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

Студент группы 8О-308

Куликов Алексей, № по списку 8.

Контакты:
kapitoshka.the.first@gmail.com
Работа выполнена: 01.05.2020
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 № 5.40)

Определить обычную функцию - предикат line-parallel-p,

- принимающий в качестве аргумента список отрезков (экземпляров класса line),
- возвращающий T, если все указанные отрезки параллельны.

Причём концы отрезков могут задаваться как в декартовых (экземплярами cart), так и в полярных координатах (экземплярами polar).

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

(defun line-parallel-p (lines)
  ...)
```

4. Оборудование студента

Ноутбук Lenovo Ideapad 310-15ABR, процессор AMD A12-9700P 2444 MHz, память 8ГБ, 64-разрядная система.

5. Программное обеспечение

ОС Ubuntu 19.10, программа SBCL 1.5.5

6. Идея, метод, алгоритм

Для решения задачи вычисляем угловой коэффициент k первого отрезка в списке. Затем проходим по списку отрезков, для каждого вычисляя угловой коэффициент k1, и сравниваем равны ли k и k1 приближенно. Если угловой коэффициент каждого отрезка в списке окажется приближенно равен угловому коэффициенту первого, то все отрезка параллельны, в противном же случае не параллельны.

Для вычисления углового коэффициента нужно привести точки, составляющие отрезок, к общей системе координат: либо точки заданные в полярной системе к декартовой, либо наоборот. Для этого определяется какого типа конечная или начальная точка отрезка, и если точка типа polar, осуществляется переход к декартовой системе.

На самом деле, удобнее даже приводить из декартовой в полярную т.к. останется только сравнить одну координату, которая и будет угловым коэффициетом, но, к сожалению понял я это на этапе написания отчета, и переписывать код очень не хочется.

Угловые коэффициенты сравниваем приближенно в силу того, что точки отрезков могут быть заданы любыми числами, в том числе с плавающей точкой. Кроме того, даже хорошо представимые в декартовой системе координаты точки, после перехода к полярной системе координат становятся чисслами с плавающей запятой.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

Программа

prog.lisp

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


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


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



(defun polar-to-decart (r phi)
    (list (* r (cos phi)) (* r (sin phi)) )
)
```

```
(defvar eps 0.0001)
(defun approx-equal (a b)
    (< (abs (- a b)) eps)
)

(defun f (k line)
    (let* (
            (start (start line))
            (end (end line))
            (p1 (if (eq (type-of start) 'polar)
                (polar-to-decart (rad start) (angle start) )
                (list (x start) (y start) ))
            )
            (p2 (if (eq (type-of end) 'polar)
                (polar-to-decart (rad end) (angle end) )
                (list (x end) (y end) ))
            )
            (k1
                (/ (- (nth 1 p2) (nth 1 p1)) (- (nth 0 p2) (nth 0 p1)))
            )
        )
        (approx-equal k1 k)
    )
)

(defun aux (k l)
    (if l
        (and (f k (car l)) (aux k (cdr l)))
        t
    )
)

(defun line-parallel-p (lines)
```

```
        (let* (
            (line (car lines))
            (start (start line))
            (end (end line))
            (p1 (if (eq (type-of start) 'polar)
                (polar-to-decart (rad start) (angle start) )
                (list (x start) (y start) ))
            )
            (p2 (if (eq (type-of end) 'polar)
                (polar-to-decart (rad end) (angle end) )
                (list (x end) (y end) ))
            )
            (k
                (/ (- (nth 1 p2) (nth 1 p1)) (- (nth 0 p2) (nth 0 p1)))
            )
        )
        (aux k (cdr lines))
    )
)
```

Результаты

```
$ sbcl --noinform --load prog.lisp
* (defvar lines (list (make-instance 'line
                  :start (make-instance 'cart :x 3 :y 5)
                  :end (make-instance 'cart :x 7 :y 10)
                 )
                 (make-instance 'line
                 :start (make-instance 'cart :x 3 :y 6)
                 :end (make-instance 'cart :x 7 :y 11)
                 )
                 (make-instance 'line
                 :start (make-instance 'polar :rad 2.25 :angle (/ pi 2))
                 :end (make-instance 'cart :x 7 :y 11)
                 )
                 (make-instance 'line
                 :start (make-instance 'polar :rad 13.9283882772 :angle
1.203622493)
                 :end (make-instance 'polar :rad 31.169897337 :angle
1.0317523415)
                 )
             )
)
LINES
```

```
* (line-parallel-p lines)
T

$ sbcl --noinform --load prog.lisp
* (defvar lines (list (make-instance 'line
                  :start (make-instance 'cart :x 3 :y 5)
                  :end (make-instance 'cart :x 7 :y 10)
                 )
                 (make-instance 'line
                 :start (make-instance 'cart :x 3 :y 6)
                 :end (make-instance 'cart :x 7 :y 11)
                 )
                 (make-instance 'line
                 :start (make-instance 'polar :rad 2.25 :angle (/ pi 2))
                 :end (make-instance 'cart :x 7 :y 11)
                 )
                 (make-instance 'line
                 :start (make-instance 'polar :rad 13.9513882772 :angle
1.203622493)
                 :end (make-instance 'polar :rad 31.169897337 :angle
1.0317523415)
                 )
             )
)
LINES
* (line-parallel-p lines)
NIL
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание
1				

10. Замечания автора по существу работы

Довольно интересная задача.

11. Выводы

В данной лабораторной работе я научился определять простейшие классы, порождать экземпляры классов, считывать и изменять значения слотов в языке Коммон Лисп, и, используя полученные знания, решил поставленную задачу. Написанная программа работает правильно и прошла все тесты.