

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

Студент группы 8О-308 МАИ *Балес Александр*, №3 по списку
Контакты: `aleks_bales@mail.ru`
Работа выполнена: 22.05.2016

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 5.25)

Дан экземпляр класса `triangle`, причем все вершины треугольника могут быть заданы как декартовыми координатами (экземплярами класса `cart`), так и полярными (экземплярами класса `polar`).

Задание: Определить обычную функцию медиана, возвращающую объект-отрезок (экземпляр класса `line`), являющийся медианой первого угла `vertex1`. Концы результирующего отрезка могут быть получены либо в декартовых, либо в полярных координатах.

```
(setq tri (make-instance 'triangle
    :1 (make-instance 'cart-или-polar ...)
    :2 (make-instance 'cart-или-polar ...)
    :3 (make-instance 'cart-или-polar ...)))
(медиана tri) => [ОТРЕЗОК ...]
```

4. Оборудование студента

Процессор Intel Core i5-3210 4@2.5GHz, память: 8192Mb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 14.04, среда GNU Common Lisp 2.6.10

6. Идея, метод, алгоритм

Все классы и обобщенные ф-ии были взяты и модернизированы с лекций, а ф-ия **median** выполняет примерно следующие действия: нам необходимо получить отрезок, соединяющий точку vertex1 треугольника с точкой лежащей на середине стороны, противоположной vertex1. Иными словами, медиана опущенная из вершины vertex1. Чтобы найти середины отрезка противоположной стороны, необходимо сложить координаты вершин vertex2 и vertex3, и поделить координаты на два. В случае же полярной системы координат, нам достаточно уметь приводить полярные координаты к декартовым.

7. Распечатка программы и её результаты

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

(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)))

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

(defmethod angle ((c cart))
  (atan (cart-y c) (cart-x 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)
  (:method ((c cart))
    c)
  (:method ((p polar))
    (make-instance 'cart
      :x (cart-x p)
      :y (cart-y p)))) )

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

(defmethod add ((p1 polar) (p2 polar))
  (make-instance 'cart
    :x (+ (cart-x p1) (cart-x p2))
    :y (+ (cart-y p1) (cart-y p2))))

(defmethod del ((c1 cart) (n number))
  (make-instance 'cart
    :x (/ (cart-x c1) n)
    :y (/ (cart-y c1) n)))

(defclass line ()
  ((start :initarg :start :accessor line-start)
   (end   :initarg :end   :accessor line-end)))

(defmethod print-object ((lin line) stream)
  (format stream OTPE3OK"[ ~s ~s]"
    (line-start lin) (line-end lin)))

(defclass triangle ()
  ((vertex1 :initarg :1 :reader vertex1)
   (vertex2 :initarg :2 :reader vertex2))

```

```

(vertex3 :initarg :3 :reader vertex3)))

(defmethod print-object ((tri triangle) stream)
  (format stream TPEYI"[ ~s ~s ~s]"
    (vertex1 tri) (vertex2 tri) (vertex3 tri)))

(defun median (tri)
  (make-instance 'line
    :start (to-cart (vertex1 tri))
    :end (del (add (vertex2 tri) (vertex3 tri)) 2)))


(setq triCart (make-instance 'triangle
  :1 (make-instance 'cart :x 4 :y 3)
  :2 (make-instance 'cart :x 7 :y 5)
  :3 (make-instance 'cart :x 5 :y -1)))

(setq triPolar (make-instance 'triangle
  :1 (make-instance 'polar :radius (radius (vertex1
triCart)) :angle (angle (vertex1 triCart)))
  :2 (make-instance 'polar :radius (radius (vertex2
triCart)) :angle (angle (vertex2 triCart)))
  :3 (make-instance 'polar :radius (radius (vertex3
triCart)) :angle (angle (vertex3 triCart)))))

(print (median triCart))
(print (median triPolar))

```

7.1. Результаты

 stdin

Standard input is empty

 stdout

[ОТРЕЗОК [CART x 4 y 3] [CART x 6 y 2]]

[ОТРЕЗОК [CART x 3.9999998 y 3.0] [CART x 6.0 y 2.0000002]]

8. Замечания, выводы

Я приобрел навыки работы с простейшими классами, научился порождать экземпляры классов, считывать и изменять значения слотов, а также определять обобщённые функции и методы.