Министерство образования Республики Беларусь

Учреждение образования «Полоцкий государственный университет имени Ефросинии Полоцкой»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа № 3**

по дисциплине «Функциональное программирование»

Вариант 3

Выполнил: Савчук Д. А., студент гр. 23-ИТ-2, ФИТ

Проверил: Старший преподаватель кафедры ТП Забелендик О.Н.

Задание

Определите тип, представляющий геометрические фигуры на плоскости.

Фигура может быть либо окружностью (характеризуется координатами центра и радиусом), прямоугольником (характеризуется координатами верхнего левого и нижнего правого углов), треугольником (координаты вершин) и текстовым полем (для него необходимо хранить положение левого нижнего угла, шрифт и строку, представляющую надпись). Шрифт задается из множества трех возможных шрифтов: Courier, Lucida и Fixedsys.

Определите следующие функции:

1) Функция area, возвращающая площадь фигуры. Для текстового поля площадь зависит от высоты и ширины буквы в шрифте. Поскольку выбранные нами шрифты – моноширинные (т.е. ширина всех букв в них одинакова), вам необходимо также определить вспомогательную функцию, для каждого шрифта воз- вращающую его габариты.

2) Функция getRectangles, из списка фигур выбирающая только прямоугольники.

3) Функция getBound, по заданной фигуре возвращающая ограничивающий ее прямоугольник.

4) Функция getBounds, по списку фигур возвращающая ограничивающий ее прямоугольник.

5) Функция getFigure, по заданному списку фигур и координатам точки возвращающая первую фигуру, для которой точка попадает в ее ограничивающий прямоугольник. Используйте тип Maybe для возвращаемого значения.

6) Функция move, по заданной фигуре и вектору сдвига возвращающая новую фигуру, сдвинутую относительно заданной на указанный вектор.

Описание проделанной работы

В рамках проделанной работы были определены Haskell типы данных для представления геометрических сущностей: Point (точка), Vector (вектор), Font (шрифт с вариантами Courier, Lucida, Fixedsys) и Figure (фигура, включающая Circle, Rectangle, Triangle, TextBox), представлено в листинге 1.

Листинг 1 – Типы данных

1. data Figure
2. = Circle Point Double
3. | Rectangle Point Point
4. | Triangle Point Point Point
5. | TextBox Point Font String
6. deriving (Show, Eq)

Этот код определяет алгебраический тип данных Figure в Haskell, который представляет различные геометрические фигуры. Он может принимать одну из четырех форм (конструкторов): Circle, определяемый центром (Point) и радиусом (Double); Rectangle, заданный двумя точками (Point), обычно противоположными углами; Triangle, определенный тремя точками (Point) вершин; и TextBox, характеризующийся положением (Point), шрифтом (Font) и текстом (String). Ключевое слово deriving (Show, Eq) автоматически создает для типа Figure возможность преобразования его значений в строки для вывода и сравнения их на равенство.

Была реализована функция area для вычисления площади любой фигуры, причем для TextBox площадь рассчитывается на основе габаритов шрифта, получаемых с помощью вспомогательной функции getFontDimensions, представлено в листинге 2.

Листинг 2 – Area

1. dist :: Point -> Point -> Double
2. dist (x1, y1) (x2, y2) = sqrt $ (x2 - x1)^2 + (y2 - y1)^2
3. area :: Figure -> Double
4. area (Circle \_ r) = pi \* r \* r
5. area (Rectangle (x1, y1) (x2, y2)) = abs $ (x2 - x1) \* (y2 - y1)
6. area (Triangle p1 p2 p3) = -- Формула Герона
7. let a = dist p1 p2
8. b = dist p2 p3
9. c = dist p3 p1
10. s = (a + b + c) / 2.0
11. in if s <= a || s <= b || s <= c
12. then 0.0
13. else sqrt $ s \* (s - a) \* (s - b) \* (s - c)
14. area (TextBox \_ font text) =
15. let (h, w) = getFontDimensions font
16. len = fromIntegral $ length text
17. in len \* w \* h
18. getFontDimensions :: Font -> (Double, Double)
19. getFontDimensions Courier = (12.0, 7.0)
20. getFontDimensions Lucida = (10.0, 6.0)
21. getFontDimensions Fixedsys = (14.0, 8.0)

Листинг 3 – GetBound

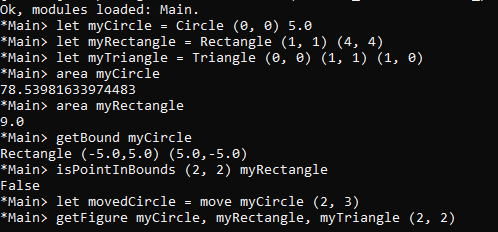
1. getBound :: Figure -> Figure
2. getBound (Circle (cx, cy) r) = Rectangle (cx - r, cy + r) (cx + r, cy - r)
3. getBound (Rectangle p1 p2) =
4. let ((minX, maxY), (maxX, minY)) = normalizeRect p1 p2
5. in Rectangle (minX, maxY) (maxX, minY)
6. getBound (Triangle (x1, y1) (x2, y2) (x3, y3)) =

Продолжение листинга 3

1. let minX = minimum [x1, x2, x3]
2. maxX = maximum [x1, x2, x3]
3. minY = minimum [y1, y2, y3]
4. maxY = maximum [y1, y2, y3]
5. in Rectangle (minX, maxY) (maxX, minY)
6. getBound (TextBox (x, y) font text) =
7. let (h, w) = getFontDimensions font
8. len = fromIntegral $ length text
9. Bottom-left is (x, y)
10. Top-left is (x, y + h)
11. Bottom-right is (x + len \* w, y)
12. Top-right is (x + len \* w, y + h)
13. topLeft = (x, y + h)
14. bottomRight = (x + len \* w, y)
15. in Rectangle topLeft bottomRight

В листинге 3 представлена функция getBound, которая определяет способ вычисления минимального ограничивающего прямоугольника для различных типов фигур. Для Circle она создает прямоугольник, касающийся окружности сверху, снизу, слева и справа, используя координаты центра и радиус. Для Rectangle она нормализует заданные углы (с помощью не показанной здесь функции normalizeRect), чтобы гарантированно получить верхний левый и нижний правый углы, и возвращает этот нормализованный прямоугольник. Для Triangle она находит минимальные и максимальные координаты x и y среди трех вершин и формирует из них новый Rectangle. Наконец, для TextBox она вычисляет ширину и высоту текстового поля на основе размеров шрифта и длины строки, а затем, предполагая, что заданная точка является левым нижним углом, конструирует соответствующий ограничивающий Rectangle.

**Результат выполнения работы**



**Вывод**

В ходе работы был успешно разработан и реализован модуль на языке Haskell для представления и манипулирования двумерными геометрическими фигурами, включая окружности, прямоугольники, треугольники и текстовые поля с различными шрифтами. Используя алгебраические типы данных и функциональный подход, были созданы функции для вычисления площади фигур, определения их ограничивающих прямоугольников (как для отдельных фигур, так и для их списков), фильтрации по типу, проверки попадания точки в границы фигуры и перемещения фигур на плоскости по заданному вектору. Итоговая реализация представляет собой функциональное решение поставленной задачи, демонстрирующее возможности Haskell в области определения структур данных и реализации алгоритмов геометрической обработки.