Trabalho de Linguagens Formais

Objetivo

Especificar e implementar um analisador léxico-sintático para a linguagem especificada neste documento.

Metodologia

A especificação do analisador léxico-sintático deverá ser realizada antes de sua implementação. A análise léxica deverá ser especificada através de um AF e a análise sintática através de uma GLC.

O analisador léxico-sintático deverá ser implementado de modo que o programa principal faça a leitura de um arquivo texto com o código fonte, realize a análise léxico-sintática do código e exiba uma mensagem informando se a análise foi completada com sucesso ou se houve erro. Em caso de erro, a mensagem deverá informar o **tipo de erro** (léxico ou sintático) e ainda em que **linha** e **coluna** do código em que ele foi encontrado. O programa deve abortar ao encontrar o primeiro erro.

Este trabalho é composto basicamente pelos seguintes componentes:

1. **O programa principal:** que deve abrir o arquivo de entrada e fazer a chamada da função *sintatico*(), a qual inicia a análise sintática do código fonte do arquivo de entrada até que o final deste seja encontrado ou ocorra um erro.

2. **O analisador léxico:** que deverá ser especificado e implementado. Ele deve ser implementado em uma função denominada *le_token*(). Esta função deve **implementar o AF** definido para especificar o analisador léxico, sendo que, a cada chamada desta função deve retornar o token corrente do código fonte.

3. **O analisador sintático:** que deverá ser especificado e implementado. O analisador a ser implementado é um **analisador recursivo preditivo**, baseado na gramática que o especifica. Ele deve ser implementado em uma função denominada *sintatico*(). As chamadas da função *le_token()* devem ser feitas pelas funções do analisador sintático cada vez que houver a necessidade de analisar um token.

O trabalho deve ser desenvolvido em grupos de 3 componentes. Os grupos deverão responder ao tópico criado no fórum do AVA informando os componentes do grupo. A implementação pode ser feita em qualquer linguagem de programação de alto nível.

Além disso, o trabalho deverá ser apresentado e para isso, os grupos deverão marcar um horário (fora do horário de aula) para a apresentação. Informarei meus horários disponíveis para que os grupos agendem suas apresentações. Quem marcar antes poderá escolher o horário.

Calendário	
08/05	Definição dos grupos
28/06	Entrega dos documentos via AVA
29/06 a 03/07	Apresentações

Formato de entrega

Entregar um documento (em formato PDF) contendo o AF que especifica o analisador léxico e a GLC que especifica o analisador sintático. Além disso, também deverá ser entregue o código fonte do analisador léxico-sintático. Esses documentos devem ser entregues compactados em um único arquivo através do AVA.

Especificação da Linguagem

A linguagem para a qual deve ser construído o analisador léxico-sintático é uma linguagem que trabalha com expressões proposicionais, isto é, fórmulas com constantes e operadores da Lógica Proposicional.

- A linguagem possui os seguintes comandos:

  1. comando de atribuição:

     ```
     variável =:= expressão
     ```

  2. comando de exibição:

     ```
     Print expressão_1, expressão_2, ..., expressão_n
     ```

  3. comando de entrada de dados:

     ```
     Read variável_1, variável_2, ..., variável_n
     ```

4. comando condicional:

```
 if expressão {
    comandos_1
}
else {
    comandos_2
}
```

- Os comandos de atribuição, exibição e de entrada de dados devem ser terminados por ponto-e-vírgula.

- Os nomes das variáveis são formadas por uma seqüência de letras, _ e/ou dígitos. Mas, sempre devem iniciar por letra ou _.

- A linguagem é *case-sensitive*.

- As expressões são formadas por parênteses balanceados, valores lógicos, variáveis e operadores lógicos.
    - Os valores lógicos podem ser:
        - `1`
        - `0`
        - `V`
        - `F`
        - `false`
        - `true`

    - Os operadores permitidos são descritos na tabela a seguir:

Operador	Símbolo	Precedência	Associatividade	Exemplos de Expressão
Negação	'	1	-	a', (aux ->x)'
Conjunção	^	2	À esquerda	a ^ b
Disjunção	v	2	À esquerda	a v b
Implicação	->	3	À direita	a -> b
Equivalência	<->	4	À direita	a <-> b

Exemplos de programas sintaticamente corretos:

```
Read a;
Print a';
```

```
Read a, b,c;
Print a ^ b, a v b, a -> c;
```

```
Read a,b,c;  d =:= a ^ (b <-> c);
if (d) { out d ^ a; }
else { out d ^ c; }
```

```
a1 =:= V;
b1 =:= F;
cc1 =:= a1 -> b1;
Print a1, cc1 ^ 1;
```

```
Print V ^ F -> true <-> false;
```

```
Read a,b,c;
d =:= (b' v a) ^ ((b <-> c) v (c'' ^ a)')';
if a ^ (b <-> c) {
e =:= 1;
f =:= 0;
Print d ^ a;
}
else {
      if d {
         e =:= 0;
      }
      else{
         f =:= 1;
      }
      Print d v a;
}
Print e ^ f;
```

Avaliação

Na avaliação do trabalho serão considerados os seguintes pontos:

- Especificação da análise léxica;
- Implementação do analisador léxico (de acordo com o autômato especificado);
- Especificação da análise sintática;
- Implementação do analisador sintático (de acordo com a gramática especificada);
- Mensagens de erro;
- Pontualidades na entrega de todas as etapas.