Especificação do Trabalho 2 – Analisador sintático

Cada equipe deve desenvolver um analisador sintático na linguagem C que receba um nome de arquivo como parâmetro. Este deve ser um arquivo de código fonte na linguagem definida para a turma COMPILADORES.2014.2. O exemplo abaixo mostra como o programa será testado.

Ex:
$ ./a.out teste.in > tmp
$ diff teste.out tmp

O arquivo “teste.in” é o arquivo de entrada e o arquivo “teste.out” é a saída esperada. Caso seja encontrada alguma diferença entre a saída do programa e a saída esperada, o teste será considerado incorreto. A análise pode ser feita através dos métodos SLR1, LL1 e LL0, valendo as respectivas notas máximas: 100, 70 e 40.

O trabalho deverá ser enviado para o e-mail “mpamplona2.ufba@gmail.com”, com o assunto “TRABALHO 2 – COMPILADORES.2014.2”. Este e-mail deve conter um único anexo de nome “trabalho2.tgz”. Dentro deste arquivo compactado deve haver uma pasta com o nome “trabalho2” que contém o código fonte e um arquivo Makefile que gere um executável chamado “slr1”, “ll1”, “ll0” ou “a.out” no diretório “trabalho2”. O nome do executável indica qual foi o método implementado, e “a.out” deve ser usado para outros métodos, que devem ser explicados e justicados em um arquivo “justificativa.txt”. Nele, o aluno deve também **sugerir** qual seria a nota máxima para o método implementado e justificar. Cada equipe deve utilizar o mesmo arquivo de identificação do primeiro trabalho no diretório “trabalho2”. Por fim, a gramática também deve ser fornecida no arquivo “gramatica.txt”, seguindo o padrão a ser especificado.

O programa deve executar a análise léxica já implementada seguindo a especificação do trabalho 1, e, caso não haja nenhum erro léxico, deve executar a análise sintática, caso contrário, deve listar os erros léxicos conforme especificado anteriormente. Caso a entrada seja sintaticamente válida, o programa deve imprimir “SIM\n”. Caso contrário, deve imprimir “NAO\n”.

Para a descrição da gramática (gramatica.txt), símbolos não-terminais devem ser precedidos do símbolo '#'. O símbolo '$' indicará final de sentença. O símbolo '&' indicará produção vazia. Use “->” para identificar produções. Todas as gramáticas devem ter o estado inicial “#SS -> #S $”. Não é permitido ter múltiplas produções em uma única linha. Use apenas um espaço em branco para separar cada elemento de um produção. Nomeie as produções com o padrão “rX”, sendo X o número da produção começando em 0, como mostrado abaixo:

Ex:
r0 #SS -> #S $
r1 #S -> c #A a
r2 #A -> c #B
r3 #A -> #B
r4 #B -> b c #B
r5 #B -> &

Casos de teste:

Entrada 1:
```
declare a123, b456, 56dfg as letter.
REAL a, b.
a = c*alfa*0,345;
RETORNAR '\a'.
IMPRIME b
sur%Gnsa$trq<<a123.
```

Saída 1:
```
LINHA 1: 56dfg
LINHA 3: ;
LINHA 4: '\a'
LINHA 6: $
```

Entrada 2:
```
declare a, b as letter.
IF a < b THEN [
        put C in d .
]
```

Saída 2:
```
SIM
```

Entrada 3:
```
declare a, b as letter.
IF a + b THEN [
        put C in d .
]
```

Saída 3:
```
NAO
```