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

Студент группы 8О-308 Коростелев Дмитрий, № по списку 11.

Контакты: dmitry.k48@yandex.ru
Работа выполнена: 06.05.2021
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

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

3. Задание (вариант № 5.38/3)

Определите обобщённую функцию SUB2, производящую вычитание двух чисел либо многочленов.

4. Оборудование студента

Ноутбук Asus VivoBook Pro 15, процессор Intel® Core™ i7-7700HQ CPU 2.80GHz 2.81GHz, память 8ГБ, 64-разрядная система.

5. Программное обеспечение

ОС Windows 10, программа LispWorks Personal Edition 7.1.2

6. Идея, метод, алгоритм

Определим generic метод и реализуем его для чисел и полиномов. Также нужно создать класс полиномов и функцию для вычитания полиномов. Вычитание будем выполнять по следующему алгоритму – заведем два указателя на термы полиномов, будем их двигать таким образом, что если степени у текущих термов совпадают, то вычитаем один терм из другого и если коэффициент не нулевой, то добавляем терм к новому полиному.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

Программа

```
; defenition of polyonom
(defclass polynom ()
     ((var-symbol :initarg :var :reader var)
      (term-list :initarg :terms :reader terms)))

(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 "[PL (~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))))

(defun same-variable-p (v1 v2)
     (and (symbolp v1) (symbolp v2) (eq v1 v2)))

(defun sub-terms-lists (tl1 tl2)
     (let
     ((i 0) (j 0 ) (ntl (list)))
          (do ()
               ((or (= i (length tl1) ) (= j (length tl2))))
               (let ((term1 (nth i tl1)) (term2 (nth j tl2)))
                    (if (= (order term1) (order term2))
                    (let ()
                    (setf ntl (join-term-to-list ntl (sub-terms term1
term2)))
                    (setf i (+ 1 i))
                    (setf j (+ 1 j))
                    )
                    (if (> (order term1) (order term2))
                    (let ()
                    (setf ntl (join-term-to-list ntl term1))
                    (setf i (+ 1 i))
                    )
                    (let ()
                    (setf ntl (join-term-to-list ntl (sub-terms (make-
term :order (order term2) :coeff 0) term2)))
                    (setf j (+ 1 j))
```

```
                    )))
               )
          )
          (do () ((= i (length tl1)))
               (let ()
                    (setf ntl (join-term-to-list ntl (nth i tl1)))
                    (setf i (+ 1 i))
               )
          )
          (do () ((= j (length tl2)))
               (let ((cur-term (nth j tl2)))
                    (setf ntl (join-term-to-list ntl (sub-terms (make-
term :order (order cur-term) :coeff 0) cur-term)))
                    (setf j (+ 1 j))
               )
          )
          (if (null ntl) (list (make-term :coeff 0 :order 0)) ntl)
     )
)

(defun join-term-to-list (ntl new-term)
     (append ntl (if (null new-term) NIL (list new-term)))
)

(defun sub-terms (t1 t2)
     (let ((c1 (coeff t1))
           (c2 (coeff t2))
           (o1 (order t1))
           (o2 (order t2)))
          (if (and (= o1 o2) (not (= (- c1 c2) 0)))
               (make-term :order o1 :coeff (- c1 c2))
               NIL
          )
     )
)

;-----------------------

;defenition of term
(defun make-term (&key order coeff)
     (list order coeff))

(defun order (term) (first term))
(defun coeff (term) (second term))
;------------------

;lab function
(defgeneric sub2 (arg1 arg2)
     (:documentation "Вычитает из arg1 arg2"))

(defmethod sub2 ((arg1 number) (arg2 number))
     (- arg1 arg2))
```

```
(defmethod sub2 ((p1 polynom) (p2 polynom))
     (if (same-variable-p (var p1) (var p2))
          (make-instance 'polynom
               :var (var p1)
               :terms (sub-terms-lists (terms p1) (terms p2))

          )
          (error "Polynoms have: different variables")
     )
)
```

Результаты

```
CL-USER 25 : 3 > p1
[PL (X) +1X]

CL-USER 26 : 3 > p2
[PL (X) +1X]

CL-USER 27 : 3 > p3
[PL (X) +1X^2]

CL-USER 28 : 3 > p4
[PL (Y) +2Y+3]

CL-USER 29 : 3 > p5
[PL (X) +1X^2+3X]

CL-USER 30 : 3 > (sub2 -1 2)
-3

CL-USER 31 : 3 > (sub2 p1 p2)
[PL (X) ]

CL-USER 32 : 3 > (sub2 p1 p3)
[PL (X) -1X^2+1X]
CL-USER 33 : 3 > (sub2 p2 p4)


Error: Polynoms have: different variables
  1 (abort) return to debug level 3.
  2 return to debug level 2.
  3 return to debug level 1.
  4 Return to top loop level 0.

Type :b for backtrace or :c <option number> to proceed.
Type :bug-form "<subject>" for a bug report template or :? for
other options.

CL-USER 34 : 4 > (sub2 p5 p1)
[PL (X) +1X^2+2X]
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы

Научился создавать и определять классы, их экземпляры, научился считывать и изменять значения слотов. Написал обобщенную функцию вычитания для чисел и полиномов. Протестировал данную функцию.