Отчет по лабораторной работе № 2 по курсу «Функциональное программирование»

Студентка группы М8О-307 МАИ *Довженко Анастасия*, №7 по списку
Контакты: tutkarma@gmail.com
Работа выполнена: 17.03.2019

Преподаватель: Иванов Дмитрий Анатольевич, доц. каф. 806
Отчет сдан:
Итоговая оценка:
Подпись преподавателя:

1. Тема работы

Простейшие функции работы со списками Common Lisp.

2. Цель работы

Научиться конструировать списки, находить элемент в списке, использовать схему линейной и древовидной рекурсии для обхода и реконструкции плоских списков и деревьев.

3. Задание (вариант №2.37)

Запрограммируйте рекурсивно на языке Коммон Лисп функционал `map-set (f X)`, аргументами которого являются функция одного аргумента `f` и список `X`, рассматриваемый как множество. Результатом вызова должно быть множество из результатов применения `f` к каждому из элементов `X`. В списки, представляющие множества, нет повторений, а порядок элементов не имеет значения.

4. Оборудование студента

Ноутбук Asus UX310U, процессор Intel Core i7-6500U CPU 2.50GHz x 4, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 16.04 LTS, компилятор clisp, текстовый редактор Sublime Text 3.

6. Идея, метод, алгоритм

Идея в том, чтобы рекурсивно обработать каждый элемент списка, применив к нему заданную функцию. При этом надо проверять, что этот элемент не встречался ранее.

Выполнение `map-set` происходит так: получаем первый элемент списка и применяем к нему переданную функцию, делаем рекурсивный вызов с частью списка, которая следует за первым элементом. При выходе из рекурсии соединяем обработанный первый элемент с результирующим списком, при этом проверяя, что такого элемента еще нет в списке. Если передаваемый в качестве аргумента список стал пустым, рекурсивные вызовы заканчиваются.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun map-set (func X)
    (cond ((null X) nil)
        (t (mycons (funcall func (car X)) (map-set func (cdr X))
   ))
    )
)


(defun mycons (X Y)
    (cond ((null (member X Y)) (cons X Y))
        (t Y))
)


(print (map-set #'abs '(1 2 -3 -2)))
(print (map-set #'identity (list 20 20 30 30 40 40)))
(print (map-set #'abs (list )))
(print (map-set #'abs (list 1 -1 1 -1 1 -1)))
(print (map-set #'sqrt (list 4 9 16 25 25 16 9 4)))
(print (map-set (lambda (x) (* x x)) (list 1 -2 3 -4 5 -6 7)))
(print (map-set #'car '((11 a) (22 b) (33 c) (22 b))))
```

8.2. Результаты работы

```
(1 3 2)
(20 30 40)
```

```
NIL
(1)
(5 4 3 2)
(1 4 9 16 25 36 49)
(11 33 22)
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Работа показалась мне интересной, иногда я использую функции высших порядков в других языках, но никогда не задумывалась, как они работают «под капотом».

11. Выводы

Я написала свою реализацию встроенного функционала `mapcar`. В отличии от нативного функционала, моя реализация убирает дубликаты из результирующего списка. Также узнала о таких функциях обработки списков, как `car, cdr, cons`. Стоит заметить, что `car` мог бы быть заменен на `first`, `cdr` на `rest`, а `cons` на `list*`.

Асимптотическая сложность работы в худшем случае – $O(n^2)$, где n – количество элементов списка. Она достигается за счет того, что на каждом шаге мы проверяем, не дублируется ли элемент.