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

Студент группы 8О-307 МАИ *Бронников Максим*, №4 по списку
Контакты: max120199@gmail.com
Работа выполнена: 18.05.2020

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 №29)

Определите метод сложения числа и многочлена

(add2 4 p1)

(defmethod add2 ((n number) (p polynom))
  ...)

4. Оборудование ПЭВМ студента

Процессор Intel© Celeron© CPU @ 2.16GHz x 2, память: 4096Mb, разрядность системы: 64.

5. Программное обеспечение ЭВМ студента

OS Linux Mint 19.3 Cinnamon, среда SLIME 2.24 с реализацией языка SBCL 1.4.5.debian

6. Идея, метод, алгоритм

За основу взят класс многочлена `polynom` и определение обобщённой функции `add2` с портала *lisp.ystok.ru*. Класс многочлена ипортирован с методом его печати `print-object` и его вспомогательными функциями. Класс был незначительно изменён, поскольку запуск исходной версии выдавал ошибку на слове *:var*, а литеры *МЧ* некорректно отображаличь в отчёте. Так как в структуре класса подразумевается, что термы отсортированны, я добавил сортировку термов при инициализации объекта класса при помощи обобщённого метода `initialize-instance :after` и *быстрой сортировки Хоара* `quicksort`.

Для добавления числа написана вспомогательная функция `add-num`, которая обновляет переданный ей список термов полинома в соответствии с заданным ей числом. Этот метод последовательно проходит по отсортированному списку, пока встречаемые ею пордки степеней больше 0. Если в процессе поиска она не находит терм с порядком 0, она создает новый терм с этим порядком и вставляет его в соответствующее место списке. Если же терм был найден, она заменяет его на терм, с увеличнным на заданное число коэффицентом.

Искомый метод `add2` просто создает новый полином на основе полинома из аргумента, где список термов возвращаемого объекта - результат работы функции add-num с задаваемым числом и списком термов аргумента метода. Также добавлена версия `add2`, которая делает обобщённый метод сложения полинома с числом коммутативным.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
;;;; Функции и объявления классов, которые я взял с сайта ystok
;;;; слово var заменено на vars для предотвращения ошибки компиляции

(defun order (term) (first term))   ; степень
(defun coeff (term) (second term)) ; коэффицент

;; инкапсулированное создание терма
(defun make-term (&key order coeff)
  (list order coeff))

;;; полином - список термов, где терм - список из коэффицента и степени
(defclass polynom ()
 ((vars :initarg :vars :reader vars)
```

```
  ;; Разреженный список термов в порядке убывания степени
  (terms :initarg :terms :reader terms)))

;;; Обобщенная функция сложения
(defgeneric add2 (arg1 arg2)
 (:method ((n1 number) (n2 number))
  (+ n1 n2)))

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

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

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



;;;; остальные методы многочлена с сайта ystok я добавлять не стал
;;;; при этом тк термы должны быть отсортированны по порядку степеней,
;;;; добавим сортировку при инициализации объекта класса:

;;; функция сортировки Хоара:
(defun quicksort (terms) (
   if (null terms) nil
```

```
        (let*
            ((x (car terms))
                (r (cdr terms))
                ;; сортируем по убыванию
                (fn (lambda (a) (> (order a) (order x)))))
            (append
                (quicksort (remove-if-not fn r))
                (list x)
                (quicksort (remove-if fn r))))))

;;; сортируем переданне термы объекта при инициализации
(defmethod initialize-instance :after ((p polynom) &key)
    (setf (slot-value p 'terms) (quicksort (slot-value p 'terms))))



;;;; Функция по заданию:

;;; функция сложения числа к терму (0 smthng), если он существует
;;; или создания нового, если он не существуетсписок( термов отсорчен)
(defun add-num (terms num)
    (if (= num 0)
        ;; если число - 0, ничего делать не надо
        terms
        ;; иначе ищем:
        (if terms
            ;; выполняем поиск пока coeff > 0
            (if (> (order (first terms)) 0)
                (cons
                    (first terms)
                    (add-num (rest terms) num))
                ;; если мы дошли до значения >= 0, дальше поиск бессмысленен
                (if (= (order (first terms)) 0)
                    (cons
                        (make-term :order 0 :coeff (+ num (coeff (first terms))))
                        (rest terms))
                    (cons
                        (make-term :order 0 :coeff num)
                        terms)))
            ;; если значение не нашли, добавляем в конец новый терм
            (list (make-term :order 0 :coeff num)))))
```

```
;;; функция принимает полином и число, после чего создает полином
;;; в котором изменен список ермовn функцией add−num
(defmethod add2 ((p polynom) (n number))
   (make−instance 'polynom
      :vars (vars p)
      :terms (add−num (terms p) n)))

;;; сложение коммутативно
(defmethod add2 ((n number) (p polynom)) (add2 p n))
```

8.2. Результаты работы

```
(base) max@max−Lenovo−B50−30:~/FuncProg/lab5$ sbcl
This is SBCL 1.4.5.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD−style licenses.  See the CREDITS and COPYING files in the
distribution for more information.
* (load "main.lisp")

T
* (defvar p1 (make−instance 'polynom
  :vars 'x
  :terms (list (make−term :order 2 :coeff 5)
                    (make−term :order 1 :coeff 3.3)
      (make−term :order 3 :coeff 9))))

P1
* p1

[Pnm (X) +9X^3+5X^2+3.3X]
* (add2 p1 4)

[Pnm (X) +9X^3+5X^2+3.3X+4]
* p1

[Pnm (X) +9X^3+5X^2+3.3X]
* (add2 4 p1)

[Pnm (X) +9X^3+5X^2+3.3X+4]
* (defvar p0 (make−instance 'polynom
```

```
  :vars 'x
  :terms '()))

P0
* p0

[Pnm (X) ]
* (add2 1 p0)

[Pnm (X) +1]
* (setq p1 (add2 p1 -13))

[Pnm (X) +9X^3+5X^2+3.3X-13]
* (add2 14 p1)

[Pnm (X) +9X^3+5X^2+3.3X+1]
*
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы

Объектно-ориентированное программирование - крайне удобная парадигма в программировании, использование которой стало неотъемлемой частью подавляющего большинства серьёзных проектов. Практически все популярные языки программирования реализуют в себя ООП и *Lisp* не стал исключением. В процессе выполнения 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. Эта задача заставила меня поломать голову, однако я уверен, что полученные мной знания пригодятся мне в будущем.

Написанный метод работает за линейное время, поскольку в процессе выполнения происходит последовательный проход по списку с целью нахождения терма с нулевой степенью. Даже при использовании бинарного поиска асимптотика метода не изменилась бы, так как метод возвращает новый объект класса, с затратами по памяти $O(n)$, на создание которого необходимо затратить линейное время. Однако использование более быстрого поиска могло бы уменьшить реальное время исполнения метода.