Visão Geral de JFlex e CUP

Rémy De Fru
Franklin Ramalho

Universidade Federal de Campina Grande - UFCG

Agenda

- O que é JFlex?
- Como é um arquivo do JFlex?
- O que é CUP?
- Como é um arquivo do CUP?
- Cuidados a serem tomados

JFlex

- Gerador de analisador léxico
- Arquivo gerado: Analisador léxico (nome indicado no arquivo .flex)
- Faz o mapeamento padrões de entrada para tokens

JFlex

Estrutura

```
package compiler.generated;
import compiler.core.Token;

%%

%class Scanner
%public
%cup

%{
  private Symbol symbol(int type) {
      return new Token(type);
  }

  private Symbol symbol(int type, Object value) {
      return new Token(type, value);
  }
%}

DecimalLiteral = 0 | [1-9][0-9]*
```

Package e imports que devem estar no analisador léxico

Nome do arquivo a ser gerado
Visibilidade do Scanner
Compatibilidade com CUP

Funções em Java a usar durante a geração dos tokens

Pré-definições

Padrões que vão ser usadas posteriormente

JFlex

Continuação da estrutura

```
%%          Sintaxe da linguagem

    "int"                   { return symbol(sym.INT); }
    {DecimalLiteral}        { return symbol(sym.INTEGER_LITERAL, new Integer(yytext())); }
```

Padrão

O que fazer ao encontrar esse padrão

CUP

- Gerador de analisador sintático
- Arquivos gerados: Parser e sym (ou xxx e xxxSym)
- Análise sintática ascendente
- Auxilia para análise semântica e geração de código
- A tabela de símbolos (sym) é gerada a partir dos símbolos terminais

CUP

Estrutura (Desconsiderando análise semântica)

```
import compiler.core.*;

parser code  {:
      // Sobrescrever funções
:};

terminal PLUS, MINUS, TIMES, DIV, EQ, EQEQ;
terminal java.lang.Number INTEGER_LITERAL;

non terminal begin;
non terminal operator;

start with begin;

begin ::= INTEGER_LITERAL operator INTEGER_LITERAL
          | ;

operator ::=
          MINUS | PLUS | TIMES | DIV ;
```

Declaração de terminais e não-terminais

Símbolo inicial

Regras da linguagem

Cuidados

- **Análise sintática**

Em conflitos Shift/Reduce, CUP prefere Shift.
Em conflitos Reduce/Reduce, CUP prefere a primeira regra.
Os conflitos são devido à ambiguidade nas regras.
Evite ambiguidade e use precedência corretamente.

A precedência mais abaixo têm maior prioridade.
A precedência na mesma linha é resolvida por **nonassoc**, **left** ou **right**.

Exemplo:

```
precedence nonassoc EQ;          //   =
precedence nonassoc EQEQ;        //   ==
precedence left     PLUS, MINUS; //   +  -
precedence left     TIMES, DIV;  //   *  /
```

Cuidados

- **Análise sintática**

Não esquecer de :

- Colocar **start with [não-terminal inicial];**
- Declarar todos os terminais usados no arquivo .flex

Referências

- http://www.decom.ufop.br/alex/arquivos/compiladores/tutorialjflex.pdf
- https://johnidm.gitbooks.io/compiladores-para-humanos/content/part2/using-jflex-java-cup.html
- https://www.cs.princeton.edu/~appel/modern/java/CUP/manual.html
- http://jflex.de/manual.html