МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ ИМЕНИ АКАДЕМИКА М.Ф. РЕШЕТНЕВА”

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**Отчёт по Лабораторной работе № 4**По дисциплине “Функциональное программирование”

**Вариант № 12**

**Выполнил:** студент 1 курса  
Группы: БПИ22-02  
Трифонов Кирилл Вячеславович

**Преподаватель:**   
доцент каф. ИВТ, к.т.н

Богданов Константин Валериевич

**Красноярск 2023**

Задания на лабораторную работу 4.

Определите тип, представляющий геометрические фигуры на плоскости. Фигура может быть либо окружностью (характеризуется координатами центра и радиусом), прямоугольником (характериуется 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.

*Определение типа фигур и шрифтов:*

data *Font* = Courier | Fixedsys | Lucida deriving (*Eq*, *Show*)

data *Figure* = Circle *Float* *Float* *Float* | Rectangle *Float* *Float* *Float* *Float* | Triangle *Float* *Float* *Float* *Float* *Float* *Float*

              | Text  *Float* *Float* *Font* *String* deriving (*Eq*, *Show*)

**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 одинаково), вам необходимо**

**также определить вспомогательную функцию, для каждого шрифта возвращающую его габариты.**

*--Размеры шрифтов*

getLetterSize Courier  = 8

getLetterSize Fixedsys = 10

getLetterSize Lucida   = 12

*--(1) Площадь фигуры*

area (Rectangle x1 y1 x2 y2) = (x2-x1)\*(y2-y1)

area (Circle \_ \_ r) = pi\*r\*r

    where pi=3.14

area (Triangle x1 y1 x2 y2 x3 y3) = x1\*x2-y1\*y2

area (Text \_ \_ f s) = (fromIntegral (length s)) \* (getLetterSize f)

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

*--(2) Выборка прямоугольников*

getRectangles :: [*Figure*] -> [*Figure*]

getRectangles [] = []

getRectangles ((Rectangle x1 y1 x2 y2):fs) = (Rectangle x1 y1 x2 y2) : (getRectangles fs)

getRectangles ((Circle \_ \_ \_):fs)          = getRectangles fs

getRectangles ((Triangle \_ \_ \_ \_ \_ \_):fs)  = getRectangles fs

getRectangles ((Text \_ \_ \_ \_):fs)       = getRectangles fs

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

*--(3) Граница 1 фигуры*

getBound :: *Figure* -> *Figure*

getBound (Rectangle x1 y1 x2 y2)     = (Rectangle x1 y1 x2 y2)

getBound (Circle x y r)              = (Rectangle (x-r) (y-r) (x+r) (y+r))

getBound (Text x y f s)           = (Rectangle x (y-(getLetterSize f)) (x+(fromIntegral (length s)) \* (getLetterSize f)) y)

getBound (Triangle x1 y1 x2 y2 x3 y3)= (Rectangle (min x1 (min x2 x3)) (max y1 (max y2 y3)) (max x1 (max x2 x3)) (min y1 (min y2 y3)))

**4) Функция getBounds, по списку фигур возвращающая список их ограничивающих прямоугольников.**

*--(4) Массив границ массива фигур*

getBounds :: [*Figure*] -> [*Figure*]

getBounds [] = []

getBounds (f:fs) = (getBound f) : (getBounds fs)

**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.**

Вспомогательные функции нахождения углов:

*--функции нахождения углов*

getX1 :: *Figure* -> *Maybe* *Float*

getX1 (Rectangle x1 \_ \_ \_ )  = Just x1

getX1 (Triangle \_ \_ \_ \_ \_ \_) = Nothing

getX1 (Circle  \_ \_ \_)        = Nothing

getX1 (Text \_ \_ \_ \_)      = Nothing

getX2 :: *Figure* -> *Maybe* *Float*

getX2 (Rectangle \_ \_ x2 \_)   = Just x2

getX2 (Triangle \_ \_ \_ \_ \_ \_) = Nothing

getX2 (Circle  \_ \_ \_)        = Nothing

getX2 (Text \_ \_ \_ \_)      = Nothing

getY1 :: *Figure* -> *Maybe* *Float*

getY1 (Rectangle \_ y1 \_ \_ )  = Just y1

getY1 (Triangle \_ \_ \_ \_ \_ \_) = Nothing

getY1 (Circle  \_ \_ \_)        = Nothing

getY1 (Text \_ \_ \_ \_)      = Nothing

getY2 :: *Figure* -> *Maybe* *Float*

getY2 (Rectangle \_ \_ \_ y2)   = Just y2

getY2 (Triangle \_ \_ \_ \_ \_ \_) = Nothing

getY2 (Circle  \_ \_ \_)        = Nothing

getY2 (Text \_ \_ \_ \_)      = Nothing

*------*

*--(5) Поиск в границах*

getFigure :: [*Figure*] -> (*Float*,*Float*) -> *Maybe* *Figure*

getFigure [] (\_,\_) = Nothing

getFigure (fg:fgs) (xx,yy) |  ((xx <= x2) && (xx >= x1) && (yy <= y1) && (yy >= y2)) = Just fg

                           |  otherwise = getFigure fgs (xx,yy)

                              where x1=(fromJust (getX1 (getBound fg)));

                                    x2=(fromJust (getX2 (getBound fg)));

                                    y1=(fromJust (getY1 (getBound fg)));

                                    y2=(fromJust (getY2 (getBound fg)))

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

*--(6) Перемещение*

move :: *Figure* -> *Float* -> *Float* -> *Figure*

move (Rectangle x1 y1 x2 y2) dx dy      = (Rectangle (x1+dx) (y1+dy) (x2+dx) (y2+dy))

move (Circle x y r) dx dy               = (Circle (x+dx) (y+dy) r)

move (Text x y f s) dx dy            = (Text (x+dx) (y+dy) f s)

move (Triangle x1 y1 x2 y2 x3 y3) dx dy = (Triangle (x1+dx) (y1+dy) (x2+dx) (y2+dy) (x3+dx) (y3+dy))

Для проверки значений используется функция:

*--Проверка a на существование*

fromJust (Just a) = a

fromJust Nothing  = error "Unknown error"