Trabalho II:
# Construção de um Analisador Léxico

Linguagens Formais e Compiladores
Prof$^a$. Jerusa Marchi

1. Definição do Trabalho:

   Construir um analisador léxico para uma linguagem de alto nível seguindo as definições:

   (a) A linguagem deve ser simples, tipada, contendo pelo menos uma estrutura de repetição e uma estrutura condicional, deve aceitar comando de leitura e escrita.

   (b) Deve aceitar operações algébricas simples (manipulação de variáveis não indexadas) e declarações e operações de estruturas de dados tipo "array" unidimensional e multidimensional.

   (c) Deve aceitar comandos de comparação (pelo menos: maior e diferente) e operadores lógicos (pelo menos "e", "ou" e "negação")

   (d) Deve permitir escrita de "strings"(identificadas como constantes).

   (e) O nome das variáveis deve poder ser feito usando "_".

   (f) Deve permitir a declaração de constantes numéricas e booleanas.

2. Passos para construção do analisador:

   (a) gere os autômatos **mínimos** para cada item da linguagem (tome cuidado para que o único estado comum entre estes autômatos seja o estado inicial)

   (b) faça a união de todos os autômatos gerados:

   $$\bigcup_{i=1}^{n} AFD_i$$

   Este autômato deve ser determinizado, mas não minimizado.

   (c) acrescente um estado de erro ao autômato. Este estado de erro deve ser aceitador e uma vez nele, um erro deve ser reportado.

   (d) implementar o autômato em uma linguagem de programação da escolha do grupo. O autômato será facilmente implementado por uma estrutura de dados do tipo tabela, porém também pode ser implementado implicitamente (Sugiro a leitura do cap 3. do livro o Aho). No reconhecimento de um token, parte-se do estado inicial, determinando-se o novo estado pela coluna correspondente ao caracter lido do arquivo de entrada. Repete-se este procedimento até achar um separador de token (espaço em branco, quebra de linha ou final do arquivo). Quando um separador de token é encontrado, verifica-se qual é o estado atual do autômato. Se for um estado

não-final, houve erro, que deve ser acusado. Se for um estado final, verifica-se o que o estado representa, e retorna o valor desejado. O token é montado pelo par (<entrada-lida>, <tipo>).

(e) a saída do analisador deve ser um arquivo onde cada linha descreve um token. O analisador deve "se recuperar" de erros léxicos, portanto, ao reportar um erro, o autômato volta para o seu estado inicial e prossegue até o processamento integral do código fonte. Use o **modo pânico de recuperação de erros**, ou seja, descarte símbolos da entrada até localizar um separador.

**Cada Grupo deve fazer uso do próprio trabalho I, ou seja, as uniões, determinizações, etc. devem ser feitas chamando aquelas do trabalho I.**

3. Grupos e Formato de Entrega: o trabalho deve ser realizado pelo mesmo grupo do trabalho I. Deve ser entregue via moodle até a data de 30/10. Devem ser entregues os fontes e executáveis, cada arquivo deve ter cabeçalho indicando os integrantes do grupo. Também devem ser entregues programas exemplos. O grupo deve ainda "produzir" um pequeno manual da linguagem: Apresentando a sua gramática e detalhando como a sua linguagem pode ser utilizada (estilo "escrevendo seu primeiro programa em *my language 1.0*-versão $\alpha$".

4. Apresentação: A apresentação do trabalho será a partir do dia 03/11, em horários a serem agendados.