Отчет по лабораторной работе № 5 по курсу «Функциональное программирование»

Студентка группы М8О-307 МАИ *Довженко Анастасия*, №7 по списку
Контакты: tutkarma@gmail.com
Работа выполнена: 30.04.2019

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

1. Тема работы

Обобщённые функции, методы и классы объектов.

2. Цель работы

Научиться определять простейшие классы, порождать экземпляры классов, считывать и изменять значения слотов, научиться определять обобщённые функции и методы.

3. Задание (вариант №5.45)

Определите обычную функцию с двумя параметрами:
p - многочлен, т.е. экземпляр класса polynom,
a - список действительных чисел $(a_1 \dots a_n)$, где n - степень многочлена p.
Функция должна возвращать список действительных чисел $(d_0 \dots d_n)$, таких что:

$$P(x) = d_0 + d_1 \cdot (x - a_1) + d_2 \cdot (x - a_1) \cdot (x - a_2) + \dots + d_n \cdot (x - a_1) \cdot \dots \cdot (x - a_n)$$

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

Сначала я вывела формулу для получения i-ого d:

$$d_i = b_i + \sum_{j=i+1}^{n} (-1)^{j-i+1} \cdot d_j \cdot S(i, j-i), i = \overline{n, 0},$$

где $S(j, j-i)$ – операция суммирования всех сочетаний C_j^{j-i} элементов списка $(a_1 \dots a_j)$, b_i – i-ый коэффициент многочлена p.

Дальше запрограммировала эти вычисления. Кратко опишу работу программы: Основная функция get-d принимает на вход полином и список коэффициентов a. Коэффициенты многочлена получены с помощью функции list-coefs, которая рекурсивно обрабатывает термы. Т.к. список термов разрежен, необходимо отслеживать изменения степеней, что делается во вспомогательной функции cur-coef. В итоге имеем полный список всех коэффициентов.

В цикле получаем d_i. Чтобы получить этот элемент, находится сумма b_i и $\sum_{j=i+1}^{n}(-1)^{j-i+1} \cdot d_j \cdot S(i, j-i)$. Последнее слагаемое считается с помощью функции sum-mult-d-S, в которой считается сумма списка слагаемых. Этот список получен с помощью функции list-d-s, а сами слагаемые вычисляются в функции d-mult-S.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defclass polynom ()
    ((polunom-symbol :initarg :var1 :reader var1)
    ;; Разреженный список термов в порядке убывания степени
     (term-list :initarg :terms :reader terms)
    )
)


(defun make-term (&key order coeff)
    (list order coeff)
)


(defun order (term) (first term))
(defun coeff (term) (second term))
```

```
(defgeneric zerop1 (arg)
 (:method ((n number))    ; (= n 0)
  (zerop n)))

(defgeneric minusp1 (arg)
 (:method ((n number))    ; (< n 0)
  (minusp n)))


(defmethod print-object ((p polynom) stream)
  (format stream МЧ"[ (~s)
   ~:{~:[~:[+~;-~]~d~[~2*~;~s~*~:;~s^~d~]~;~]~}]"
          (var1 p)
          (mapcar (lambda (term)
                    (list (zerop1 (coeff term))
                          (minusp1 (coeff term))
                          (if (minusp1 (coeff term))
                              (abs (coeff term))
                              (coeff term))
                          (order term)
                          (var1 p)
                          (order term)))
                  (terms p))))


(defun list-coefs (p)
    (if p (cur-coef (first p) (second p) (list-coefs (rest p))))
)


(defun cur-coef (cur next tail)
    (cond ((null next) (if (= 0 (order cur))
                           (cons (coeff cur) tail)
                           (cons (coeff cur) (append (get-zeros
   (order cur)) tail)))
          )
          ((= (order cur) (1+ (order next))) (cons (coeff cur)
   tail))
          (t (cons (coeff cur) (append (get-zeros (1- (- (order
   cur) (order next)))) tail)))
    )
)
```

```
(defun get-zeros (n)
    (make-list n :initial-element '0)
)


(defun get-d (p a)
    (let  ((b (list-coefs (terms p)))
           (d (list (first (list-coefs (terms p)))))
           )
        (loop for i in (rest b)
            do (nconc d (list (+ i (sum-mult-d-S d
   (remove-last-el a)))))
        )
    (reverse d)
    )
)


; сумма всех слагаемых, кроме первого (a_i)
(defun sum-mult-d-S (d a)
    (sum-list (list-d-s 0 d a (list-length d)))
)

; получить все пары d_j и S, перемножить их, получить список конечных
   слагаемых
(defun list-d-s (j d a i)
    (if d (cons (d-mult-S (first d) (get-last-n-elems (-
   (list-length a) j) a) i) (list-d-s (1+ j) (rest d) a (1- i))))
)


; d_j * S -- одно слагаемое
(defun d-mult-S (d a j)
    (cond
        ((oddp j) (* d (sum-list (mult-list (combinations j a)))))
        (t (* -1 (* d (sum-list (mult-list (combinations j a))))))
    )
)

; последних, потому что список коэффициентов перевернут (an ... a1)
```

```
(defun get-last-n-elems (count list)
    (last list count)
)


(defun combinations (count list)
    (cond
        ((zerop count) '(()))
        ((endp list)   '())
        (t (nconc (mapcar (let ((item (first list)))
                            (lambda (comb) (cons item comb)))
                            (combinations (1- count) (rest
   list)))
                (combinations count (rest list))))
    )
)


(defun mult-list (list)
    (mapcar #'(lambda(x) (reduce '* x)) list)
)


(defun sum-list (list)
    (reduce '+ list)
)


(defun remove-last-el (list)
    (loop for i on list
        while (rest i)
            collect (first i)
    )
)


(defun main ()
    (let ((p1 (make-instance 'polynom
          :var1 'x
          :terms (list (make-term :order 2 :coeff 5)
                       (make-term :order 1 :coeff 3.3)
                       (make-term :order 0 :coeff -7))))
```

```
        (p2 (make-instance 'polynom
         :var1 'x
         :terms (list (make-term :coeff 1 :order 3)
                      (make-term :coeff 2 :order 1)
                      (make-term :coeff 1 :order 0))))
        (p3 (make-instance 'polynom
         :var1 'x
         :terms (list (make-term :order 5 :coeff -2)
                      (make-term :order 3 :coeff 4)
                      (make-term :order 1 :coeff -6))))
        (a1 (list 1 2 3))
        (a2 (list 2 2 2 2))
        (a3 (list 1 1 1 1 1 1)))

    (print "Polynom:")
    (print p1)
    (print "List:")
    (print a1)
    (print "Result:")
    (print (get-d p1 a1))

        (print "Polynom:")
    (print p2)
    (print "List:")
    (print a2)
    (print "Result:")
    (print (get-d p2 a2))

        (print "Polynom:")
    (print p3)
    (print "List:")
    (print a3)
    (print "Result:")
    (print (get-d p3 a3))

    (values))
)
```

8.2. Результаты работы

"Polynom:"
[МЧ $(X) + 5X^2 + 3.3X - 7$]
"List:"
(1 2 3)

"Result:"
(19.599998 18.3 5)
"Polynom:"
[МЧ $(X) + 1X^3 + 2X + 1$]
"List:"
(2 2 2 2)
"Result:"
(13 14 6 1)
"Polynom:"
[МЧ $(X) - 2X^5 + 4X^3 - 6X$]
"List:"
(1 1 1 1 1 1)
"Result:"
(-4 -4 -8 -16 -10 -2)

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание
30.04.2019	Задание было понятно неправильно	Изменена формула подсчета d так, как требовалось в задании	

10. Замечания автора по существу работы

Было бы очень хорошо, если в будущем в задание был бы добавлен пример работы требуемой функции, чтобы другие студенты не совершили ту же ошибку, что и я.

11. Выводы

Я научилась работать с простейшими классами, порождать экземпляры классов, производить различные действия над ними. Также мне пригодились навыки работы со списками, полученные в прошлых лабораторных.