Especificação do Trabalho 3 – Analisador semântico

Cada equipe deve desenvolver um analisador semântico 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.

O trabalho deverá ser enviado para o e-mail “mpamplona2.ufba@gmail.com”, com o assunto “TRABALHO 3 – COMPILADORES.2014.2”. Este e-mail deve conter um único anexo de nome “trabalho3.tgz”. Dentro deste arquivo compactado deve haver uma pasta com o nome “trabalho3” que contém o código fonte e um arquivo Makefile que gere um executável chamado “a.out” no diretório “trabalho3”. Cada equipe deve incluir o mesmo arquivo de identificação dos dois trabalhos anteriores no diretório “trabalho3”.

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 inválida, o programa deve imprimir “NAO\n”. Caso contrário, deve imprimir “SIM\n” e seguir para a análise semântica. Caso a entrada seja semanticamente válida, o programa deve imprimir “SIM\n”. Caso contrário, deve imprimir “NAO\n” .

São considerados erros semânticos:

1) Não declarar variáveis (ex: “declare a as letter. put a in b.”)
2) Redeclarar variáveis (ex: “declare a as letter. declare a as number.”)
3) Usar vetores não alocados (ex: “declare a,b[] as number. put a in b[0].”)
4) Alocar variáveis que não sejam vetores (“declare a as letter. resize a to 10.”)
5*) Indexar variáveis que não sejam vetores (“declare a,b as number. put b[0] in a.”)
6**) Não indexar variáveis que sejam vetores (“declare a[],b as number. resize a to 10. put a in b.”)
7) Utilizar variáveis que não sejam vetores como vetores em loops vetoriais (“declare a,b as number. foreach a in b do []”)
8) Destino da atribuição de constante do tipo string não é vetor de letter (“declare a[] as number. resize a to 10. put "abc" in a.”)

* Em expressões aritméticas, como destino de uma atribuição, como iterador em um FOR (“FOR $iterador$ FROM $ea1$ TO $ea2$ DO”), e como elemento em um FOREACH (“FOREACH $elemento$ IN $vetor$ DO”).
** Em expressões aritméticas, como destino de uma atribuição (exceto para atribuição de constante do tipo string), como iterador em um FOR (“FOR $iterador$ FROM $ea1$ TO $ea2$ DO”), e como elemento em um FOREACH (“FOREACH $elemento$ IN $vetor$ DO”).

FIQUE ATENTO às declarações em contextos diferentes. Uma declaração feita dentro de um contexto

não existe fora dele. Na atribuição do exemplo abaixo, a variável “a” deve ser vista como uma variável não declarada:

“declare i as number. for i from 0 to 10 do [declare a as letter.] put '\n' in a.”

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:
NAO

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

Saída 3:
SIM
NAO

Entrada 4:
declare a, B, c, d as letter.
IF a < b THEN [
    put C in d .
]

Saída 4:
SIM
SIM