CUP

CUP

CUP significa Construction of Useful Parsers.

O CUP é um gerador de analisadores sintáticos escrito em Java. O CUP é um gerador de analisador sintático LALR. Ele implementa a geração padrão do analisador LALR(1).

Utilizaremos o CUP em conjunto com o JFlex.

Site oficial do CUP
http://www2.cs.tum.edu/projects/cup/

Manual do CUP
http://www2.cs.tum.edu/projects/cup/docs.php

Exemplos
http://www2.cs.tum.edu/projects/cup/examples.php

CUP

Como escrever uma especificação que será processada pelo CUP ?

A especificação é dividida em quatro seções:

Seção 1: declaração de “packages” e “imports” que serão inseridos no topo do arquivo gerado pelo CUP e diretivas do CUP.

Seção 2: declaração de terminais e não terminais.

Seção 3: precedência e associatividade de terminais.

Seção 4: gramática.

CUP

Seção 1 - Especificação de “packages” e “imports”.

Exemplos:
package compilador.parser;
import compilador.scanner;

Diretivas

- parser code {: ... :};

Permite que você declare variáveis e métodos na classe do parser. Similar à diretiva %{...%} do Jflex.

- init with {: ... :};

O código entre chaves vai ser executado antes que o parser peça o primeiro token ao scanner.

- scan with {: ... :};

Serve para que você escreva o código que o parser vai executar sempre que ele quiser pedir um token ao scanner. Se essa diretiva não for utilizada, o parser chama scanner.next_token() para receber tokens.

CUP

Seção 2 - Lista de símbolos

terminal [classe] nome0, nome1, ...;
non terminal [classe] nome0, nome1, ...;

Em tempo de execução, os símbolos são representados por objetos da classe java_cup.runtime.Symbol. Essa classe possui uma variável chamada “value” que contém o valor do símbolo.

Exemplo:
terminal Integer NUMERO;

Quando o parser recebe do scanner um NUMERO, ele cria um objeto da classe Symbol. A variável “value” será um objeto da classe Integer. Assim, o valor do número pode ser obtido através de simbolo.value.intValue();

Se não for fornecida uma classe na declaração do não-terminal, a variável “value” ficará com valor null.

CUP

Seção 3 - Precedência e Associatividade

precedence left terminal [, terminal...];
precedence right terminal [, terminal...];
precedence nonassoc terminal [, terminal...];

A precedência cresce de cima para baixo, por exemplo:
precedence left ADD, SUBTRACT;
precedence left TIMES, DIVIDE;
Significa que a multiplicação e a divisão têm maior precedência.

A precedência de um operador pode ser forçada em função do contexto em que este aparece. Isso é feito através da diretiva %prec.
Exemplo:

```
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE, MOD;
precedence left UMINUS;
...
 expr ::= MINUS expr:e
       {: RESULT = new Integer(0 - e.intValue()); :}
       %prec UMINUS
```

CUP

Seção 4 - Gramática

Especifica as produções da gramática da linguagem.

start with non terminal; (diretiva opcional)

Indica qual é o não terminal inicial da gramática. Se essa diretiva for omitida, o parser assume o primeiro não terminal declarado nas produções da gramática.

As produções possuem o formato:
não terminal ::= <símbolos e ações>

Os símbolos à direita de “::=“ podem ser terminais ou não-terminais.

As ações correspondem ao código que é executado quando a regra de produção é aplicada.

CUP

Exemplo:

```
expr ::= NUMBER:n
{:
      RESULT=n;
:}
| expr:r PLUS expr:s
{:
      RESULT=new Integer(r.intValue() + s.intValue());
:}
```

Observe que é possível especificar várias produções para um mesmo não-terminal através do uso da barra “|”.
Pode-se nomear símbolos para referenciá-los no código da ação.
O resultado da produção deve ser armazenado na variável implícita “RESULT”.