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

Студент группы 8О-307Б-18 МАИ *Токарев Никита*, №21 по списку
Контакты: tokarevnikita08@mail.ru
Работа выполнена: 06.04.2021

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

Изучить Последовательности, массивы и управляющие конструкции Коммон Лисп.

3. Задание (вариант № 3.15)

Запрограммировать на языке Коммон Лисп функцию, принимающую в качестве аргументов следующие:
A - двумерный массив, представляющий действительную матрицу размера m*n,
u - вектор (строка) действительных чисел длины n,
v - вектор (столбец) действительных чисел длины m+1,
i - номер строки, 1<=i<=m,
j - номер столбца, 1<=j<=n.

Функция должна возвращать новую матрицу размера (m+1)*(n+1), полученную из A путём вставки элементов u в качестве новой строки после строки с номером i. последующей вставки элементов v в качестве нового столбца после столбца с номером j.

4. Оборудование студента

Процессор Intel® Core™ i3-5005U CPU @ 2.00GHz × 4, память: 3,8 Gb, разрядность системы: 64.

5. Программное обеспечение

UBUNTU 18.04.5 LTS, компилятор sbcl

6. Идея, метод, алгоритм

Метод состоит из двух итераций. Идея в том, чтобы для начала вставить u-вектор на место i+1 строки матрицы b. Затем же в полученную матрицу на место j+1 столбца вставить v-вектор. Мне предстоит разбить матрицу на составные части и по новому собрать ее. В программе две основные функции:

- (line-extend-matrix a k v) - В данной функции происходит создание матрицы b и копирование элементов матрицы до элемента a[k][...]. Затем же вставка вектора строки на позицию b[k][...] и копирование остальных элементов, если оно необходимо.
- (sum-product1 list1 list2) - рекурсивная функция: произведение n-х элементов списков list1 и list2 суммируется с результатом данной вызванной функции, где входными данными являются списки: list1 = list1(n + 1,...), list2 = list2(n + 1,...).

7. Сценарий выполнения работы

- Анализ возможных реализаций поставленной задачи на common Lisp
- Изучение синтаксиса и основных функций работы со списками common Lisp
- Реализация поставленной задачи на common Lisp

8. Распечатка программы и её результаты

8.1. Исходный код

Значения тестовых глобальных переменных представлено в исходном коде.

```
(defun column-extend-matrix (a k u)
  (let ((m (array-dimension a 0))
        (n (array-dimension a 1)))
    (let ((b (make-array (list m (+ 1 n))))) ; b = Matrix(* m (1 + n))
      (dotimes (i m)
        (dotimes (j k)
          (setf (aref b i j) (aref a i j)))) ; b[i][j] = a[i][j]
      (loop
        :for i :below m
        :do (setf (aref b i k) (aref u i))) ; b[i][k] = u[i]
      (dotimes (i m)
        (loop :for j :from k :to (- n 1) :do
```

```
          (setf (aref b i (+ 1 j)) (aref a i j)))) ; b[i][1 + j] = a[i][j]
      b)))

(defun line-extend-matrix (a k v)
  (let ((m (array-dimension a 0))
        (n (array-dimension a 1)))
    (let ((b (make-array (list (+ 1 m) n)))) ; b = Matrix(* (1 + m) n)
      (dotimes (i k)
        (dotimes (j n)
          (setf (aref b i j) (aref a i j)))) ; b[i][j] = a[i][j]
      (loop
        :for j :below n
        :do (setf (aref b k j) (aref v j))) ; b[k][j] = v[j], j : 0 to n
      (dotimes (j n)
        (loop :for i :from k :to (- m 1) :do
          (setf (aref b (+ 1 i) j) (aref a i j)))) ; b[i + 1][j] = a[i][j]
      b)))

(defun extend-matrix (a u v i j)
  (array-dimension a 0)
  (array-dimension a 1)
  (if (and (and (<= i (array-dimension a 0)) (<= j (array-dimension a 1))) (and (> i (
)
;test_variables
(defvar m1 (make-array '(3 3) :initial-contents '((0 0 0 ) (0 0 0) (0 0 0))))
(defvar u1 #(1 1 1))
(defvar v1 #(2 2 2 2))
(defvar m2 (make-array '(2 4) :initial-contents '((0 0 0 0) (0 0 0 0))))
(defvar u2 #(1 1 1 1))
(defvar v2 #(2 2 2))
(defvar m3 (make-array '(3 4) :initial-contents '((0 0 0 0) (0 0 0 0) (0 0 0 0))))
(defvar u3 #(1 1 1 1))
(defvar v3 #(2 2 2 2))
(defvar m4 (make-array '(1 1) :initial-contents '((0))))
(defvar u4 #(1))
(defvar v4 #(2 2))
;(extend-matrix m1 u1 v1 0 0)
;(extend-matrix m2 u2 v2 1 1)
;(extend-matrix m3 u3 v3 2 3)
;(extend-matrix m4 u4 v4 0 0)
;(extend-matrix m4 u4 v4 1 1)
;(extend-matrix m4 u4 v4 2 2)
```

8.2. Результаты работы

```
* (extend-matrix m1 u1 v1 0 0)

NIL
* (extend-matrix m1 u1 v1 1 1)

#2A((0 2 0 0) (1 2 1 1) (0 2 0 0) (0 2 0 0))
* (extend-matrix m3 u3 v3 2 3)

#2A((0 0 0 2 0) (0 0 0 2 0) (1 1 1 2 1) (0 0 0 2 0))
* (extend-matrix m4 u4 v4 2 2)

NIL
* (extend-matrix m4 u4 v4 1 1)

#2A((0 2) (1 2))
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание
04.04.2021	Вставка u и v на позицию i и j соответсвенно. По условию вставка происходит после i и j.	Изменил входные данный у функции: (column-extend-matrix (line-extend-matrix a (+ 1 i) u) (+ 1 j) v) .	
06.04.2021	i >=0 и j>=0	Добавил ограничение, такое что 0<i<=m,0<j<=n в (defun extend-matrix a u v i j)	

10. Замечания автора по существу работы

Замечаний нет.

11. Выводы

В ходе данной работы мне удалось познакомиться со встроенными функциями/инструментами, а также управляющими конструкциями коммон лисп с помощью которых мне удалось реализовать классический обход по матрице, используя циклы.