Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет: “Информационные технологии и прикладная математика”
Кафедра: 806 “Вычислительная математика и программирование”

Отчет по лабораторной работе №5

по курсу «Функциональное программирование»

Студент: Полей-Добронравова Амелия Вадимовна

Группа: М8О-307Б, № по списку 16.

Преподаватель: Иванов Д. А., доц. каф. 806

Дата: 18.05.2021

Итоговая оценка:

Подпись преподавателя:

Москва, 2021

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 № 5.44)

Определите обычную функцию с двумя параметрами:

a - список действительных чисел (a_1 ... a_k),

в - список действительных чисел (d_0 ... d_m).

Функция должна конструировать многочлен, т.е. экземпляр класса polynom, степени n, n = min(k,m), следующего вида:

$$d_0 + d_1*(x-a_1) + d_2*(x-a_1)*(x-a_2) + ... + d_n*(x-a_1)*...*(x-a_n)$$

4. Оборудование студента

MacBook Pro (15-inch, Mid 2012), процессор 2,3 GHz Intel Core i7, память 8 ГБ.

5. Программное обеспечение

macOS Mojave 10.14.6

6. Идея, метод, алгоритм

Основная функция, выполняющая задачу my_polynom.
Она проверяет на пустоту два входных списка. Если пуст второй список (d_0 ... d_m), то возвращает NIL. Если пуст первый с коэффициентами a=(a_1 ... a_k) и не пуст второй, то возвращает полином из одного элемента d_0.
Иначе выполняется сама задача. Формируется полином от переменной z, и её термы имеют order:0, чтобы сама буква z не фигурировала. Далее создается список терм: отдельно полином из элемента d_0, отдельно всё остальное. Циклом от 1 до посчитанного по условию задачи n посчитаются недостоющие слагаемые с помощью функции glitch, которая соберет $d_i*(x-a_1)*...*(x-a_i)$ для каждого i в цикле.

Функция glitch работает следующим образом: формирует полином с аргументом d_i, термы которого образуются в функции braces.

Функция braces циклом заменяет полином br, состоящий из первой скобки $(x-a_1)$ на полином от аргумента br, у которого коэффициентом будет следующая по счету скобка. Таким образом накапливается произведение скобок, благодаря тому, что каждая скобка становится коэффициентом предшествующего ей многочлена.

Также используется функция minus, которая образует многочлен вида $(x-a_i)$, и на вход ей подается $-a_i$.

7. Сценарий выполнения работы

Продумать рекурсивную логику накопления многочленов и написать программу, отладить на тестах.

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))


(defmethod zerop1 ((p polynom))

  (null (terms p)))


(defmethod minusp1 ((p polynom))

   nil)


(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 "[МЧ (~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 glitch (list1 list2 i)
      (cond ((null list1) null)
        ((null list2) null)
        (t
          (make-instance 'polynom
                         :var (nth i list2)
                         :terms (braces list1 i)))))

(defun braces(list1 i)
      (let ((br (minus (* -1 (nth 0 list1)))))
```

```
      (loop for j upfrom 1 below i do
            (setq br (make-instance 'polynom
                    :var br
                    :terms (list (make-term
                                          :order 1
                                  :coeff (minus (* -1 (nth j list1)))))))))
    (list (make-term
            :order 1
            :coeff br))))


(defun minus(cof) ;;works
      (make-instance 'polynom
               :var 'x
               :terms (list (make-term :order 1
                                       :coeff 1)
                            (make-term :order 0
                                       :coeff cof))))


(defun pair(cof1 cof2) ;;works
      (make-instance 'polynom
                  :var (minus cof2)
                  :terms (list (make-term
                                          :order 1
                                          :coeff (minus cof1)))))


(defun my_polynom(list1 list2)
      (cond ((null list2) nil)
            ((null list1)
                  (make-instance 'polynom
```

```
                                        :var 'z
                                        :terms (list (make-term
                                                      :order 0
                                                      :coeff (nth 0
list2)))))
        (t
      (let ((n (min (length list1) (1- (length list2)))))
            (make-instance 'polynom
               :var 'z
               :terms (nconc (list
                                    (make-term
                                                :order 0
                                   :coeff(nth 0 list2)))
                                (loop for j upfrom 1 below (1+ n)
                                      collect (make-term :order 0
                                                         :coeff (glitch list1
list2 j)))))))))
```

```
(my_polynom '(1 2 3 4) '(6 7 8 9 10 11))
(my_polynom '(1 2 3 4) '())
(my_polynom '() '(1 2 3 4))
(my_polynom '(6 7 8 9 10 11) '(1 2 3 4))
```

Результаты

```
CL-USER 15 >
(my_polynom '(1 2 3 4) '(6 7 8 9 10 11))
[M4 (Z) +6+[M4 (7) +[M4 (X) +1X-1]7]+[M4 (8) +[M4 ([M4 (X) +1X-1]) +[M4 (X) +1X-2][M4 (X) +1X-1]]8]+[M4 (9) +[M4 ([M4 ([M4 (X) +1X-1]) +[M4 (X)
 +1X-2][M4 (X) +1X-1]]) +[M4 (X) +1X-3][M4 ([M4 (X) +1X-1]) +[M4 (X) +1X-2][M4 (X) +1X-1]]]9]+[M4 (10) +[M4 ([M4 ([M4 ([M4 (X) +1X-1]) +[M4 (X)
 +1X-2][M4 (X) +1X-1]]) +[M4 (X) +1X-3][M4 ([M4 (X) +1X-1]) +[M4 (X) +1X-2][M4 (X) +1X-1]]]) +[M4 (X) +1X-4][M4 ([M4 ([M4 (X) +1X-1]) +[M4 (X)
+1X-2][M4 (X) +1X-1]]) +[M4 (X) +1X-3][M4 ([M4 (X) +1X-1]) +[M4 (X) +1X-2][M4 (X) +1X-1]]]]10]]

CL-USER 16 > (my_polynom '(1 2 3 4) '())
NIL

CL-USER 17 > (my_polynom '() '(1 2 3 4))
[M4 (Z) +1]

CL-USER 18 >
```

Рассмотрим первый приведенный мной тест.

Ответ должен иметь вид:
6+7(x-1)+8(x-1)(x-2)+9(x-1)(x-2)(x-3)+10(x-1)(x-2)(x-3)(x-4)

Мой вывод:

[МЧ (Z) +6+[МЧ (7) +[МЧ (X) +1X-1]7]+[МЧ (8) +[МЧ ([МЧ (X) +1X-1]) +
[МЧ (X) +1X-2][МЧ (X) +1X-1]]8]+[МЧ (9) +[МЧ ([МЧ ([МЧ (X) +1X-1]) +
[МЧ (X) +1X-2][МЧ (X) +1X-1]]) +[МЧ (X) +1X-3][МЧ ([МЧ (X) +1X-1]) +
[МЧ (X) +1X-2][МЧ (X) +1X-1]]]9]+[МЧ (10) +[МЧ ([МЧ ([МЧ ([МЧ (X) +1X-1]) +
[МЧ (X) +1X-2][МЧ (X) +1X-1]]) +[МЧ (X) +1X-3][МЧ ([МЧ (X) +1X-1]) +
[МЧ (X) +1X-2][МЧ (X) +1X-1]]]) +[МЧ (X) +1X-4][МЧ ([МЧ ([МЧ (X) +1X-1]) +
[МЧ (X) +1X-2][МЧ (X) +1X-1]]) +[МЧ (X) +1X-3][МЧ ([МЧ (X) +1X-1]) +
[МЧ (X) +1X-2][МЧ (X) +1X-1]]]]10]]

Сделаем наоборот, чтобы второй список был короче:

```
CL-USER 18 > (my_polynom '(6 7 8 9 10 11) '(1 2 3 4))
[МЧ (Z) +1+[МЧ (2) +[МЧ (X) +1X-6]2]+[МЧ (3) +[МЧ ([МЧ (X) +1X-6]) +[МЧ (X »
) +1X-7][МЧ (X) +1X-6]]3]+[МЧ (4) +[МЧ ([МЧ ([МЧ (X) +1X-6]) +[МЧ (X) +1X- »
7][МЧ (X) +1X-6]]) +[МЧ (X) +1X-8][МЧ ([МЧ (X) +1X-6]) +[МЧ (X) +1X-7][МЧ »
(X) +1X-6]]]4]]
```

Как мы видим, последний многочлен от 4, значит все коэффициенты задействованы

9. Дневник отладки

№ Дата, время Событие Действие по исправлению Примечание

10. Замечания автора по существу работы

Данная лабораторная работа оказалась для меня самой сложной из всех по данному предмету.

11. Выводы

Как я раньше писала в отчетах, мне лично совершенно не понравилось формирование объектов в Common Lisp, и основная проблема, которая передо мной возникла, был сам процесс образования. Также не сразу пришло понимание, что полиномы могут иметь в аргументах не только буквы, но и целые выражения, что и привело к данному решению.