МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра ВСТ

Отчёт

по лабораторной работе 3

по дисциплине

Функциональное и логическое программирование

РУКОВОДИТЕЛЬ:

Емельянов А.А.

СТУДЕНТ:

Докукин Д.В.

Нижний Новгород

**Задание:**

Написать функцию на пересечение

**Код:**

import Data.List

-- тип данных для точки

data Point = Point Double Double deriving Show

-- тип данных для вектора

data Vector = Vector Point deriving Show

-- стандартные единичные вектора

up = Vector (Point 0 1)

right = Vector (Point 1 0)

left = Vector (Point (-1) 0)

down = Vector (Point 0 (-1))

-- тип данных для линии, заданной точкой и вектором

data PVLine = PVLine Vector Point deriving Show

-- тип данных для линии, заданной точкой и нормалью

data PNLine = PNLine Vector Point deriving Show

-- тип данных для линии, заданной точкой и точкой

data PPLine = PPLine Point Point deriving Show

-- класс, обобщающий заданные ранее линии

class Line line where

point :: line -> Point

svector :: line -> Vector

-- линия, заданная вектором и точкой, имеет тип данных класса линии

instance Line PVLine where

point line = case line of

(PVLine \_ point) -> point

svector line = case line of

(PVLine v \_) -> v

-- линия, заданная нормалью и точкой, имеет тип данных класса линии

instance Line PNLine where

point line = case line of

(PNLine \_ point) -> point

svector line = case line of

(PNLine (Vector (Point x y)) \_) -> Vector (Point x1 y1) where

x1 = y

y1 = -x

-- линия, заданная точкой и точкой, имеет тип данных класса линии

instance Line PPLine where

point line = case line of

(PPLine point \_) -> point

svector line = case line of

(PPLine (Point x1 y1) (Point x2 y2)) -> Vector (Point x y) where

x = x2 - x1

y = y2 - y1

-- функция, возвращающая нормаль к вектору

normal :: Line line => line -> Vector

normal line = Vector (Point x y) where

(Vector (Point sx sy)) = svector line

x = sy

y = -sx

-- функция перерсечения линий

cross :: Line a => Line b => a->b-> Maybe Point

cross line1 line2 = result where

Vector vector1 = svector line1 -- берем точку из направляющих векторов линий

Vector vector2 = svector line2 -- для работы с веторами как с точками

unit\_vector1 = signum vector1 -- единичные вектора

unit\_vector2 = signum vector2 -- соответствующих линий

point1 = point line1 -- берем точки из линий

point2 = point line2 -- для дальнейшей работы с ними

result = case (unit\_vector1==unit\_vector2 || (unit\_vector1==(negate unit\_vector2))) of -- проверка на параллельность линий

(True) -> Nothing -- если параллельны, то ничего не возвращаем

(False) -> case (unit\_vector1, unit\_vector2) of -- проверка на вертикальность линий

-- проверка, возможно ли деление на х (координата x вектора)

((Point 0 1), \_) -> Just (Point x y) where -- первая линия вертикальна и направляющий вектор направлен вверх

x = x0 where

Point x0 y0 = point1

k2 = y0/x0 where

Point x0 y0 = vector2

b2 = y0 - (k2 \* x0) where

Point x0 y0 = point2

y = k2 \* x + b2

(\_, (Point 0 1)) -> Just (Point x y) where -- вторая линия вертикальна и направляющий вектор направлен вверх

x = x0 where

Point x0 y0 = point2

k1 = y0/x0 where

Point x0 y0 = vector1

b1 = y0 - (k1 \* x0) where

Point x0 y0 = point1

y = k1 \* x + b1

((Point 0 (-1)), \_) -> Just (Point x y) where -- первая линия вертикальна и направляющий вектор направлен вниз

x = x0 where

Point x0 y0 = point1

k2 = y0/x0 where

Point x0 y0 = vector2

b2 = y0 - (k2 \* x0) where

Point x0 y0 = point2

y = k2 \* x + b2

(\_, (Point 0 (-1))) -> Just (Point x y) where -- вторая линия вертикальна и направляющий вектор направлен вниз

x = x0 where

Point x0 y0 = point2

k1 = y0/x0 where

Point x0 y0 = vector1

b1 = y0 - (k1 \* x0) where

Point x0 y0 = point1

y = k1 \* x + b1

(\_, \_) -> Just (Point x y) where --вместе обе прямые не вертикальны

-- уравнение прямой y = kx + b

k1 = y0/x0 where -- коэффициент наклона первой прямой

Point x0 y0 = vector1

k2 = y0/x0 where -- коэффициент наклона второй прямой

Point x0 y0 = vector2

b1 = y0 - (k1 \* x0) where -- свободный член b первой прямой

Point x0 y0 = point1

b2 = y0 - (k2 \* x0) where -- свободный член b второй прямой

Point x0 y0 = point2

x = (b2 - b1)/(k1 - k2) -- координату пересечения по оси x

-- (k1 \* x) + b1 = (k2 \* x) + b2

y = k1 \* x + b1 -- подставляем x в уравнение первой прямой

-- точка имеет тип данных класса сравниваемые

instance Eq Point where

(==) (Point x1 y1) (Point x2 y2) = (x1 == x2 && y1 == y2)

-- точка имеет тип данных класса числовых

instance Num Point where

(+) (Point x1 y1) (Point x2 y2) = Point (x1+x2) (y1+y2)

(-) (Point x1 y1) (Point x2 y2) = Point (x1-x2) (y1-y2)

(\*) (Point x1 y1) (Point x2 y2) = Point (x1\*x2 - y1\*y2) (x1\*y2 + x2\*y1)

negate (Point x y) = Point (0-x) (0-y)

abs (Point x y) = Point (sqrt(x^2 + y^2)) 0

signum (Point x y) = Point (x/p) (y/p) where

Point p \_ = abs (Point x y)

fromInteger a = Point (fromInteger a) 0