Compiladores

Ficha prática 3 – Lex (continuação)

Num analisador lexical, existem por vezes situações em que o comportamento deve depender do contexto. Por exemplo, se estivermos a interpretar código que possa estar em duas linguagens distintas (por exemplo, Java e Javadoc...), é necessário que efectuemos os efeitos correctos (e distintos) para cada um dos casos. Para que tal aconteça, é fundamental que hajam *delimitadores* que nos ajudem a distinguir os blocos de código.

No caso do Java e Javadoc, temos que a linguagem Java é assumida como *default* num ficheiro Java. Para inserirmos comandos Javadoc, temos que colocar sempre uma entrada "/**", a partir da qual sabemos que, até aparecer um "*/", estamos numa área de Javadoc (veja na figura abaixo um exemplo de Javadoc e Java).

```
* Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
argument is a specifier that is relative
 * to the url argument.
 * 
 * This method always returns immediately, whether or not the image
exists. When this applet attempts to
 * draw the image on the screen, the data will be loaded. The graphics
primitives that draw the image will
 * incrementally paint on the screen.
 * @param url an absolute URL giving the base location of the image
 * @param name the location of the image, relative to the url
arqument
 * @return
               the image at the specified URL
 * @see
                Image
 * /
 public Image getImage(URL url, String name) {
            return getImage(new URL(url, name));
        } catch (MalformedURLException e) {
           return null;
        }
 }
```

Figura 1. Um pequeno programa com Javadoc

Para simplificar situações deste género, o método tipicamente escolhido é usando os *start states*. Um start state corresponde a um "estado" do analisador e serve para o utilizador definir acções dependentes do contexto. Para utilizar start states (imaginemos dois estados, **estado1** e **estado2**), é necessário acrescentar a seguinte linha na secção de definições:

%S ESTADO1 ESTADO2

A utilização dos estados respeita também um conjunto de normas. No início de uma regra, pode-se incluir um estado (significando que essa regra só deve ser considerada se o autómato estiver no estado referido) da seguinte forma:

Para obrigar o algoritmo a mudar para um estado qualquer, utiliza-se a instrução BEGIN <estado>. Para voltar ao estado inicial (ou default do lex), faz-se BEGIN 0.

Usando os start states, podemos definir com exactidão o comportamento de acordo com o contexto, por exemplo para detecção de erros (podemos por exemplo fazer detecção de erros de sintaxe para Javadoc, diferenciada da detecção de erros de sintaxe em Java).

Por exemplo, utilizando o lex, poderá fazer um programa que processa apenas o código Javadoc, nomeadamente os comandos à lá HTML (por exemplo, para mudar de parágrafo), e que ignore a parte em Java:

```
%S JAVADOC
응응
"/**"
                  { BEGIN JAVADOC; }
<JAVADOC>
                  {printf("\n");}
<JAVADOC>"*/"
                  {BEGIN 0; printf("\n");} //regressar ao modo
normal
<JAVADOC>\n
<JAVADOC>.
                  ECHO;
                 ; //Caso seja codigo normal java
\n
                  //nao faz nada
% %
int main()
yylex();
int yywrap()
return 1;
```

Note-se, por exemplo, as entradas assinaladas com (1) e (2): caso se esteja num contexto de Javadoc, o programa copia para o output o caracter encontrado; caso contrário, ignora. Naturalmente, quando se volta ao "estado inicial" (BEGIN 0), o comportamento corresponde aos casos que não têm nenhum start state associado.

1. A conhecida linguagem de edição LaTeX (lê-se "latec"), essencialmente dedicada à elaboração de artigos e relatórios científicos (e para situações onde se pretende alta qualidade gráfica em texto e fórmulas) consiste num conjunto de comandos que definem como deve aparecer o texto no resultado final. Cada comando é sempre precedido do caracter "\".

Um documento em LaTeX deve começar sempre pelo preâmbulo, que começa com a instrução "\documentclass[opcoes]{<Classe do Documento>}", algumas chamadas a packages (comandos do tipo "\usepackage...") até ao início do texto propriamente dito. O texto consiste num bloco delimitado pelo comandos "\begin{document}" e "\end{document}". Cada bloco definido por begin/end é chamado *ambiente* e identificado dentro de chavetas (neste caso o ambiente é "document").

Dentro do texto, existem vários comandos e ambientes específicos a cada situação. Neste exercício, vamos utilizar apenas os comandos "\section", "\chapter", "\title" e os ambientes "itemize" e "enumerate". Em baixo, vemos um ficheiro LaTeX de exemplo:

```
\documentclass[onecolumn, 10pt] {article}
\usepackage{graphix}
\begin{document}
\title{O Gato}
\chapter{Introducao}
Era uma vez um gato maltes...\\
\section{Seccao1}
Tocava piano e falava frances\\
\section{Seccao2}
 ..e dava uns toques de guitarra e falava ingles tambem\\
\chapter{Desenvolvimento}
Na verdade, o gato apenas:
\begin{itemize}
\item Miava
\item Ronronava
\item Fugia dos caes
\end{itemize}
\chapter{Conclusoes}
Moral da historia:
\begin{enumerate}
\item Os animais sao nossos amigos
\item Mesmo quando nao falam frances nem ingles
\item Mesmo quando nao tocam piano nem guitarra
\end{enumerate}
\end{document}
```

e o respectivo resultado poderia ser algo do género:

```
O GATO
1. Introducao
Era uma vez um gato maltes...
1. 1. Seccao1
Tocava piano e falava frances
1. 2. Seccao2
...e dava uns toques de guitarra e falava ingles tambem
2. Desenvolvimento
Na verdade, o gato apenas:
     Miava
     Ronronava
     Fugia dos caes
3. Conclusoes
Moral da historia:
 1.
      Os animais sao nossos amigos
 2.
      Mesmo quando nao falam frances nem ingles
 3.
      Mesmo quando nao tocam piano nem guitarra
```

No sentido de ajudar os menos habituados ao LaTeX (e que pretendem apenas compreender e extrair o conteúdo de um ficheiro ".tex"), pretende-se que faça em Lex um mini-interpretador que produza ficheiros de texto simples com atenção aos seguintes pontos:

- a) Ignore completamente o preâmbulo (ou seja, não deve aparecer no ficheiro de saída)
- b) Coloque o título em letras maiúsculas (antecedido de um tab "\t").
- c) Elimine as indicações de \begin{document} e \end{document}.
- d) Crie capítulos e secções, numerados como apresentado acima.
- e) Interprete os ambientes *itemize* (colocar um ". " no início de cada linha precedida por \item) e *enumerate* (colocar numeração sequencial em cada \item)
- **2.** Para fazer a sua package completa, é necessária também a ferramenta que converta um texto seu em LaTeX. Assim, poderá escrever os seus textos numa linguagem de edição inventada por si e ter o output com a qualidade LaTeX:
 - a) Garanta que o ficheiro de saída começa sempre com os seguintes comandos: \documentclass[a4paper,10pt]{report}

```
\begin{document}
e termina com o seguinte:
\end{document}
```

- b) Converta sequências de maiúsculas seguidas de vários "\t" em comandos "\title{...}"
- c) Converta sequências de dígitos isolados (seguidos de ponto), por exemplo "1." em comandos "\chapter{...}".
- d) Converta sequências de pares de dígitos (intercalados com pontos), por exemplo "1.1." em comandos "\section{...}"
- e) (**opcional**) Converta as frases iniciadas com "." ou "*" como sendo comandos "\itemize"
- f) (**opcional**) Converta as frases iniciadas com *dígito seguidas de* "-" como sendo comandos "\enumerate"

Bibliografia recomendada:

- . Anexo A de Processadores de Linguagens. Rui Gustavo Crespo. IST Press. 1998
- . A Compact Guide to Lex & Yacc. T. Niemann. http://epaperpress.com/lexandyacc/ epaperpress.
- . Manual do lex/flex em Unix (comando "man lex" na shell)