии с методическими указаниями, содержащий подготовленные определения функций и вызовы функций при различных значениях аргументов, в том числе и недопустимых.

4. Выполнить отладку проекта.

5. Зафиксировать результаты выполнения функций и ошибки, возникающие при недопустимых значениях аргументов (в виде экранных копий).

6. Объяснить результаты и локализовать вызовы функции, порождающие ошибки.

7. Исследовать отдельно каждую базовую и дополнительную функцию, входящую в определения функций, выполненных в п.1.4.3. Привести примеры правильных и неправильных вызовов.

Задание по варианту: описать на языке Лисп функцию $f(x\ y\ z)$ от трёх аргументов, которая формирует из своих аргументов список и выполняет его обработку следующим образом.

Проверить, является ли хотя бы один из элементов списка списком. Если является, то вернуть этот список без первого элемента, иначе поменять местами первый и второй элементы исходного списка.

3 ХОД РАБОТЫ

3.1 Определения функций

Определим функции в соответствии с вариантом. Функция `f1` определена с помощью базовых функций языка LISP.

Использованы функции `car` (возвращает первый элемент списка), `cdr` (возвращает второй элемент списка), `cons` (составляет список). Для ветвления использовалась функция `if` (аналог тернарного оператора).

Функция `let` используется для создания локальных переменных и области их видимости.

```

1 (defun f1 (x y z)
2   (let ((items (cons x (cons y (cons z nil)))))
3     (if (some #'listp items)
4         (cdr items)
5         (cons
6           (car (cdr items))
7           (cons
8             (car items)
9             (cdr (cdr items))
10          )
11        )
12      )
13    )
14 )

```

Функция `f2` определена с помощью дополнительных функций. Кроме функций, перечисленных выше, были использованы: `list` – функция для компоновки списков, `some` – возвращает признак того, выполняется ли данный предикат хотя бы для одного элемента списка, `rest` – возвращает все элементы списка кроме первого, `first`, `second`, `third` – возвращают соответствующий (по счету) элемент списка.

```

1 (defun f2 (x y z)
2   (let ((items (list x y z)))
3     (if (some #'listp items)
4         (rest items)
5         (list (second items) (first items) (third items))
6       )
7     )
8 )

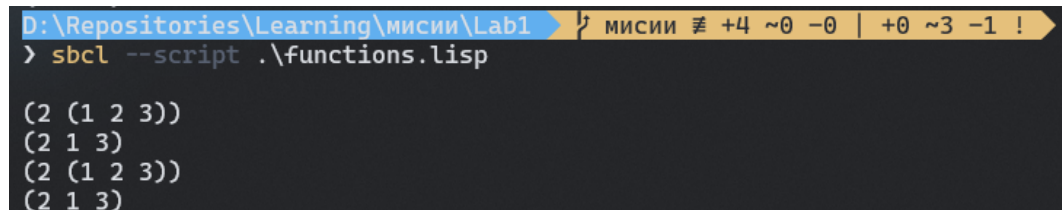
```

3.2 Выполним программу

Передавая этим функциям одинаковые параметры, проверим их работу.

```
1 (print (f1 1 2 (list 1 2 3)))
2 (print (f1 1 2 3))
3 (print (f2 1 2 (list 1 2 3)))
4 (print (f2 1 2 3))
```

Результат работы программы изображен на рисунке 1:



```
D:\Repositories\Learning\мисии\Lab1 > sbcl --script .\functions.lisp
(2 (1 2 3))
(2 1 3)
(2 (1 2 3))
(2 1 3)
```

Рисунок 1 – Результат работы программы

Из результатов работы программы видно, что обе функции работают одинаково верно.

ВЫВОДЫ

В ходе лабораторной работы были использованы базовые функции языка LISP и способы описания функций. Для описания функций используются функция `defun`.

Изучены функции:

- `car` – возвращает первый элемент списка;
- `cdr` – возвращает второй элемент списка;
- `cons` – составляет список;
- `if` – используется для организации ветвления (аналог тернарного оператора);
- `let` – используется для создания локальных переменных и области их видимости;
- `list` – создает список из параметров;
- `some` – возвращает признак того, выполняется ли данный предикат хотя бы для одного элемента списка;
- `rest` – возвращает все элементы списка кроме первого;