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

Студент группы 8О-308 Иванов Федор, № по списку 5.

Контакты: kenola82007@gmail.com
Работа выполнена: 03.06.2022

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

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

3. Задание (вариант № 5.31)

Определите обобщенную функцию MINUS1, позволяющую сменить знак числа либо многочлена на противоположный.

4. Оборудование студента

Ноутбук HP-Probook 440G5, Intel Core i5-8250U: 1,6 ГГц, до 3,4 ГГц при использовании технологии Intel Turbo Boost 2.0, 4 ядра, разрядность системы: 64

5. Программное обеспечение

ОС Linux Mint, программа LispWorks Personal Edition 6.1.1

6. Идея, метод, алгоритм

Для реализации данной задачи мною был заимствован код с сайта http://lisp.ystok.ru для реализации полиномов и некоторые методы. Был незначительно изменен метод mul2, тем что первый множитель свернул в терминальный объект. Были написаны 2 метода домножения на -1 для терма и полинома.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

Программа

```
(defclass polynom ()
 ((var-symbol :initarg :var :reader 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))
(defun coeff (term) (second term))


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

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

(defgeneric mul2 (arg1 arg2)
 (:method ((n1 number) (n2 number))
  (* n1 n2)))

(defgeneric minus1 (arg))

(defmethod add2 ((p1 polynom) (p2 polynom))
  (if (same-variable-p (var p1) (var p2))
     (make-instance 'polynom
                    :var (var p1)
                    :terms (add-terms (terms p1)
                                      (terms p2)))
     (error "Многочлены от разных переменных: ~s и ~s"
          p1 p2)))

(defmethod mul2 ((n number) (p2 polynom))
     (make-instance 'polynom
                    :var (var p2)
                    :terms (mul-terms (list (make-term :order 0
:coeff n)) (terms p2))))

(defmethod minus1((n number))
  (* -1 n))

(defmethod minus1((p2 polynom))
  (mul2 -1 p2))

(defun same-variable-p (v1 v2)
  ;; Переменные v1 и v2 - совпадающие символы
  (and (symbolp v1) (symbolp v2) (eq v1 v2)))
```

```
(defun add-terms (tl1 tl2)
  ;; Объединить списки термов tl1 и tl2,
  ;; упорядочивая по убыванию степеней
  (cond ((null tl1) tl2)
     ((null tl2) tl1)
     (t
          (let ((t1 (first tl1))
               (t2 (first tl2)))
          (cond ((> (order t1) (order t2))
               (adjoin-term t1
                              (add-terms (rest tl1) tl2)))
               ((< (order t1) (order t2))
               (adjoin-term t2
                              (add-terms tl1 (rest tl2))))
               (t
               ;; степени совпадают - суммируем коэффициенты
               (adjoin-term
                    (make-term :coeff (add2 (coeff t1) (coeff t2))
                               :order (order t1))
                    (add-terms (rest tl1)
                               (rest tl2)))))))))

(defun mul-terms (tl1 tl2)
  ;; Скрестить каждый терм из списка tl1 с каждым из списка tl2
  (if (null tl1)
     ()
     (add-terms (mul-term-by-all-terms (first tl1) tl2)
               (mul-terms (rest tl1) tl2))))

(defun mul-term-by-all-terms (t1 term-list)
  ;; Скрестить терм t1 с каждым из списка term-list
  (if (null term-list)
     ()
     (let ((t2 (first term-list)))
     ;; Коэффициенты перемножаем, а степени суммируем
     (adjoin-term (make-term :coeff (mul2 (coeff t1) (coeff t2))
                              :order (+ (order t1) (order t2)))
                    (mul-term-by-all-terms t1 (rest
term-list))))))


(defun adjoin-term (term term-list)
  ;; Добавить term к списку term-list
  (if (zerop1 (coeff term))   ; если коэффициент нулевой,
     term-list                ; то отбрасываем терм,
     (cons term term-list))) ; иначе накапливаем




(defmethod print-object ((p polynom) stream)
  (format stream "[Polynom(~s):
~:{~:[~:[+~;-~]~d~[~2*~;~s~*~:;~s^~d~]~;~]~}]"
```

```
          (var p)
          (mapcar (lambda (term)
                    (list (zerop1 (coeff term))
                          (minusp1 (coeff term))
                          (if (minusp1 (coeff term))
                              (abs (coeff term))
                              (coeff term))
                          (order term)
                          (var p)
                          (order term)))
                  (terms p))))


(setq p1 (make-instance 'polynom ; 5x^2 + 3.3x - 7
          :var 'x
          :terms (list (make-term :order 2 :coeff 5)
                       (make-term :order 1 :coeff 3.3)
                       (make-term :order 0 :coeff -7))))
(setq p2 (make-instance 'polynom ; x^10 - x^9 + 2x^8 - 3x^7 + 5x^6
- 8x^5 + 13x^4 - 21x^3 + 35x^2 - 56x + 91
          :var 'x
          :terms (list
                       (make-term :order 10 :coeff 1)
                       (make-term :order 9 :coeff -1)
                       (make-term :order 8 :coeff 2)
                       (make-term :order 7 :coeff -3)
                       (make-term :order 6 :coeff 5)
                       (make-term :order 5 :coeff -8)
                       (make-term :order 4 :coeff 13)
                       (make-term :order 3 :coeff -21)
                       (make-term :order 2 :coeff 35)
                       (make-term :order 1 :coeff -56)
                       (make-term :order 0 :coeff 91))))
```

Результаты

```
(print (minus1 3))
(print p1)
(print (minus1 p1))
(print p2)
(print (minus1 p2))

-3
[Polynom(X): +5X^2+3.3X-7]
[Polynom(X): -5X^2-3.3X+7]
[Polynom(X):
+1X^10-1X^9+2X^8-3X^7+5X^6-8X^5+13X^4-21X^3+35X^2-56X+91]
[Polynom(X):
-1X^10+1X^9-2X^8+3X^7-5X^6+8X^5-13X^4+21X^3-35X^2+56X-91]
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание
1	—	—	—	—

10. Замечания автора по существу работы

Мне было интересно изучить ООП в Lisp. В процессе выполнения работы я познакомился с классами и обобщенными методами в Common Lisp.

11. Выводы

ООП - крайне удобная парадигма в программировании. ООП используется повсеместно, с его помощью было написано крайне большое число рабочих проектов. Множество популярных языков программирования реализуют в себя ООП. И Lisp стал не исключением.

В ходе реализации этой лабораторной работы я познакомился с классами и обобщенными методами в Common Lisp, узнал как их объявлять и даже написал простейшую программу с использованием объектно ориентированного программирования.
Лабораторная работа оказалась крайне интересной.