Universidade Tecnológica Federal do Paraná
Curso de Engenharia da Computação
Compiladores
Prof. Marco Antonio Barbosa

Trabalho Prático de Implementação 1.

Objetivo: Construção de um analisador léxico simplificado.

Data de entrega/apresentação: a definir

1. A Construção do Analisador Léxico

A função do analisador léxico é reconhecer as palavras que fazem parte de uma linguagem (lexemas) e classificar estar palavras (determinar os tokens). Portanto, caberá ao analisador léxico *ler um arquivo texto como entrada* e *fornecer como saída a lista dos tokens reconhecidos.*

Para esta etapa do trabalho não será necessário o uso da ***tabela de símbolos*** portanto a lista de tokens será formada por uma tupla:

<nome_do_token, posição_do_token>

Onde:
nome_do_token – indica a classe do token reconhecido
posição do token - indica a linha em que determinada palavra foi reconhecida.

Por exemplo:

Entrada: valor := 50 ;
Saída: <Id, 1><Atrib, 1><Num, 1><Scolon, 1>

Sugestão: Como ponto de partida estabelecer os tokens para todos os lexemas da linguagem. Por exemplo:

Lexema	Token
identificador	Id
:=	Atrib
Numero	Num
;	Scolon
programa	Prog
Se	if
...	...

A escolha pela estratégia de implementação fica a cargo do aluno, portanto, poderá ser implementada uma das três alternativas vistas:
código, tabela de transição ou geradores automáticos (FLEX, JFLEX, etc.).

2. Especificação de uma Linguagem Simplificada de Programação

2.1 Descrição BNF da Linguagem Simplificada

A linguagem que será definida representa uma linguagem de programação estruturada semelhante à linguagem de programação estruturada PASCAL. Esta linguagem receberá o nome de **LPD** (Linguagem de Programação Didática). O compilador a ser desenvolvido receberá o nome de **CSD** (Compilador Simplificado Didático). Os símbolos não terminais da linguagem serão mnemônicos colocados entre parênteses angulares < e >, sendo os símbolos terminais colocados em negrito.

Descrição BNF da Linguagem Simplificada

<programa>::= **programa** <identificador> **;** <bloco> **.**

<bloco>::= [<etapa de declaração de variáveis>]
[<etapa de declaração de sub-rotinas>]
<comandos>

DECLARAÇÕES

<etapa de declaração de variáveis>::= **var** <declaração de variáveis> **;**
{<declaração de variáveis>**;**}

<declaração de variáveis>::= <identificador> {**,** <identificador>} **:** <tipo>

<tipo> ::= (**inteiro** | **booleano**)

<etapa de declaração de sub-rotinas> ::= (<declaração de procedimento>**;**|
<declaração de função>**;**)
{<declaração de procedimento>**;**|
<declaração de função>**;**}

<declaração de procedimento> ::= **procedimento** <identificador>**;**
<bloco>
<declaração de função> ::= **funcao** <identificador>**: <**tipo>**;**
<bloco>

COMANDOS

<comandos>::= **inicio**
<comando>{**;**<comando>}[**;**]
fim

<comando>::= (<atribuição_chprocedimento>|
<comando condicional> |
<comando enquanto> |
<comando leitura> |
<comando escrita> |
<comandos>)

<atribuição_chprocedimento>::= (<comando atribuicao>|
<chamada de procedimento>)

<comando atribuicao>::= <identificador> **:=** <expressão>

<chamada de procedimento>::= <identificador>

<comando condicional>::= **se** <expressão>
entao <comando>
[**senao** <comando>]

<comando enquanto> ::= **enquanto** <expressão> **faca** <comando>

<comando leitura> ::= **leia (** <identificador> **)**

<comando escrita> ::= **escreva (** <identificador> **)**

EXPRESSÕES

<expressão>::= <expressão simples> [<operador relacional><expressão simples>]

<operador relacional>::= (**<>** | **=** | **<** | **<=** | **>** | **>=**)

<expressão simples> ::= [**+** | **-**] <termo> {(**+** | **-** | **ou**) <termo> }

<termo>::= <fator> {(* | **div** | **e**) <fator>}

<fator> ::= (<variável> |
<número> |
<chamada de função> |
(<expressão>**)** | **verdadeiro** | **falso**
nao <fator>)

<variável> ::= <identificador>

<chamada de função> ::= <identificador >

NÚMEROS E IDENTIFICADORES

<identificador> ::= [_]<letra> {<letra> | <dígito>}

<número> ::= <dígito> {<dígito>}

<dígito> ::= (**0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**)

<letra> ::= (**a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z|**
A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z)

COMENTÁRIOS

Uma vez que os comentários servem apenas como documentação do código fonte, ao realizar a compilação deste código faz-se necessário eliminar todo o conteúdo entre seus delimitadores.

delimitadores : { }

Exemplo de Execução

Entrada: test.l

```
programa test ;
 var v : inteiro ;
      i , max , juro : inteiro ;
 inicio
      enquanto v <> 0 faca
      inicio
          leia ( v ) ;        {leia o valor inicial}
          leia ( juro) ;    {leia a taxa de juros }
          leia ( max ) ;   { Leia o periodo };
          valor := 1 ;
          i := 1 ;
          enquanto i <= max { (1+juro) elevado a n } faca
          inicio
             valor := valor * ( 1 + juro) ;
             i := i + 1 ;
          fim
          escreva ( valor ) ;
      fim
 fim .
```

saída:

<PROG, 1><ID, 1><PVIRG, 1><VAR, 2>...<DOT, 20>

Referências

Flex & Bison. John Levine, 2009.
Compiladores. Princípios, Técnicas e Ferramentas. Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman.
Compiladores Princípios e Práticas. Kenneth C. Louden.
Implementação de Linguagens de Programação: Compiladores. Ana Maria de Alencar Price e Simão Sirineo Toscani