Descrição do Trabalho Prático 01 de Compiladores

Eduardo Miranda

Março 2024

1 Introdução

O trabalho prático (TP) consiste na implementação de um **Analisador Léxico (Parser)** para uma versão simplificada da linguagem de programação Pascal. Tal analisador pode ser implementado em qualquer linguagem de programação.

2 Proposta de TP

Conforme mencionado, o TP proposto consiste em implementar analisador léxico para uma simplificação da linguagem Pascal, chamaremos essa nova linguagem de Pascal−−.

2.1 Sobre o Parser

O programa resultante deve possibilitar que o usuário informe um arquivo escrito na linguagem Pascal−− e retorne uma lista com os seus respectivos (token, lexama, linha e coluna) para cada token encontrado no arquivo.

2.1.1 Possíveis tipos de Tokens

O parser poderá possuir os seguintes tipos de tokens:
Operadores aritméticos:

- Adição: +;
- Subtração: -;
- Multiplicação: *;
- Divisão Real: /;
- Módulo: 'mod';
- Divisão Inteira: 'div';

Operadores lógicos, relacionais e atribuição:

- OU lógico: 'or';
- E lógico: 'and';
- NÃO lógico: 'not';
- Igualdade: ==;
- Diferença: <>;
- Maior que: >;
- Maior igual que: $\geq$;
- Menor que: <;
- Menor igual que: $\leq$;
- Atribuição: :=

Palavras reservadas:

- **program:** palavra reservada para iniciar o programa;
- **var:** palavra reservada para iniciar a declaração de variáveis;
- **integer:** identifica declaração de variáveis inteiras;
- **real:** identifica declaração de variáveis ponto flutuante;
- **string:** identifica declaração de variáveis do tipo string;
- **begin:** identifica o inicio do programa ou de um novo bloco de comandos ;
- **end:** identifica o fim do programa ou o fim de bloco de comandos;
- **for:** identifica início de bloco de repetição do tipo for;
- **to:** identifica até quando o comando for deve se repetir;
- **while:** identifica início de bloco de repetição do tipo while;
- **do:** palavra reservada para indicar o inicio de um laço de repetição;
- **break:** chamada para sair de um laço de repetição;
- **continue:** chamada para pular uma iteração de um laço de repetição;
- **if:** identifica início de bloco de condicional;
- **else:** identifica segunda parte do bloco condicional caso não verdadeiro;

- **then:** palavra reservada para indicar o inicio de uma estrutura condicional;
- **write:** chamada para mostrar alguma informação na tela;
- **read:** chamada para ler alguma informação do teclado;

Símbolos aceitos:

- **;** - ponto e virgula;
- **,** - virgula;
- **.** - ponto final;
- **:** - dois pontos;
- **(** - abre parênteses;
- **)** - fecha parênteses

Strings:

- Toda string deve começar e terminar com o caractere **"**.
- Strings podem contem o caractere '\', dependendo da linguagem será necessário algum tipo de tratamento especial.

Números:

- Números inteiros são escritos seguindo a seguinte expressão regular: $[0..9]^+$
- Números flutuantes são escritos seguindo a seguinte expressão regular: $[0..9]^+.[0..9]^*$

  Observação: Como a expressão regular dos números flutuantes pode não ter nada na segunda parte e isso pode tornar mais complexa as próximas etapas. Transformaremos a expressão em $[0..9]^+.[0..9]^*0$ visto que adicionar um '0' a direita depois da virgula não vai modificar nosso número original. Esse zero será adicionado somente a nível de lexema, o fonte original não tem a obrigação de colocar esse zero no fim de um número flutuante.

Variáveis:

- As variáveis são escritas seguindo a seguinte expressão regular:

$$([a-z]||[A-Z])([a-z]||[A-Z]||[0-9])^*$$

Observação: Esta expressão regular que gera os lexemas das variáveis dos programas também gera as palavras reservadas, logo será necessário criar tratamento especificos para identificação dos tokens corretos.

Comentários: Os comentários podem ser escritos de duas formas:

- Comentários de uma única linha que começam com '//':
- Comentários de múltiplas linhas que começam com '{' e terminam com '}':

3 Algumas Dicas

Durante o desenvolvimento, é importante não se perder nos detalhes. Portanto, é recomendado que a implementação inicie somente pelas funcionalidades básicas. Só depois de garantir que as funcionalidades básicas estão funcionando conforme planejado, deve-se considerar a implementação de melhoramentos e funcionalidades adicionais. Também recomenda-se que trechos mais complicados do código sejam acompanhados de comentários que esclareçam o seu funcionamento/objetivo/parâmetros de entrada e resultados.

O TP também será avaliado em termos de:

- Modularização;
- Legibilidade (nomes de variáveis significativos, código bem formatado, uso de comentários);
- Consistência (formatação uniforme);

4 Funcionalidades

As seguintes funcionalidades devem estar funcionando corretamente ao entregar o trabalho prático:

4.1 Funcionalidade 1

Retornar uma lista de (token, lexema, linha, coluna) ao executar o parser para algum arquivo escrito na linguagem Pascal−− arbitrário.

4.2 Funcionalidade 2

Quaisquer tokens identificados fora dos padrões apresentados na subseção 2.1.1 devem apresentar um erro de token inválido para a linguagem apontando a linha e coluna que está o erro.

4.3 Funcionalidade 3

Permitir que o arquivo a ser compilado seja informado via terminal e mostre a lista dos tokens, exemplo:

```
python3 <analisadorLexico.py> <arqTeste.pmm>
```

5 Submissão

Além de serem submetidos via link no Portal Didático da disciplina, o TP também deve ser enviado para o endereço eletrônico `eduardomiranda@cefetmg.br`, tendo como assunto TP_Compiladores_2024_1. No corpo do email deve aparecer o nome completo de cada integrante do grupo e um arquivo .zip com o programa e o manual de utilização.