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

Студент группы М8О-307-19 МАИ *Инютин Максим Андреевич*, №10 по списку
Контакты: mainyutin@gmail.com
Работа выполнена: 22.05.2022

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

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

3. Задание (вариант № 5.42)

Определите обычную функцию funcall-polynom с двумя параметрами:

- P — многочлен, т.е. экземпляр класса polynom,
- a — действительное число.

Функция должна вычислять значение многочлена P(x) при значении x := a.

4. Оборудование студента

Процессор Intel Core i7-9750H (12) @ 4.5GHz, память: 32 Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 20.04.4 LTS, комилятор GNU CLISP 2.49.92, текстовый редактор Atom 1.58.0

6. Идея, метод, алгоритм

Для представления многолочена я использую список пар из показателей степеней и коэффициентами. Для вычисления значения многочлена я прохожу по этому списку и вычисляю последовательно произведение коэффициента на значение в степени. Результат — сумма этих произведений.

Для возведения в степень я использую алгоритм бинарного возведений в степень, который имеет сложность $O(\log(y))$, где y — показатель степени. Я использовал наработки лаборатной работы № 1 для реализации возведения в степень.

Итоговая асимптотика алгоритма $O(\sum_{i=1}^{n} \log(y_i))$, где n — количество членов в многочлене, а y_i — показатель степени у i-го члена. Упрощёно можно записать $O(n \cdot \log(y))$.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
; 5.42

(defun make-term (&key order coeff)
        (list order coeff))

(defun order (term) (first term))

(defun coeff (term) (second term))

(defclass polynom ()
        ((var-symbol :initarg :var :reader var)
        (term-list :initarg :terms :reader terms)))

(defun odd (x) (logand x 1))

(defun div2 (x) (ash x -1))

(defun bpow (x y)
        (cond ((= y 0) 1.0)
        ((= (odd y) 1) (* x (bpow (* x x) (div2 y))))
        ((bpow (* x x) (div2 y)))))

(defun calc (order-coef-lst x0)
        (if (null order-coef-lst) 0.0
```

```
        (let* ((order-coef (first order-coef-lst))
        (order (first order-coef))
        (coef (second order-coef)))
        (+ (* coef (bpow x0 order)) (calc (rest order-coef-lst)
   x0)))))

(defmethod funcall-polynom ((p polynom) x0)
        (calc (terms p) x0))
```

8.2. Результаты работы

```
[1]> (load "lab5.lisp")
;; Loading file lab5.lisp ...
;; Loaded file lab5.lisp
#P"/home/engineerxl/Study/6
   term/functional-programming/lab5/lab5.lisp"
[2]> (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))))
#<POLYNOM #x000000800032DA99>
[3]> (funcall-polynom p1 10)
526.0
[4]> (funcall-polynom p1 -10)
460.0
[5]> (funcall-polynom p1 0)
-7.0
[6]> (setq p2 (make-instance 'polynom
          :var 'x
          :terms (list (make-term :order 3 :coeff 1)
                       (make-term :order 2 :coeff 2)
                       (make-term :order 1 :coeff 3)
                       (make-term :order 0 :coeff 4))))
#<POLYNOM #x000000800033EAB9>
[7]> (funcall-polynom p2 1)
10.0
[8]> (funcall-polynom p2 -1)
2.0
[9]> (funcall-polynom p2 0)
4.0
[10]> (setq p3 (make-instance 'polynom
          :var 'x
          :terms (list (make-term :order 16 :coeff 4)
                       (make-term :order 8 :coeff 3)
                       (make-term :order 4 :coeff 2)
                       (make-term :order 2 :coeff 1))))
#<POLYNOM #x0000008000329809>
[11]> (funcall-polynom p3 0)
0.0
[12]> (funcall-polynom p3 2)
262948.0
```

```
[13]> (funcall-polynom p3 -2)
262948.0
[14]> (funcall-polynom p3 3)
1.7220674E8
[15]> (funcall-polynom p3 -3)
1.7220674E8
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Сложность полученного решения $O(n \cdot \log(y))$. Если хранить все члены многочлена, а именно хранить все коэффициенты и степени, не превыщающие максимальную, то можно добиться линейной сложности $O(\log(y))$.

Однако в случае с многочленом $P(x) = x^{10000}$ этот подход будет плохо себя показывать, а мой алгоритм будет работать быстро.

11. Выводы

Я познакомился с обобщёнными функциями, методами и классами объектов в Common Lisp. Описание классов напомнило язык C#, где нужно описывать getter и setter для полей. До этого я не использовал ООП вместе с функциональной парадигмой, в целом, я не заметил ощутимой разницу между ООП на Common Lisp и другими языками (не функциональными).