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

Студент группы 8О-306 Гаврилов Максим, № по списку 7.

Контакты: cobraj@yandex.ru
Работа выполнена: 24.05.2022
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
Цель работы: научиться определять простейшие классы, порождать экземпляры классов, считывать и изменять значения слотов, научиться определять обобщённые функции и методы.

3. Задание (вариант № 5.43)

Определите обычную функцию der-polynom с одним параметром - многочленом, т.е. экземпляром класса polynom.

Функция должна вычислять производную P'(*x*), например:

```
;; P(x) = 5x^2 + 3.3x - 7
(setq p1 (make-instance 'polynom
        :var 'x
        :terms (list (make-term :order 2 :coeff 5)
                     (make-term :order 1 :coeff 3.3)
                     (make-term :order 0 :coeff -7))))
(der-polynom p1) ; => 10x + 3.3
```

4. Оборудование студента
процессор AMD FX(tm)-6300 Six-Core Processor 3.50 GHz, память 16ГБ, 64-разрядная система.

5. Программное обеспечение
ОС Windows 10, программа portacle, версия slime 2.24

6. Идея, метод, алгоритм
Обработка списков с использованием mapcar

7. Сценарий выполнения работы
1. Изучить функционал lisp в области ООП
2. Определить, какие функции необходимы для выполнения задания.
3. Если необходимо, написать вспомогательные функции.

8. Распечатка программы и её результаты

Программа

```
;;функции из лекций

(defclass polynom ()
 ((var-symbol :initarg :vari :reader vari) ; vari, а не var потому что моя программа не дает
                                           ; называть что-либо “var”
  ;; 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))   ; степень
(defun coeff (term) (second term))  ; коэффициент

(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~]~;~]~}]"
    (vari p)
    (mapcar (lambda (term)
      (list (zerop1 (coeff term))
        (minusp1 (coeff term))
        (if (minusp1 (coeff term))
          (abs (coeff term))
          (coeff term))
        (order term)
        (vari p)
        (order term)))
    (terms p))))

;;моя функция:

(defun der-polynom (pl)
  (make-instance 'polynom
    :vari
      (vari pl)
    :terms
      (mapcar (lambda (term)
        (list (- (order term) 1)
              (* (coeff term) (order term))))
        (terms pl))))


(defvar poly (make-instance 'polynom
  :vari 'x
  :terms (list (make-term :order 2 :coeff 5)
               (make-term :order 1 :coeff 3.3)
               (make-term :order 0 :coeff -7))))
```

Результаты

```
; SLIME 2.24
CL-USER> ;;функции из лекций

(defclass polynom ()
 ((var-symbol :initarg :vari :reader vari)
  ;; 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))   ; степень
(defun coeff (term) (second term))  ; коэффициент

(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~]~;~]~}]"
    (vari p)
    (mapcar (lambda (term)
      (list (zerop1 (coeff term))
        (minusp1 (coeff term))
        (if (minusp1 (coeff term))
          (abs (coeff term))
          (coeff term))
        (order term)
        (vari p)
        (order term)))
    (terms p))))

;;моя функция:

(defun der-polynom (pl)
  (make-instance 'polynom
    :vari
      (vari pl)
    :terms
      (mapcar (lambda (term)
        (list (- (order term) 1)
              (* (coeff term) (order term))))
        (terms pl))))



(defvar poly (make-instance 'polynom
  :vari 'x
  :terms (list (make-term :order 2 :coeff 5)
               (make-term :order 1 :coeff 3.3)
               (make-term :order 0 :coeff -7))))
POLY
```

```
CL-USER> poly
[МЧ (X) +5X^2+3.3X-7]
CL-USER> (der-polynom poly) ;полином из примера
[МЧ (X) +10X+3.3]
CL-USER> (der-polynom (der-polynom poly)) ;повторное дифференцирование
[МЧ (X) +10]
CL-USER>
(setf  poly (make-instance 'polynom ;для отрицательных степеней
  :vari 'x
  :terms (list (make-term :order 2 :coeff 5)
               (make-term :order -1 :coeff 3.3)
               (make-term :order 0 :coeff -7))))
[МЧ (X) +5X^2+3.3X^-1-7]
CL-USER> (der-polynom poly)
[МЧ (X) +10X-3.3X^-2]
CL-USER> (der-polynom (der-polynom poly))
[МЧ (X) +10+6.6X^-3]
CL-USER>
(setf  poly (make-instance 'polynom
  :vari 'x
  :terms (list (make-term :order 1 :coeff 1))))
[МЧ (X) +1X]
CL-USER> (der-polynom poly)
[МЧ (X) +1]
CL-USER> (der-polynom (der-polynom poly)) ;дифференцирование константы
[МЧ (X) ]
CL-USER>
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание
1				

10. Замечания автора по существу работы

REPL-среда, в которой я работаю, не позволяет называть пользовательские переменные и давать пользовательским слотам имя селектора «var». Так что я дал слоту var класса polynom селектор vari. Соответственно, при построении экземпляра класса plolynom, задавать значения в слоте надо, обращаясь по этому имени.

В работе я использовал функции и классы, определенные в курсе лекций по функциональному программированию, а именно:
polynom,
  print-object
make-term,
order,
coeff,
zerop1 (вспомогательная для polynom -> print-object),
minusp1 (вспомогательная для polynom -> print-object).

Функция дифференцирования с помощью mapcar формирует список измененных термов и затем на его основе строит новый экземпляр класса polynom, являющийся производной исходного.

Слагаемые многочлена, коэффициенты при которых обратились в ноль остаются в структуре и при дальнейшем дифференцировании арифметические действия над соответствующими термами продолжают выполняться. Это недостаток, но, на мой взгляд, не критический, так как не приводит к существенному росту вычислительной сложности, хоть в определенных специфических ситуациях может привести к более медленной работе функции, чем если бы неиспользуемые члены удалялись. Зато функция дифференцирования получается лаконичной и компактной, коей она бы не была, если бы в ней производилось удаление неиспользуемых членов.

11. Выводы

В ходе выполнения этой лабораторной работы я получил опыт работы на языке Коммон Лисп в стиле ООП. Также я получил опыт в программной реализации некоторых простых действий над многочленами и в работе с представлением многочленов в виде списка термов. Я, кстати, как-то раньше не задумывался о том, что обращение константы при дифференцировании в 0 без введения дополнительных правил согласуется с алгоритмом дифференцирования многочленов.