Trabalho 2ª parte - Análise sintática

Implementação

A implementação da análise sintática deverá ser realizando utilizando o mesmo conjunto de ferramentas compatíveis utilizado na análise léxica (apresentados na próxima seção).

A análise sintática deverá ser implementada baseada em uma descrição formal (GLC - Gramática Livre de Contexto) constando na documentação. Os símbolos analisados sintaticamente poderão ser obtidos por meio de uma função `getToken`obtido da análise léxica ou um método apropriado existente na biblioteca utilizada para o desenvolvimento das duas primeiras análises.

A estrutura da árvore sintática já deverá estar empregada. Para cada derivação da análise sintática, deverá ser observado a necessidade de adicionar um nó na árvore sintática. No final da implementação desta primeira parte do trabalho, uma **árvore sintática abstrata** (AST) deverá estar montada na memória.

Linguagens de programação para a implementação

Para a implementação do compilador, pode ser utilizado qualquer linguagem de programação. É recomendado que seja utilizado uma linguagem que dê suporte à estruturas de dados de alto nível e preferencialmente que exista bibliotecas para a construção da varredura e gramática. Algumas LPs/ferramentas conhecidas são:

- `C/C++ - Flex/Bison` [3][4]
- `Python - PLY` (que possui ferramenta léxica e sintática) [5]

Documentação

A documentação nesta etapa do trabalho deverá conter os seguintes itens:

- A descrição da gramática no padrão BNF;
- Discutir o formato na análise sintática realizado pela ferramenta (se é LL(1), LR(1), LALR(1), SLR(1), etc)

Avaliação

A avaliação da análise sintática será realizada da seguinte forma:

Programa de exemplo `T++` de **entrada** na linha de comando. **Saída** será a validação da análise sintática. Além disto uma **representação da árvore sintática abstrata** (ASA) deverá ser mostrada na saída do programa. A saída da análise sintática por meio de uma ASA é de extrema importância para a

continuação da terceira parte do trabalho (análise sintática). Deverá ser possível verificar erros `shift-reduce` e `reduce-reduce` (debuggers deverão estar ativos para a realização da avaliação).

É importante que uma classe de análise de erros já esteja preparada. Os erros mais comuns em desenvolvimento de programas estão relacionados com a análise sintática.

Entrega e apresentação

O trabalho será **individual** e deverá ser entregue até o dia 04/10/2016 no moodle da disciplina em um pacote compactado. Deverá ser especificado na entrega o mecanismo de execução do *parser* para a realização da correção.

Referências

[1] LOUDEN, Kenneth C. Compiladores: princípios e práticas. São Paulo, SP: Thomson, c2004. xiv, 569 p. ISBN 8522104220.

[2] http://web.cecs.pdx.edu/~mpj/jacc/

[3] http://flex.sourceforge.net/

[4] http://www.gnu.org/software/bison/

[5] http://www.dabeaz.com/ply/