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

Студент группы М8О-307Б-18 МАИ *Скворцов Кирилл Алексеевич*, №20
Контакты: kilyla2@yandex.ru
Работа выполнена: 07.05.2021

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

Ноутбук Xiaomi mi Pro 15.6, процессор Intel Core i7-8550U CPU 1.80GHz, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Windows 10, онлайн компилятор для common-lisp, текстовый редактор VSCode (использовал т.к. там есть встроенный синтаксический валидатор).

6. Идея, метод, алгоритм

Идея заключается в сборе коэффициентов полинома в порядке возрастания от 0 до высшей степени. По правилу взятия производной новым коэффициентом становится произведение степени на текущий коэффициент. После получения новых коэффициентов просто осуществляется сдвиг данного списка, что сопоставить новые коэффициенты с новыми степенями многочлена. Далее по этому списку создается новый экземпляр класса.

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))


(defmethod print-object ((p polynom) stream)
  (format stream "(~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))))


(defgeneric zerop1 (arg)
 (:method ((n number))
  (zerop n)))


(defgeneric minusp1 (arg)
 (:method ((n number))
  (minusp n)))


(defun order (term)
    (first term))


(defun coeff (term)
    (second term))


(defun get-coef (order coef)
    (* (max order 0) coef))


(defun get-order (order)
    (max (- order 1) 0))


(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-coefs (p)
    (let  ((b (list-coefs (terms p)))
           (d (list (first (list-coefs (terms p)))))
           )(remove-nth 1 (reverse b))))


(defun remove-nth (n list)
  (remove-if (constantly t) list :start (1- n) :count 1))


(defun get-new-coefs (old-coefs)
    (loop for coef in old-coefs
          for index from 0 below (length old-coefs)
          collect (get-coef (+ 1 index) coef)))


(defun get-new-polynom (coefs)
    (let ((p1 (make-instance 'polynom
          :var1 'x
          :terms (loop for coef in (reverse coefs)
                       for index from 0 below (length coefs)
                             collect (make-term :order (- (length
   coefs) (+ 1 index)) :coeff coef)))))

    p1))


(defun der-polynom (p)
    (get-new-polynom (get-new-coefs (get-coefs p))))
```

```
(defun main ()
    (let ((p (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)))))
    (print (der-polynom p)))

    (let ((p (make-instance 'polynom
          :var1 'x
          :terms (list (make-term :order 5 :coeff 5)
                       (make-term :order 4 :coeff 3.3)
                       (make-term :order 3 :coeff -7)))))
    (print (der-polynom p)))

    (let ((p (make-instance 'polynom
          :var1 'x
          :terms (list (make-term :order 50 :coeff -7)))))
    (print (der-polynom p)))

    (let ((p (make-instance 'polynom
          :var1 'x
          :terms (list (make-term :order 6 :coeff 12)
                       (make-term :order 5 :coeff 3)
                       (make-term :order 4 :coeff 4)
                       (make-term :order 3 :coeff 5)
                       (make-term :order 2 :coeff 11.1)
                       (make-term :order 1 :coeff 19.9)
                       (make-term :order 0 :coeff 0)))))
    (print (der-polynom p)))

    (let ((p (make-instance 'polynom
          :var1 'x
          :terms (list (make-term :order 1 :coeff 5555.21)))))
    (print (der-polynom p))))

(main)
```

8.2. Результаты работы

(X) +10X+3.3
(X) +25X^4+13.2X^3−21X^2
(X) −350X^49
(X) +72X^5+15X^4+16X^3+15X^2+22.2X+19.9
(X) +5555.21

9. Дневник отладки

10. Замечания автора по существу работы

Замечаний нет.

11. Выводы

Благодаря данной лабораторной работе я научился работать с простейшими классами, порождать экземпляры классов, производить различные действия над ними. Также пригодились навыки работы со списками, полученные в прошлых лабораторных.