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

Студент группы 8О-308 МАИ *Марков Александр*, №15 по списку
Контакты: markov.lifeacc@gmail.com
Работа выполнена: 15.05.2021

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 №5.23)

Определить обобщённую функцию и методы on-single-line3-p - предикат,

- принимающий в качестве аргументов три точки (радиус-вектора) и необязательный параметр tolerance (допуск),
- возвращающий T, если три указанные точки лежат на одной прямой (вычислять с допустимым отклонением tolerance).

Точки могут быть заданы как декартовыми координатами (экземплярами cart), так и полярными (экземплярами polar).

```
(defgeneric on-single-line3-p (v1 v2 v2 &optional tolerance))
(defmethod on-single-line3-p ((v1 cart) (v2 cart) (v3 cart) &optional (tolerance 0.001))
  ...)
```

4. Оборудование студента

Процессор AMD Ryzen 5 4600H 3.00 GHz, память: 16Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Windows 10, среда LispWorks Personal Edition 7.1.2

6. Идея, метод, алгоритм

Пусть заданы три точки p_1, p_2, p_3. Рассмотрим два вектора $\overrightarrow{p_1p_2}$, $\overrightarrow{p_1p_3}$. Точки p_1, p_2, p_3 лежат на одной прямой, если косинус угла между векторами $\overrightarrow{p_1p_2}$, $\overrightarrow{p_1p_3}$ равен по модулю 1. Если задан необязательный параметр tolerance, то значение модуля косинуса должно находиться в отрезке значений [1 - tolerance; tolerance]. Чтобы найти координаты векторов $\overrightarrow{p_1p_2}$ и $\overrightarrow{p_1p_3}$, используются функции из лекций add2 и mul2.

- Полярные координаты. Находим косинус разности углов векторов $\overrightarrow{p_1p_2}$, $\overrightarrow{p_1p_3}$. Сравниваем разность 1 и модуля полученного косинуса с tolerance.
- Декартовы координаты. Находим косинус угла между $\overrightarrow{p_1p_2}$, $\overrightarrow{p_1p_3}$ по следующей формуле: $(\overrightarrow{p_1p_2}, \overrightarrow{p_1p_3})/(|\overrightarrow{p_1p_2}| \cdot |\overrightarrow{p_1p_3}|)$. Для нахождения скалярного произведения используется обобщённая функция scalar-product, а именно метод, уточнённый для векторов, заданных в декартовых координатах. Затем разность 1 и модуля полученного косинуса сравнивается с tolerance.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defclass cart ()
 ((x :initarg :x :reader cart-x)
  (y :initarg :y :reader cart-y)))

(defmethod print-object ((c cart) stream)
  (format stream "[CART x ~d y ~d]"
          (cart-x c) (cart-y c)))

(defclass polar ()
 ((radius :initarg :radius :accessor radius)
  (angle  :initarg :angle  :accessor angle)))

(defmethod print-object ((p polar) stream)
  (format stream "[POLAR radius ~d angle ~d]"
          (radius p) (angle p)))

```

```
(defun square (x) (* x x))

(defmethod radius ((c cart))
  (sqrt (+ (square (cart-x c))
           (square (cart-y c)))))

(defmethod angle ((c cart))
  (atan (cart-y c) (cart-x c)))

(defgeneric to-polar (arg)
 (:documentation "Преобразование аргумента в полярную систему.")
 (:method ((p polar))
  p)
 (:method ((c cart))
  (make-instance 'polar
                 :radius (radius c)
                 :angle (angle c))))

(defmethod cart-x ((p polar))
  (* (radius p) (cos (angle p))))

(defmethod cart-y ((p polar))
  (* (radius p) (sin (angle p))))

(defgeneric to-cart (arg)
 (:documentation "Преобразование аргумента в декартову систему.")
 (:method ((c cart))
  c)
 (:method ((p polar))
  (make-instance 'cart
                 :x (cart-x p)
                 :y (cart-y p))))

(defgeneric add2 (arg1 arg2)
 (:method ((n1 number) (n2 number))
  (+ n1 n2)))

(defmethod add2 ((c1 cart) (c2 cart))
  (make-instance 'cart
                 :x (+ (cart-x c1) (cart-x c2))
                 :y (+ (cart-y c1) (cart-y c2))))

```

```
(defmethod add2 ((p1 polar) (p2 polar))
  (to-polar (add2 (to-cart p1)
                  (to-cart p2))))

(defmethod add2 ((c cart) (p polar))
  (add2 c (to-cart p)))

(defmethod mul2 ((n number) (c cart))
  (make-instance 'cart
                 :x (* n (cart-x c))
                 :y (* n (cart-y c))))

(defun normalize-angle (a)
  ;; Привести полярный угол a в диапазон [pi, -pi)
  (let* ((2pi (* 2 pi))
         (rem (rem a 2pi)))
    (cond ((< pi rem)
           (- rem 2pi))
          ((<= rem (- pi))
           (+ 2pi rem))
          (t rem))))

(defmethod mul2 ((n number) (p polar))
  (let ((a (angle p)))
    (make-instance 'polar
                   :radius (abs (* n (radius p)))
                   :angle (if (< n 0)
                              (normalize-angle (+ a pi))
                              a))))

(defgeneric scalar-product (arg1 arg2)
  (:documentation "Скалярное произведение")
  (:method ((c1 cart) (c2 cart))
    (+ (* (cart-x c1) (cart-x c2)) (* (cart-y c1) (cart-y c2)))))

(defgeneric on-single-line3-p (v1 v2 v3 &optional tolerance))

(defmethod on-single-line3-p ((v1 cart) (v2 cart) (v3 cart)
  &optional (tolerance 0.001))
  (let ((v12 (add2 v2 (mul2 -1 v1)))
        (v13 (add2 v3 (mul2 -1 v1)))
       )
```

```
    (< (- 1 (abs (/ (scalar-product v12 v13) (radius v12) (radius v13))))
      tolerance
    )))

(defmethod on-single-line3-p ((v1 polar) (v2 polar) (v3 polar)
  &optional (tolerance 0.001))
  (let ((v12 (add2 v2 (mul2 -1 v1)))
        (v13 (add2 v3 (mul2 -1 v1)))
       )
    (< (- 1 (abs (cos (- (angle v12) (angle v13)))))
      tolerance
    )))
```

8.2. Результаты работы

```
CL-USER 5 > first
[CART x 0 y 0]

CL-USER 6 > second
[CART x -1 y -1]

CL-USER 7 > third
[CART x 1 y 1]

CL-USER 8 > (on-single-line3-p first second third)
T

CL-USER 9 > (on-single-line3-p (to-polar first) (to-polar second) (to-polar third))
T

CL-USER 21 > first
[POLAR radius 0 angle 0]

CL-USER 22 > second
[POLAR radius 5 angle 2.356194490192345D0]

CL-USER 23 > third
[POLAR radius 3 angle 0]

CL-USER 24 > (on-single-line3-p first second third)
NIL

CL-USER 25 > (on-single-line3-p (to-cart first) (to-cart second) (to-cart third))
```

```
NIL

CL-USER 47 > first
[POLAR radius 0 angle 0]

CL-USER 48 > second
[POLAR radius 1 angle 0]

CL-USER 49 > third
[POLAR radius 1 angle 0.2617993877991494D0]

CL-USER 50 > (on-single-line3-p first second third)
NIL

CL-USER 51 > (on-single-line3-p first second third 0.04)
T
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы

Я научился работать с обобщёнными функциями и методами, определять простейшие классы, порождать экземпляры классов, считывать и изменять значения слотов.