Проект на лабораторные работы

В рамках лабораторных работ вам предлагается реализовать законченный проект, включающий в себя:

1. Приложение, разработанное в соответствии с заданием.
2. Систему сборки.
3. Unit-тесты.
4. Настроенные автоматические сборки (CI).

Работа над проектом разбивается на короткие итерации. Процесс разработки должен быть зафиксирован в системе контроля версий (git). Проект должен быть опубликован на github.

Вам предлагается выполнить проект «геометрия»:

> Нужно реализовать операции с геометрическими фигурами: ввод и вывод в одном или нескольких текстовых форматах, вычисление площади, периметра и факта пересечения двух выбранных фигур. Сложность проекта регулируется за счет добавления дополнительных типов фигур.

В проекте возможно несколько вариантов считывания входных данных.

1. Считывание со стандартного потока ввода. Наиболее простой в реализации, но наименее удобный вариант. Ввод больших массивов данных слишком трудоемок, поэтому удобнее хранить их в файле, а для ввода использовать перенаправление.
2. Считывание данных из файла. В этом случае синтаксис запуска приложения выглядит следующим образом:

   ```
   geometry <FILE>
   ```

   При невозможности открыть файл приложение должно выводить человекопонятное сообщение об ошибке и завершаться с ненулевым кодом.

3. Гибридный вариант. Если в командной строке указан путь к файлу, то ходы считываются из него. Иначе приложение считывает данные из `stdin`. Аналогичное поведение можно наблюдать у многих стандартных утилит, например, cat

   Синтаксис запуска в этом случае:

   ```
   geometry [FILE]
   ```

Ниже приводятся полное описание проекта. Отдельно взятая лабораторная работа предполагает реализацию небольшой части проекта.

Геометрия

Приложение принимает на вход геометрические фигуры различных типов в WKT-подобном формате.

Для каждой фигуры приложение определяет:

1. Периметр.
2. Площадь.
3. С какими фигурами пересекается текущая.

Поддерживаемые фигуры в зависимости от уровня сложности:

Easy

Окружность.

Normal

Окружность, треугольник.

Hard

Окружность, треугольник, полигон.

Грамматика (EBNF):

```
Object = 'circle' '(' Point ',' Number ')'
       | 'triangle' '(' '(' Point ',' Point ',' Point ',' Point ')' ')'
       | 'polygon' '(' '(' Point ',' Point ',' Point {',' Point } ')' ')'
Point = Number Number
Number = (* Floating-point number *)
```

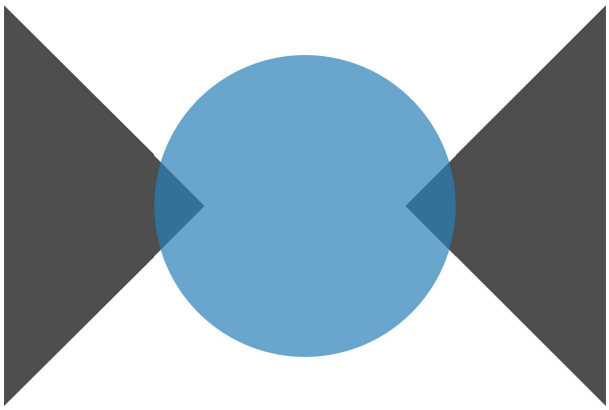
Дополнительные замечания:

1. Типы фигур нечувствительны к регистру (case insensitive).
2. Между токенами может быть произвольное количество пробельных символов.
3. Для простоты можно считать, что одна фигура занимает ровно одну строку. В строке не может быть нескольких фигур.

Пример входных данных:

```
triangle((-3.0 -2, -1 0.0, -3.0 2.0, -3 -2))
circle(0 0, 1.5)
triangle((3 -2.0, 3.0 2, 1.0 0, 3.0 -2))
```

Иллюстрация:



Вывод:

```
1. triangle((-3 -2, -1 0, -3 2, -3 -2))
    perimeter =  9.657
    area = 4
    intersects:
      2. circle

2. circle(0 0, 1.5)
    perimeter = 9.4247
    area = 7.0686
    intersects:
      1. triangle
      3. triangle

3. triangle((3 -2, 3 2, 1 0, 3 -2))
    perimeter = 9.657
    area = 4
    intersects:
      2. circle
```

Приложение должно обрабатывать некорректные входные данные и выводить сообщения об ошибках. Примеры:

```
circlee(1.0 2.0, 3)
^
Error at column 0: expected 'circle', 'triangle' or 'polygon'


circle(x1 2, 3.0)
       ^
Error at column 7: expected '<double>'


circle(1 2, 3.1(
               ^
Error at column 15: expected ')'


circle(1.0 2.1, 3) 123
                   ^
Error at column 19: unexpected token
```