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

Студент группы 8О-308Б Ивенкова Любовь, № по списку 6.

Контакты: lyubov.iven@mail.ru
Работа выполнена: 28.05.2022
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 № 5.39)

Определить обычную функцию line-intersections, принимающий один аргумент - список отрезков (экземпляров класса line). Причём концы отрезков могут задаваться как в декартовых (экземплярами cart), так и в полярных координатах (экземплярами polar).

Функция должна возвращать список всех точек взаимного пересечения отрезков между собой.

Результирующие точки могут быть получены либо в декартовых, либо в полярных координатах - на усмотрение выполняющего задание. Точки пересечения отрезков за их границами (как пересечения прямых) должны быть исключены из результирующего списка.

"Вырожденные" случаи: параллельность, взаимное наложение и т.п. - следует исключить из рассмотрения и считать, что такого во входных данных не бывает.

```
(setq lines (list (make-instance 'line
                   :start (make-instance 'cart-или-polar ...)
                   :end (make-instance 'cart-или-polar ...))
                  ...))

(line-intersections lines)
=> (cart-или-polar1 ...)
```

4. Оборудование ПЭВМ студента

PC, процессор Intel Core i5-4460 @ 3,2 GHz (32 x 100), память 11,7 Gb, 64-разрядная система.

5. Программное обеспечение ЭВМ студента

ОС Windows 10, программа LispWorks 7.1 Personal (64-bit).

6. Идея, метод, алгоритм

В начале перебираем всевозможные пары заданных нам отрезков. Для каждой пары находим их точку взаимного пересечения следующим образом.

По координатам начала-конца отрезков можно записать соответствующие им уравнения прямой вида $y = k * x + b$ и найти для них коэффициенты k и b. Потом, используя эти коэффициенты можно найти координаты точки пересечения. В целом нужные формулы выглядят так:

Пусть изначально нам задано два отрезка $S1$ и $S2$. Начало отрезка $S1$ задано точкой $P_{11}(x_{11}, y_{11})$, а конец – точкой $P_{12}(x_{12}, y_{12})$. Начало отрезка $S2$ задано точкой $P_{21}(x_{21}, y_{21})$, а конец – точкой $P_{12}(x_{22}, y_{22})$.

Уравнения прямых, соответствующих исходным отрезкам:

$$y = k_1 * x + b_1$$

$$y = k_2 * x + b_2$$

Уравнения для подсчёта коэффициентов k и b:

$$k_1 = {(y_{12} - y_{11})}\Big/{(x_{12} - x_{11})}$$

$$k_2 = {(y_{22} - y_{21})}\Big/{(x_{22} - x_{21})}$$

$$b_1 = {(x_{12} * y_{11} - x_{11} * y_{12})}\Big/{(x_{12} - x_{11})}$$

$$b_2 = {(x_{22} * y_{21} - x_{21} * y_{22})}\Big/{(x_{22} - x_{21})}$$

Итоговые координаты точки пересечения:

$$x = \frac{(b_2 - b_{1)}}{(k_1 - k_{2)}}$$

$$y = k_1 * x + b_1 = x = k_1 * \frac{(b_2 - b_{1)}}{(k_1 - k_{2)}} + b_1$$

В конце проверяем, что наша точка принадлежит обоим отрезкам, то есть координата x принадлежит одновременно диапазонам значений $[x_{11}, x_{12}]$ и $[x_{21}, x_{22}]$, а координата y принадлежит одновременно диапазонам значений $[y_{11}, y_{12}]$ и $[y_{21}, y_{22}]$.

7. Сценарий выполнения работы

В начале я взяла из курса описание классов cart, polar и line, методы для их более понятного вывода, а также методы cart-x и cart-y для преобразования полярных координат к декартовым. Далее я написала основную функцию line-intersections, которая перебирает всевозможные пары входных отрезков и находит их точку пересечения. После этого она добавляет её в результирующий список точек пересечения (перед эти проверяя, что там уже нет точки с такими координатами).

Далее я написала функцию, принимающую на вход два отрезка, и ищущую их точку пересечения по указанным выше формулам. Кроме того, так как в этих формулах присутствует риск деления на 0, то я отдельно рассматривала случаи, когда один или оба

отрезка параллельны какой-либо координатной оси. При этом я исключала случаи вырожденности (когда отрезки параллельны, накладываются друг на друга, или кода координаты начала и конца отрезка совпадают).

В конце я формировала экземпляр класса cart. То есть в итоге мы получаем список точек пересечения в декартовых координатах.

8. Распечатка программы и её результаты

Программа

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

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

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

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

(defun eqcart (f s)
  (let ((x1 (cart-x f))
        (y1 (cart-y f))
        (x2 (cart-x s))
        (y2 (cart-y s)))
       (and (= x1 x2) (= y1 y2))))

(defun line-intersections (lines)
  (let ((tmp lines)
        (int)
        (res))
    (dolist (i lines)
      (setq tmp (rest tmp))
       (dolist (j tmp)
```

```
          (setq int (intersect i j))
          (if int
          (if (null res)
           (setq res (list int))
           (if (not (member int res :test #'eqcart))
               (setq res (append res (list int)))))))))
    res))

(defun intersect (i j)
  (let ((x11 (cart-x (line-start i)))
        (y11 (cart-y (line-start i)))
        (x12 (cart-x (line-end i)))
        (y12 (cart-y (line-end i)))
        (x21 (cart-x (line-start j)))
        (y21 (cart-y (line-start j)))
        (x22 (cart-x (line-end j)))
        (y22 (cart-y (line-end j)))
        (k1)
        (k2)
        (b1)
        (b2)
        (x)
        (y))
       (if (eql x11 x12)
           (let ()
            (setq x x11)
            (if (eql y21 y22)
               (setq y y21)
               (let()
                (setq k2 (/ (- y22 y21) (- x22 x21)))
                (setq b2 (/ (- (* x22 y21) (* x21 y22)) (- x22
x21)))
                (setq y (+ (* x k2) b2)))))
           (if (eql x21 x22)
               (let()
                (setq x x21)
                (if (eql y11 y12)
                  (setq y y11)
                  (let()
                   (setq k1 (/ (- y12 y11) (- x12 x11)))
                   (setq b1 (/ (- (* x12 y11) (* x11 y12)) (- x12
x11)))
                   (setq y (+ (* x k1) b1)))))
                (if (eql y11 y12)
                    (let()
                     (setq y y11)
                     (if (eql x21 x22)
                         (setq x x21)
                         (let()
                          (setq k2 (/ (- y22 y21) (- x22 x21)))
                          (setq b2 (/ (- (* x22 y21) (* x21 y22))
(- x22 x21)))
                          (setq x (/ (- y b2) k2)))))
                    (if (eql y21 y22)
                     (let()
```

```
                      (setq y y21)
                      (if (eql x11 x12)
                         (setq x x11)
                         (let()
                          (setq k1 (/ (- y12 y11) (- x12 x11)))
                          (setq b1 (/ (- (* x12 y11) (* x11 y12))
(- x12 x11)))
                          (setq x (/ (- y b1) k1)))))
                     (let()
                       ;there are no lines parallel to OX or OY
                       (setq k1 (/ (- y12 y11) (- x12 x11)))
                       (setq k2 (/ (- y22 y21) (- x22 x21)))
                       (setq b1 (/ (- (* x12 y11) (* x11 y12)) (-
x12 x11)))
                       (setq b2 (/ (- (* x22 y21) (* x21 y22)) (-
x22 x21)))
                       (setq x (/ (- b2 b1) (- k1 k2)))
                       (setq y (+ (* x k1) b1)))))))
        ;we check that it is the lines (segments) that intersect,
and not their continuations
        (if (and (or (and (and (>= x x11) (<= x x12)) (and (>= x
x21) (<= x x22)))   ;x in [x11,x12] and [x21,x22]
                     (and (and (>= x x12) (<= x x11)) (and (>= x
x21) (<= x x22)))   ;x in [x12,x11] and [x21,x22]
                     (and (and (>= x x11) (<= x x12)) (and (>= x
x22) (<= x x21)))   ;x in [x11,x12] and [x22,x21]
                     (and (and (>= x x12) (<= x x11)) (and (>= x
x22) (<= x x21))))  ;x in [x12,x11] and [x22,x21]
                 (or (and (and (>= y y11) (<= y y12)) (and (>= y
y21) (<= y y22)))   ;y in [y11,y12] and [y21,y22]
                     (and (and (>= y y12) (<= y y11)) (and (>= y
y21) (<= y y22)))   ;y in [y12,y11] and [y21,y22]
                     (and (and (>= y y11) (<= y y12)) (and (>= y
y22) (<= y y21)))   ;y in [y11,y12] and [y22,y21]
                     (and (and (>= y y12) (<= y y11)) (and (>= y
y22) (<= y y21))))) ;y in [y12,y11] and [y22,y21]
        (make-instance 'cart :x x :y y))))
```

Результаты

```
CL-USER 1 > (setq l (list (make-instance 'line
           :start (make-instance 'cart :x 0 :y 0)
           :end (make-instance 'cart :x 2 :y 2))
                     (make-instance 'line
           :start (make-instance 'cart :x 1 :y 0)
           :end (make-instance 'cart :x 1 :y 2))
     (make-instance 'line
           :start (make-instance 'cart :x 0 :y 2)
           :end (make-instance 'cart :x 2 :y 2))))
([ОТРЕЗОК [CART x 0 y 0] [CART x 2 y 2]] [ОТРЕЗОК [CART x 1 y 0]
[CART x 1 y 2]] [ОТРЕЗОК [CART x 0 y 2] [CART x 2 y 2]])

CL-USER 2 > (line-intersections l)
([CART x 1 y 1] [CART x 2 y 2] [CART x 1 y 2])
```

```
CL-USER 3 > (setq l (list (make-instance 'line
           :start (make-instance 'cart :x 2 :y 0)
           :end (make-instance 'cart :x 0 :y 2))
                     (make-instance 'line
           :start (make-instance 'polar :radius 0 :angle 0)
           :end (make-instance 'polar :radius 5 :angle (/ pi
4)))))
([ОТРЕЗОК [CART x 2 y 0] [CART x 0 y 2]] [ОТРЕЗОК [POLAR radius 0
angle 0] [POLAR radius 5 angle 0.7853981633974483D0]])

CL-USER 4 > (line-intersections l)
([CART x 1.0D0 y 1.0D0])

CL-USER 5 > (setq l (list (make-instance 'line
           :start (make-instance 'cart :x 0 :y 0)
           :end (make-instance 'cart :x 2 :y 2))
                     (make-instance 'line
           :start (make-instance 'cart :x 0 :y 2)
           :end (make-instance 'cart :x 2 :y 0))
     (make-instance 'line
           :start (make-instance 'cart :x 1 :y 0)
           :end (make-instance 'cart :x 1 :y 2))))
([ОТРЕЗОК [CART x 0 y 0] [CART x 2 y 2]] [ОТРЕЗОК [CART x 0 y 2]
[CART x 2 y 0]] [ОТРЕЗОК [CART x 1 y 0] [CART x 1 y 2]])

CL-USER 6 > (line-intersections l)
([CART x 1 y 1])

CL-USER 7 > (setq l (list (make-instance 'line
           :start (make-instance 'cart :x 0 :y 0)
           :end (make-instance 'cart :x 2 :y 2))
                     (make-instance 'line
           :start (make-instance 'cart :x -2 :y 1)
           :end (make-instance 'cart :x 0 :y 1))))
([ОТРЕЗОК [CART x 0 y 0] [CART x 2 y 2]] [ОТРЕЗОК [CART x -2 y 1]
[CART x 0 y 1]])

CL-USER 8 > (line-intersections l)
NIL

CL-USER 9 > (setq l (list (make-instance 'line
           :start (make-instance 'cart :x 0 :y 0)
           :end (make-instance 'cart :x 2 :y 2))))
([ОТРЕЗОК [CART x 0 y 0] [CART x 2 y 2]])

CL-USER 10 > (line-intersections l)
NIL
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Возможно, можно было бы найти точку пересечения более рациональным способом, без таких громоздких формул, но я не смогла придумать такого.

11. Выводы

В процессе выполнения работы я познакомилась с классами и методами в языке Common Lisp. Эта работа была довольно легкой, но муторной из-за громоздких формул. При кажущейся легкости изначальных формул мне пришлось отдельно расписывать все возможные случаи параллельности отрезков координатным осям, что значительно увеличило объём кода.