

# Construção de Compiladores

## Período Especial

### Aula 6: Primeiras Regras

Bruno Müller Junior

Departamento de Informática  
UFPR

2020

- 1 Objetivos
- 2 Declaração de Variáveis
  - Regras Sintáticas
  - Ações do compilador
- 3 Comando Composto
  - Regras Sintáticas
- 4 Atribuição
  - Regras Sintáticas
- 5 Entrada/Saída
- 6 Próxima Aula

# Objetivos

- Após a aula de hoje, o compilador desenvolvido pelos alunos deverá ser capaz de receber o código da esquerda como entrada e gerar os comandos da direita.

```
program varsGlobais (input, output);  
var a, b: integer;  
    k1, temp : integer;  
begin  
    a:=0;  
    temp:=10;  
end.
```

```
INPP  
AMEM 5  
CRCT 0  
ARMZ 0,0  
CRCT 10  
ARMZ 0,3
```

# Regras Sintáticas

## ● Regras para declaração de variáveis: 2,8,9,10

1. <programa> ::= **program** <identificador> (<lista de identificadores>);  
                  <bloco>.
2. <bloco> ::= ...  
              [ <parte\_declara\_vars> ]  
              ...
8. <parte\_declara\_vars> ::= **var** <declara\_vars> {; <declara\_vars> };
9. <declara\_vars> ::= <lista\_identificadores> : <tipo>
10. <lista\_identificadores> ::= <identificador> { , <identificador> }

## ● Exemplos:

```
var a1: integer;  
var a2, b2: integer;  
var a3: integer;  
    b3: integer;
```

# Ações do compilador

- As regras tem de ser convertidas para a notação LR.
- Ao encontrar a regra 8 (parte declara variáveis), o compilador deve:
  - inserir os símbolos na Tabela de Símbolos;
  - gerar o código AMEM  $k$ , onde  $k$  é o número de variáveis.
  - problemas:
    - Ao encontrar a var. simples, sabe-se o nível léxico e deslocamento. Já o tipo das variáveis, só se sabe ao fim da declaração. Como atualizar o tipo?
    - Como tratar o caso abaixo (vários AMEM)?

```
1 program declaraVars (input, output);
2     var a, b: integer
3         c, d, e: integer
4         f: integer
5     ...
6 end.
```

```
1      INPP
2      AMEM 2
3      AMEM 3
4      AMEM 1
5      ...
6      PARA
```

# Regras Sintáticas

- O comando composto é quem tem as regras de tradução dos comandos pascal, e será alvo de várias aulas.
- Hoje, só atribuição.

```
16. <comando composto> ::= begin <comando>
                                { ; <comando> } end
17. <comando> ::= [numero :] <comando sem rótulo>
18. <comando sem rótulo> ::= <atribuição> |
                                <chamada de procedimento> |
                                <desvio> |
                                <comando composto> |
                                <comando condicional> |
                                <comando repetitivo>
```

# Regras Sintáticas

- Primeira regra de comando composto;

19. `<atribuição> ::= <variavel> := <expressão>`

- Desafios:

- não for um símbolo  $\in$  Variável Simples, Parâmetro formal ou Função, deve indicar erro.
- Calcular o tipo resultante da expressão, como por exemplo em: `a:=a+b` ou `a:=a>b` ou `a:=1`. Este será o tópico da próxima aula.
- Como armazenar o elemento da esquerda para fazer a comparação com o resultado da expressão.
  - Sugestão: variável global `l_elem`.

# Entrada/Saída

Instrução	Ação	Significado
LEIT	$s := s + 1;$ $M[s] := stdin$	Leitura
IMPR	$stdout := M[s]$ $s := s - 1$	Imprime

```
...
read(a,temp);
...
write(10,k1,a);
```

```
...      | CRCT 10
LEIT     | IMPR
ARMZ 0,0 | ARMZ 0,0
LEIT     | IMPR
ARMZ 0,3 | CRVL 0,3
...      | IMPR
```



# Próxima Aula

```
program varsGlobais (input, output);  
var a, b: integer;  
    k1, temp : integer;  
begin  
    a:=0;  
    temp:=10;  
    k1:=(a+temp) div 1;  
    k1:=(a+temp) > 1; (* Acusa Erro *)  
end.
```

```
INPP  
AMEM 5  
CRCT 0  
ARMZ 0,0  
CRCT 10  
ARMZ 0,3
```

- Página para anotações

# Licença

- Slides desenvolvidos somente com software livre:
  - $\text{\LaTeX}$  usando beamer;
  - Inkscape.
- Licença:
  - Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License. <http://creativecommons.org/licenses/by-nc-nd/2.5/br/>