

Processamento de Linguagens

MiEI (3ºano)

Trabalho Prático nº 1 (FLex)

Ano lectivo 19/20

1 Objectivos e Organização

Este trabalho prático tem como principais **objectivos**:

- aumentar a experiência de uso do ambiente Linux e de algumas ferramentas de apoio à programação;
- aumentar a capacidade de escrever *Expressões Regulares (ER)* para descrição de *padrões de frases*;
- desenvolver, a partir de ERs, sistemática e automaticamente *Processadores de Linguagens Regulares*, que filtrem ou transformem textos com base no conceito de regras de produção *Condição-Ação*;
- utilizar o Flex para gerar *filtros de texto em C*.

Neste TP que se pretende que seja resolvido rapidamente (2 semana). Aprecia-se a imaginação/criatividade dos grupos ao incluir outros processamentos!

Deve entregar a sua solução **até Domingo dia 22 de Março**. O ficheiro com o relatório e a solução deve ter o nome 'p119TP1GrNN' — *em breve serão dadas indicações precisas sobre a forma de submissão*.

O programa desenvolvido será apresentado aos membros da equipa docente, totalmente pronto e a funcionar (acompanhado do respectivo relatório de desenvolvimento) e será defendido por todos os elementos do grupo, em data a marcar.

O **relatório** a elaborar, deve ser claro e, além do respectivo enunciado, da descrição do problema, das decisões que lideraram o desenho da solução e sua implementação (incluir a especificação **FLex**), deverá conter exemplos de utilização (textos fontes diversos e respectivo resultado produzido). Como é de tradição, o relatório será escrito em **L^AT_EX**.

2 Enunciados

Para sistematizar o trabalho que se pede em cada uma das propostas seguintes, considere que deve, em qualquer um dos casos, realizar a seguinte lista de tarefas:

1. Especificar os padrões de frases que quer encontrar no texto-fonte, através de ERs.
2. Identificar as acções semânticas a realizar como reacção ao reconhecimento de cada um desses padrões.
3. Identificar as Estruturas de Dados globais que possa eventualmente precisar para armazenar temporariamente a informação que vai extraindo do texto-fonte ou que vai construindo à medida que o processamento avança.
4. Desenvolver um Filtro de Texto para fazer o reconhecimento dos padrões identificados e proceder à transformação pretendida, com recurso ao Gerador **FLex**.

2.1 Template multi-file

Para várias projectos de software, é habitual soluções envolvendo vários ficheiros, várias pastas. Exemplo: um ficheiro, uma makefile, um manual, uma pasta de exemplos, etc.

Pretende-se criar um programa "mkfromtemplate", capaz de aceitar um nome de projecto, e um ficheiro descrição de um template-multi-file e que crie os ficheiros e pastas iniciais do projecto.

No exemplo abaixo apresenta-se um *template-multi-file* para um projecto geral de um filtro flex.

O template inclui:

- metadados (author, email) a substituir nos elementos seguintes
- tree (estrutura de directorias e ficheiros a criar)
- template da cada ficheiro

O metadado "name" vai ser processado via argumento de linha de comando.

```
=== meta

email: jj@di.uminho.pt
author: J.João
# "name" é dado por argumento de linha de comando (argv[1])

=== tree
{%name%}/
- {%name%}.fl
- doc/
-- {%name%}.md
- exemplo/
- Makefile

=== Makefile

{%name%}: {%name%}.fl
        flex {%name%}.fl
        cc -o {%name%} lex.yy.c

install: {%name%}
        cp {%name%} /usr/local/bin/

=== {%name%}.md
# NAME

{%name%} - o nosso fabuloso filtro ...FIXME

## Synopsis

    {%name%} file*

## Description
## See also
## Author

Comments and bug reports to {%author%}, {%email%}.

=== {%name%}.fl
%option noyywrap yylineno
%%

%%
int main(){
    yylex();
```

```
    return 0;
}
```

Modo de executar o programa: `mkfromtemplate name template`.

Como resultado da execução serão criados os ficheiros e directorias descritos em **tree**, com os conteúdos definidos nos templates de ficheiro, e as *variáveis* substituídas.

2.2 Filtro para gramáticas

Considere o seguinte extracto yacc contido num ficheiro de nome `'f.y'`.

```
....
%union ....
%token ID MKLISTA NULA
%type ....
%%
    lista : MKLISTA '(' ids ')' { ... ação semântica em C }
          | NULA
          ;
    ids : ID ',' ids { ignorar as constantes char (ex: '!') das ações semânticas }
         | ID       { if ( ... == 'a') { ... } else { ... } }
         ;
%%
... codigo C...
```

a) **Escreva um filtro flex** `extraigIC` que extraia de `'f.y'` apenas a gramática pura, para ajudar na documentação:

```
    lista : MKLISTA '(' ids ')'
          | NULA
          ;
    ids : ID ',' ids
         | ID
         ;
```

b) Depois de escrever uma gramática, é preciso construir o analisador léxico associado, sendo fácil esquecer algum dos símbolos terminais.

Para ajudar os mais esquecidos, **escreva um filtro flex** `lexgen` que, dada um texto contendo uma gramática em notação yacc, gere o esqueleto de texto de um analisador léxico para essa gramática.

Este filtro deverá procurar a lista dos símbolos terminais da gramática (tokens) – tanto os expressamente definidos como tokens (`%token ...`), como os terminais que aparecem entre apostrofes nas produções da gramática (exemplo `' '`).

A execução de `lexgen f.y` (recorde que `'f.y'` é a gramática yacc apresentada acima) deverá gerar algo como se esquematiza abaixo (Note que a palavra **FIXME** no template será para o utilizador depois substituir pelas expressões regulares que definem os terminais (tokens) se podem escrever na sua linguagem concreta):

```
%%
FIXME {return ID;}
FIXME {return MKLISTA;}
FIXME {return NULA;}
[( ),] {return yytext[0];}
%%
```

2.3 Drum Machine

Em notação abc podemos construir acompanhamentos rítmicos. (ver [abcplus music notation](http://abcplus.musicnotation.com/) para detalhes)

Considere o seguinte exemplo (Página html)http://natura.di.uminho.pt/~jj/pl-20/TP1/abc_ui-perc.html:

```
<!DOCTYPE html> <html> <head lang="en">
  <meta charset="UTF-8">
  <title>$ABC_UI</title>
  <link rel="stylesheet" href="http://dev.music.free.fr/css/music.min.css" />
  <script src="http://dev.music.free.fr/js/abc-ui-1.0.0.min.js"></script>
  <script src="http://dev.music.free.fr/js/music-ui-1.0.0.min.js"></script>
  <script src="http://dev.music.free.fr/soundfonts/percussion-mp3.js"></script>
  <style>pre { border: 2px solid lightgrey; border-radius: 3px; background: #EEE;}</style>
</head>
<body>
  <h4>Afoxé percussion patterns</h4>
  <div class="abc-source">
    X:1
    M:4/4
    L:1/4
    K:C clef=perc
    %%player_no_voice
    %%player_right
    %%map HT G print=d % Agogo bell
    %%map LT _A print=G % Agogo bell
    %%map TD _G print=B % Tumba drum
    %%score [(H L) | T]
    %%stafflines .|. |
    %
    V:H name="Agogo"
    %%MIDI program 128
    %%voicemap HT
    %%pos stem down
    %
    V:L name="bell"
    %%MIDI program 128
    %%voicemap LT
    %%pos stem up
    %
    V:T name="Tumba\ndrum"
    %%MIDI program 128
    %%stafflines 1
    %%voicemap TD
    %%pos stem down
    %
    [V:H] x          x/ G/x/ G/ G | x x G G :|
    [V:L] _A/_A/ z/ x/z/ x/ x | _A _A x x :|
    [V:T] z2          _G _G | z2 _G _G :|
  </div>

<script>
  $ABC_UI.init();
</script>
</body>
</html>
```

Abrindo em HTML, podemos ver e ouvir esta padrão rítmico.

Considere agora os seguintes padrões rítmicos:

[illegible]

Pretende-se arranjar uma notação compacta e intuitiva de os descrever.

Defina uma notação para uma máquina de ritmos, faça um filtro flex que a transforme numa página `$ABC_UI` que os toque.

2.4 Transformador Publico2NetLang

Análise com todo o cuidado o ficheiro http://natura.di.uminho.pt/~jj/pl-20/TP1/Publico_extraction_portuguese_comments_4.html o qual contém os comentários (85 neste exemplo) a uma notícia publicada no jornal O Público, extraídos da página HTML da versão online do dito jornal.

Para se fazer um estudo sócio-linguístico de forma e conteúdo dos comentários que a notícia suscitou, os dados relevantes à análise pretendida devem ser extraídos do ficheiro HTML fornecido e devem ser transformados no formato JSON a seguir mostrado.

```
"commentThread": [
  {
    "id": "STRING",
    "user": "STRING",
    "date": "STRING",
    "timestamp": NUMBER,
    "commentText": "STRING",
    "likes": NUMBER,
    "hasReplies": TRUE/FALSE,
    "numberOfReplies": NUMBER

    "replies": [ ]
  },.....
]
```

Construa então um filtro de texto, recorrendo ao gerador FLex, que realize o processamento explicado, tendo em consideração que as respostas¹ que surjam a um dado comentário devem ser aninhadas, na forma de uma lista, dentro do campo **"replies"** do dito comentário, seguindo evidentemente o mesmo formato apresentado².

2.5 Transformador Sol2NetLang

Analise com todo o cuidado o ficheiro <http://natura.di.uminho.pt/~jj/pl-20/TP1/Sol4.html> o qual contém os comentários a uma notícia publicada no jornal Sol, extraídos da página HTML da versão online do dito jornal.

¹Que também são comentários.

²Note que deve ser um processo recursivo.

Para se fazer um estudo sócio-linguístico de forma e conteúdo dos comentários que a notícia suscitou, os dados relevantes à análise pretendida devem ser extraídos do ficheiro HTML fornecido e devem ser transformados no formato JSON a seguir mostrado.

```
"commentThread": [
  {
    "id": "STRING",
    "user": "STRING",
    "date": "STRING",
    "timestamp": NUMBER,
    "commentText": "STRING",
    "likes": NUMBER,
    "hasReplies": TRUE/FALSE,
    "numberOfReplies": NUMBER

    "replies": [ ]
  },.....
]
```

Construa então um filtro de texto, recorrendo ao gerador FLex, que realize o processamento explicado, tendo em consideração que as respostas³ que surjam a um dado comentário devem ser aninhadas, na forma de uma lista, dentro do campo "replies" do dito comentário, seguindo evidentemente o mesmo formato apresentado⁴.

2.6 Transformador DailyExpress2NetLang

Analise com todo o cuidado o ficheiro <http://natura.di.uminho.pt/~jj/pl-20/TP1/DailyExpress1.html> o qual contém os comentários a uma notícia publicada no jornal inglês Daily Express, extraídos da página HTML da versão online do dito jornal.

Para se fazer um estudo sócio-linguístico de forma e conteúdo dos comentários que a notícia suscitou, os dados relevantes à análise pretendida devem ser extraídos do ficheiro HTML fornecido e devem ser transformados no formato JSON a seguir mostrado.

```
"commentThread": [
  {
    "id": "STRING",
    "user": "STRING",
    "date": "STRING",
    "timestamp": NUMBER,
    "commentText": "STRING",
    "likes": NUMBER,
    "hasReplies": TRUE/FALSE,
    "numberOfReplies": NUMBER

    "replies": [ ]
  },.....
]
```

Construa então um filtro de texto, recorrendo ao gerador FLex, que realize o processamento explicado, tendo em consideração que as respostas⁵ que surjam a um dado comentário devem ser aninhadas, na forma de uma lista, dentro do campo "replies" do dito comentário, seguindo evidentemente o mesmo formato apresentado⁶.

³Que também são comentários.

⁴Note que deve ser um processo recursivo.

⁵Que também são comentários.

⁶Note que deve ser um processo recursivo.