Парсер

Напишите парсер графов в формате DOT.

1. Изучите синтаксис языка DOT
2. Выберите уровень сложности (ниже), в соответствии с которым нужно будет реализовать определенное подмножество языка DOT
3. Реализуйте парсер, который по текстовому представлению построит граф. Для написания парсера используйте библиотеку Parsec
    1. Лекция №18
    2. Документация

Основным результатом задания должна быть функция `graph`:

```
import Text.ParserCombinators.Parsec

data Graph = Digraph ID
             [Node] --список вершин
             [Edge] --список ребер
             [Attr] --список атрибутов графа (сложный уровень)
         deriving (Show)
data Node = Node ID [Attr] --у вершины могут быть атрибуты (сложный уровень)
         deriving (Show)
data Edge = Edge ID ID [Attr] --у ребра могут быть атрибуты (сложный уровень)
         deriving (Show)
data Attr = Attr ID ID
         deriving (Show)
type ID = String

graph :: Parser Graph
graph = undefined
```

Эта функция может быть использована следующим образом:

```
main = do
    text <- readFile "graph.txt"
    let result = parse graph  "" text
    case result of
        Left e -> print e
        Right g -> do
```

```
            print "Read graph:"
            print g
```

Например, для файла со следующим содержимым:

```
digraph graphname {
    a[color=green];
    a -> b -> c
}
```

результат будет:

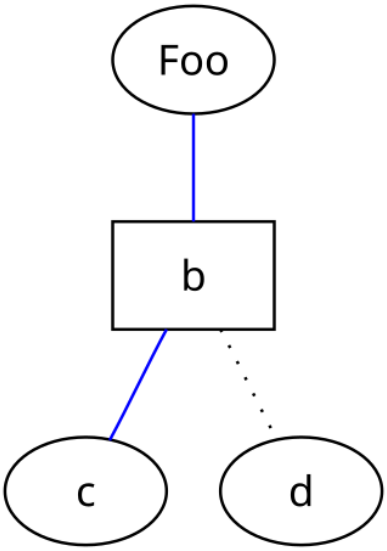
```
ghci> main
"Read graph:"
Digraph "graphname" [Node "a" [Attr "color" "green"],Node "b" [],Node "c" []]
[Edge "a" "b" [],Edge "b" "c" []] []
```

Используйте упрощенный синтаксис, который включает только вершины, ребра и атрибуты (атрибуты могут быть у вершины, ребра и самого графа)

```
graph   :   'digraph' ID  '{' stmt_list '}'
stmt_list   :   [ stmt [ ';' stmt_list ] ]
stmt    :   node_stmt | edge_stmt | ID '=' ID
attr_list   :   '[' [ a_list ] ']'
a_list  :   ID '=' ID [ (';' | ',')  a_list ]
edge_stmt   :   node_id edgeRHS [ attr_list ]
edgeRHS :   edgeop node_id [ edgeRHS ]
node_stmt   :   node_id [ attr_list ]
node_id :   ID
ID: 'a' | 'b' | ... | 'z' | '0' | '1' |...| '9'
edgeop : '->'
```

Например:

```
digraph graphname {
    size=1;
    a [label=foo];
    b [shape=box];
    a -> b -> c [color=blue];
    b -> d [style=dotted];
}
```


Foo
b
c
d