Centro Universitário de Belo Horizonte – Uni-BH
Departamento de Ciências Exatas e Tecnologia
Curso: Ciência da Computação
Disciplina: Teoria da Computação e Linguagens Formais
Professor: Moisés Henrique Ramos Pereira

Trabalho Prático I – Implementação de um AFD

Descrição do trabalho

Um compilador é um programa de computador que transforma um programa em uma linguagem fonte (chamado de código-fonte) para uma linguagem alvo (linguagem objeto, linguagem de máquina, código-objeto).

A linguagem fonte deve ser uma linguagem formal. Sendo assim, ela consiste de um alfabeto e possui regras bem definidas para a formação de programas. Os símbolos do alfabeto de uma linguagem fonte (uma linguagem de programação, por exemplo), bem como derivações resultantes da combinação desses símbolos (terminais), são chamados *tokens*. Os tokens de *Java*, por exemplo, são identificadores, comando *if*, constantes reais, constantes inteiras, os operadores, etc.

Os *tokens* da linguagem, por sua vez, são conjuntos de lexemas que são instâncias daquele token. Exemplificando, o token identificador possui vários lexemas: *nome*, *idade*, etc. Todos os nomes que atribuímos às variáveis são lexemas do token identificador.

Os padrões de formação dos lexemas de cada *token* são expressões regulares que determinam como avaliar se um lexema é de um determinado *token*. Em C, por exemplo, o *token* identificador possui o seguinte padrão de formação: *ID* = (_ + letra)(_ + letra + digito)*

Dado que o padrão de formação dos lexemas de um token são expressões regulares, os lexemas podem ser reconhecidos através de autômatos finitos. Este trabalho consiste em implementar um analisador léxico (que é um autômato finito) que percorra um arquivo fonte e encontre os tokens presentes no arquivo. Os tokens que devem ser reconhecidos com seus respectivos padrões de formação de lexemas são dados abaixo:

Token	padrão de formação
ID	$letra(letra + digito)^*$
$nreal$	$digito^+.digito^*$
$nint$	$digito^+$
$nstring$	$"(letra + digito + simbolo)^*"$
op	$'+' + '-' + '*' + '/' + '='$

Na definição acima, letra é o conjunto das letras maiúsculas e minúsculas do alfabeto latino, dígito é o conjunto dos 10 dígitos numéricos e símbolo é o conjunto dos símbolos do teclado exceto (").

Durante o reconhecimento dos tokens, espaços em branco e quebras de linha devem ser descartados. Logo, o programa deve funcionar como a seguir:

Entrada	Saída
valor = 5555.6 + 324 + "Oi mundo"	ID op nreal op nint op nstring

O que fazer?

Você deverá criar um AFD que reconheça os lexemas de cada token. Deverá criar um programa que implemente esse AFD. Seu programa principal deverá chamar *lexan*. A execução do programa deve ser *lexan <fonte>*. A saída do programa deve ser um segundo arquivo com os tokens reconhecidos.

O trabalho poderá ser implementado em Pascal, C/C++ ou Java. Implementações noutras linguagens não serão aceitas.

O que entregar?

Você deverá entregar um relatório descrevendo todo o seu trabalho. Nesse relatório, você deverá explicar como alcançou a solução do problema, dar exemplos de testes realizados, etc. Em anexo, você deverá fornecer um CD ou enviar um e-mail para *moises.ramos@prof.unibh.br* com a listagem dos códigos-fonte e arquivos de teste.

Considerações gerais

1. O trabalho poderá ser feito por grupos de até três pessoas.

2. Bibliografia recomendada:

   AHO, Alfred V. ; SETHI, Ravi ; ULLMAN, Jeffrey D.; LAM, Monica S. Compiladores: Princípios, Técnicas e Ferramentas . 2a ed. São Paulo: Pearson, 2007.

   LOUDEN, Kenneth C. Compiladores: princípios e práticas . Thomson, 2004.

3. Trabalhos copiados da internet ou de outros colegas sofrerão sanções (perdas parcial ou total dos pontos do trabalho).

4. O trabalho vale 15 pontos.

5. A data de entrega do trabalho será 17/10/12.