EXA869 - MI Processadores de Linguagem de Programação

Problema 1: A primeira etapa de uma tradução

Cronograma

Aula	Data	Assunto
1	04/09	Siecomp
2	06/09	Siecomp
3	11/09	Apresentação do Problema 1
4	13/09	Problema 1
5	18/09	Problema 1
6	20/09	Problema 1
7	25/09	Problema 1
8	27/09	Problema 1

Problema

Uma linguagem de programação é um método padronizado para comunicar instruções para um computador[1], permitindo que um programador especifique precisamente sobre quais dados um computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias.

Uma das principais metas das linguagens de programação é que programadores tenham uma maior produtividade, permitindo expressar suas intenções mais facilmente do que quando comparado com a linguagem que um computador entende nativamente (código de máquina)[2]. Assim, linguagens de programação são projetadas para adotar uma sintaxe de nível mais alto, que pode ser mais facilmente entendida por programadores humanos.

Os programas escritos em linguagens de programação são traduzidos para o código de máquina do computador no qual será executado, em vez de ser diretamente executados. O responsável por realizar esta tradução é o compilador. A primeira etapa desta tradução é a análise léxica, que tem por objetivo identificar e classificar os tokens da linguagem de programação, de acordo com a sua estrutura léxica.

Portanto, você aluno deste MI, tem como primeiro objetivo entender todos os aspectos relacionados a um analisador léxico de uma linguagem de programação, que começa a ser construída por você a partir de agora!

Produto

Cada dupla deverá entregar um código-fonte em linguagem **Java (ou em outra linguagem em concordância com o tutor),** do analisador léxico, considerando a Tabela 1.

O código-fonte do analisador léxico deve ser entregue por e-mail, ao seu respectivo tutor, até às **23h59min** do dia **30/09/2018**. A entrada e a saída para o analisador devem ser feitas por meio de arquivos-texto. O analisador deverá ler um conjunto de arquivos-texto de entrada do diretório chamado **teste**, **localizado na raiz do projeto**. Para cada arquivo analisado será gerado um arquivo de saída no mesmo diretório. No arquivo de saída deve ser impressa a lista de *tokens* e exibidos os erros léxicos, caso existam, de forma padronizada, de acordo com as instruções do seu tutor. Se não houver erros, uma mensagem de sucesso deve ser gravada no arquivo de saída.

É importante destacar que a correção do produto por parte dos tutores será feita por meio de arquivos de teste, seguindo estritamente o padrão de entrada e saída já mencionado. Qualquer discrepância em relação ao solicitado implicará na não correção do produto, e consequentemente, atribuição de nota zero no referido problema.

Em caso de atraso na entrega, perdem-se 2 (dois) pontos por UM dia de atraso. Após um dia de atraso, o trabalho não será mais aceito.

Recursos para Aprendizagem

- AHO, A. V., LAM, M. S., SETHI, R., ULLMAN, J. D. **Compiladores: Princípios, Técnicas e Ferramentas**, 2ª ed., Addison-Wesley, 2008.
- AHO, A. V.; SETHI, S. & ULMMAN, J. D. **Compiladores: princípios, técnicas e ferramentas.** Rio de Janeiro: LTC, 1995.
- LOUDEN, K. C. **Compiladores – Princípios e Práticas.** São Paulo, Thomson, 2004.
- HOPCROFT, J. E. *et al*. **Introdução à Teoria dos Autômatos, Linguagens e Computação**. 1ª edição, Editora Campus, 2002.
- MENEZES, P. F. B. **Linguagens Formais e Autômatos**. 5ª edição, Editora Sagra-Luzzatto, 2005.

[1] Dershem, H. L.; Jipping, M. J. (1995). **Programming Languages. Structures and models**. 2ed. Boston: PWS Publishing Company.

[2] Melo, A. C. V. de; Silva, F. S. C. (2003). **Princípios de Linguagens de Programação**. São Paulo: Edgard Blücher Ltda.

Tabela 1. Estrutura Léxica da Linguagem

Palavras reservadas	**class, const, variables, method, return, main, if, then, else, while, read, write, void, int, float, bool, string, true, false, extends**
identificador	**letra (letra | digito | _)***
número	**(-)? Espaco* digito digito* (. digito (digito)*)?**
dígito	**[0-9]**
letra	**[a-z] | [A-Z]**
operadores aritméticos	**+ | - | * | / | ++ | --**
operadores relacionais	**!= | == | < | <= | > | >= | =**
operadores lógicos	**! | && | ||**
Delimitador de comentários	**// isso é um comentário de linha** **/* isso** **é um comentário de** **bloco */**
Delimitadores	**; | , | (|) | [|] | { | } | .**
Cadeia de caracteres	**" (letra | digito | Simbolo | \")* "**
Simbolo	**ASCII de 32 a 126 (exceto ASCII 34)**
Espaco	**ASCII 9 | ASCII 32**