Отчет по лабораторной работе № 3 по курсу «Функциональное программирование»

Студент группы М8О-306 МАИ *Дубинин Артем*, №5 по списку
Контакты: rusartdub@gmail.com
Работа выполнена: 08.04.2020

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

1. Тема работы

Последовательности, массивы и управляющие конструкции Коммон Лисп.

2. Цель работы

Научиться создавать векторы и массивы для представления матриц, освоить общие функции работы с последовательностями, инструкции цикла и нелокального выхода.

3. Задание (вариант №3.16)

Запрограммировать на языке Коммон Лисп функцию, принимающую два аргумента:

- A - двумерный массив, представляющий действительную матрицу,
- lis - список действительных чисел.

Функция должна возвращать новую матрицу, являющуюся копией A, но в которой заменены нулями элементы с чётной суммой индексов, встречающиеся в списке lis.
Исходный массив A должен оставаться неизменным.

4. Оборудование студента

Ноутбук Dell Vostro 5568, процессор Intel Core i5-7200U @ 4x 3.1GHz, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 18.04 bionic, компилятор sbcl, текстовый редактор Atom.

6. Идея, метод, алгоритм

Создаем новый массив ответов той же размерности, что и оригинальный массив. Проходим по оригинальному двумерному массиву рекурсивно, если сумма индексов элементов четна и элемент от этих индексов присутствует в списке, то записываем в новый массив под этими индексами ноль, иначе копируем элемент из изначального массива. Проходимся по списку рекурсивно, проверяя головной элемент списка и если он не подходит, то запускаем эту же функцию от хвоста списка.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun find-elem-in-list (l elem)
  (when l
    (if (= (car l) elem)
      t
      (find-elem-in-list (cdr l) elem))))

(defun traverse-array (ans arr lst n m i j)
  (if (< i n)
    (if (< j m)
      (progn
        (if (and (= (rem (+ i j) 2) 0) (find-elem-in-list lst
   (aref arr i j)))
          (setf (aref ans i j) 0)
          (setf (aref ans i j) (aref arr i j)))
        (traverse-array ans arr lst n m i (+ 1 j)))
      (traverse-array ans arr lst n m (+ 1 i) 0)
      )
    t))

(defun array-manip (A lis)
  (let* ((n (car (array-dimensions A)))
      (m (car (cdr (array-dimensions A))))
      (ans (make-array (list n m))))
    (traverse-array ans A lis n m 0 0)
    ans))
```

```
;(print (array-manip #2A((1 2 3) (4 5 6) (7 8 9)) '(1 2 3 4 5)))
```

8.2. Результаты работы

```
* (array-manip #2A((1 2 3) (4 5 6) (7 8 9)) '(1 2 3 4 5))
#2A((0 2 0) (4 0 6) (7 8 9))
* (array-manip #2A((1 2 3 4) (5 6 7 8)) '(6 8))
#2A((1 2 3 4) (5 0 7 0))
```

9. Замечания автора по существу работы

Чтобы выполнить работу, нужно было хорошо разобраться с массивами и ветлениями в common lisp.

10. Выводы

При выполнении работы старался избегать ошибок, которые были совершенны в прошлой лабораторной работе, в частности не использовать записывающие конструкции(хэш таблицы) и для поиска обходится просто обходом по списку.