INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

ECM253 – Linguagens Formais, Autômatos e Compiladores

Atividade 03
Construção de analisadores sintáticos com CUPS

Prof. Marco Furlan

Setembro/2018

1 A atividade

Alterar o projeto de **interpretador** de expressões com **CUP** (em Netbeans), apresentado em aula, `SimpleExpr`, para **atender os requisitos a seguir** (implemente e teste na sequência apresentada, que já está na ordem lógica):

i. **Manipular números reais**. **Sugestão**: no analisador léxico, utilizar a expressão regular:

```
[0-9]+|[0-9]*\.[0-9]+([eE][-+]?[0-9]+)?
```

ii. **Adicionar** na gramática uma **regra** de **atribuição** de variável:

```
expr_part ::= ID ATTRIB expr SEMI
;
```

Onde `ID` é uma expressão regular que casa com qualquer cadeia iniciada por letra maiúscula ou minúscula seguida de zero ou mais letras e números. É necessário alterar o analisador léxico e ter símbolo correspondente no CUP com tipo `String` para `ID`. O símbolo `ATTRIB`, de atribuição, é o terminal '=' . A atribuição é um operador cuja precedência é a mais baixa dos operadores e cuja associatividade é da direita para a esquerda. Deve-se declarar este símbolo no CUP e utilizar no JFlex como outros símbolos de operadores.

iii. **Implementar** um **sistema** de **ambiente** (uma classe com um `Hashtable` é suficiente) para que, assim que uma **atribuição correta** for analisada pelo analisador sintático, o **nome** da variável e o seu **valor** sejam **armazenados** neste **ambiente**. **Sugestão:** no arquivo `Parser.cup`, adicione uma seção de código logo após os `imports`:

```
action code
{:
   private Hashtable<String, Double> environment = new Hashtable<>();
:}
```

Não esquecer de importar `java.util.Hashtable`. Depois, utilizar os métodos `put()`, `get()`, `containsKey()` etc desta classe convenientemente. **Referência**: https://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html.

iv. **Adicionar** uma **regra** à **gramática** para **consultar valores** de variáveis e também permitir que elas sejam utilizadas em expressões. Basta adicionar uma regra assim (a ação semântica associada, que retorna o valor da variável, deve ser adicionada):

```
expr ::= ID
;
```

Depois, pensar em uma solução para retornar seu valor à expressão ou gerar um erro, caso a variável não esteja presente no ambiente.

v. **Adicionar** as **funções matemáticas** `sin(x)`, `cos(x)`, `exp(x)`, onde `x` é qualquer expressão da gramática. Será necessário "filtrar" identificadores no analisador léxico e retornar terminais adequados ao analisador sintático. A gramática também deverá ser alterada para que se possa contemplar expressões deste tipo (pense!).

2 Instalação do projeto do Moodle

Baixar do Moodle o projeto original (em Netbeans) que foi apresentado em sala de aula:

1. Salvar o arquivo `SimpleExpr-NetBeans.zip` do Moodle;
2. Abrir o Netbeans;
3. Executar o menu `Arquivo|Importar Projeto|De ZIP...` e, depois, na caixa de diálogo que aparecer, escolher o diretório destino de seu projeto;
4. Para executar os alvos do **Ant**: clique na aba `Arquivo` (ao lado da aba `Projeto`) e, depois, clique com o botão direito sobre o arquivo `build.xml` e então selecionar o item de menu `Executar destino` e então selecionar o alvo desejado;
5. Para executar o projeto (admitindo que o arquivo compilado está na pasta `dist` e a biblioteca do CUP está em `tools` – altere se forem utilizadas pastas diferentes):
   - No Windows: `java -cp simple_expr.jar;../tools/java-cup-11b.jar parser.Main`
   - No Linux/Mac: `java -cp simple_expr.jar:../tools/java-cup-11b.jar parser.Main`

3 O que enviar?

O projeto modificado e compactado no formato ZIP.