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

Студент группы 8О-307 МАИ *Ефимов Александр*, №7 по списку
Контакты: aleks.efimov2011@yandex.ru
Работа выполнена: 09.04.2021

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: №5.37

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 высота, возвращающую объект-отрезок (экземпляр класса *line*), являющийся высотой 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 ...)))

(высота tri) =>  [ОТРЕЗОК ...]
```

4. Оборудование студента

Процессор Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, память: 7.6Gi, разрядность системы: 64.

5. Программное обеспечение

ОС Arch Linux, система CLisp.

6. Идея, метод, алгоритм

При вызове функции все точки треугольника считаются однотипными. Если точки полярные, то перевести их в декартовы координаты.

Пусть нам даны точки треугольника A, B, C, причем необходимо найти конец вектора высоты из точки A. Для этого проектируем вектор $\vec{BA}$ на вектор $\vec{BC}$ с помощью скалярного произведения. Полученное произведение даст длину проектированного сторону $\vec{BC}$. Поделив её на длину самой стороны, можно получить коэффициент, который после умножения на сторону дает смещение точки B для достижения конца вектора отрезка высоты P. Формульно это будет иметь вид:

$$\vec{BA} = A - B$$
$$\vec{BC} = C - B$$
$$coef = (BA \cdot BC)/(BC \cdot BC)$$
$$P = B + coef \cdot BA$$

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
;; Helper

(defun square (value) (* value value))

;; Cartesian point definition

(defclass cartesian ()
  ((x
     :initarg :x
     :initform 0
     :reader x-val)
   (y
     :initarg :y
     :initform 0
     :reader y-val)))
```

```
16
17 (defmethod radius ((c cartesian))
18   (sqrt (+ (square (x-val c))
19            (square (y-val c)))))
20
21 (defmethod angle ((c cartesian))
22   (if (= (x-val c) 0)
23       (/ pi 2)
24       (atan (y-val c) (x-val c))))
25
26 ;; Polar point definition
27
28 (defclass polar ()
29   ((r
30      :initarg :r
31      :initform 0
32      :reader radius)
33    (phi
34      :initarg :phi
35      :initform 0
36      :reader angle)))
37
38 (defmethod x-val ((p polar))
39   (* (radius p) (cos (angle p))))
40
41 (defmethod y-val ((p polar))
42   (* (radius p) (sin (angle p))))
43
44 ;; Converters between point types
45
46 (defgeneric to-polar (arg)
47  (:documentation "Convert point to polar")
48  (:method ((p polar))
49   p)
50  (:method ((c cartesian))
51   (make-instance 'polar
52                  :r   (radius c)
53                  :phi (angle c))))
54
55 (defgeneric to-cartesian (arg)
56  (:documentation "Convert point to cartesian")
57  (:method ((c cartesian))
```

```
58    c)
59   (:method ((p polar))
60    (make-instance 'cartesian
61                   :x (x-val p)
62                   :y (y-val p))))
63
64  ;; Functions, assisting in calculation of height
65
66  (defun apply-cartesians (func a b)
67    (make-instance 'cartesian
68                   :x (funcall func (x-val a) (x-val b))
69                   :y (funcall func (y-val a) (y-val b))))
70
71  (defun apply-scalar (func a coef)
72    (make-instance 'cartesian
73                   :x (funcall func (x-val a) coef)
74                   :y (funcall func (y-val a) coef)))
75
76  (defun dot-product (a b)
77    (+ (* (x-val a) (x-val b)) (* (y-val a) (y-val b))))
78
79  ;; Shapes
80
81  (defclass line ()
82   ((start
83      :initarg :start
84      :reader line-start)
85    (end
86      :initarg :end
87      :accessor line-end)))
88
89  (defclass triangle ()
90   ((vertex1 :initarg :1 :reader vertex1)
91    (vertex2 :initarg :2 :reader vertex2)
92    (vertex3 :initarg :3 :reader vertex3)))
93
94  ;; Function, calculating the height line
95
96  (defmethod height ((tri triangle))
97    (let* ((cart1     (to-cartesian (vertex1 tri)))
98           (cart2     (to-cartesian (vertex2 tri)))
99           (cart3     (to-cartesian (vertex3 tri)))
```

```
          (BC        (apply-cartesians #'- cart3 cart2))
          (BA        (apply-cartesians #'- cart1 cart2))
          (coef      (/ (dot-product BA BC) (dot-product BC BC)))
          (end-point (apply-cartesians #'+ cart2 (apply-scalar #'* BC coef))))
    (if (typep (vertex1 tri) 'cartesian)
        (make-instance 'line :start cart1 :end end-point)
        (make-instance 'line :start (vertex1 tri) :end (to-polar end-point)))))

;; Class printers

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

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

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

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

8.2. Результаты работы

```
[1]> (load "classes.lisp")
;; Loading file classes.lisp ...
;; Loaded file classes.lisp
[2]> (setq c1 (make-instance 'cartesian :x 0 :y 0))
[CART x 0 y 0]
[3]> (setq c2 (make-instance 'cartesian :x 4 :y 4))
[CART x 4 y 4]
[4]> (setq c3 (make-instance 'cartesian :x -1 :y 5))
[CART x -1 y 5]
[5]> (height (make-instance 'triangle :1 c3 :2 c1 :3 c2))
[LINE [CART x -1 y 5] [CART x 2 y 2]]
[6]> (setq p1 (to-polar c1))
[POLAR radius 0 angle 1.570796326794896619L0]
[7]> (setq p2 (to-polar c2))
```

```
[POLAR radius 5.656854 angle 0.7853981]
[8]> (setq p3 (to-polar c3))
[POLAR radius 5.0990195 angle 1.7681919]
[9]> (height (make-instance 'triangle :1 p3 :2 p1 :3 p2))
[LINE [POLAR radius 5.0990195 angle 1.7681919] [POLAR radius 2.8284266 angle 0.7853981]]
[10]> (to-cartesian (line-end (height (make-instance 'triangle :1 p3 :2 p1 :3 p2))))
[CART x 1.9999998 y 1.9999996]
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы

Классы систематизируют данные, инкапсулируя их. Обобщенные функции позволяют присваивать одни и те же названия схожим действиям.