Sintaxe do Pascal Simplificado Estendido de 12 novas construções em Notação EBNF (BNF estendida)

- Não-terminais são nomes mnemônicos colocados entre parênteses angulares.
- Vocabulário terminal formado por identificadores, número, símbolos especiais simples e compostos, e palavras-chaves que aparecem negrejadas.
- Símbolo inicial é <programa>.
- Conjunto de produções dado abaixo.

As expressões podem ser **inteiras ou booleanas** e a distinção deverá ser feita pelo compilador.

Programa e Bloco

1. <programa> ::=

program <identificador> **;**

<bloco>**.**

2. <bloco> ::=

[<parte de definições de constantes>]

[<parte de definições de tipos>]

[<parte de declarações de variáveis>]

[<parte de declarações de sub-rotinas>]

<comando composto>

Declarações

3. <parte de definições de constantes> :: =

const <definição de constante> {; <definição de constante>}

4. <definição de constante> ::= <identificador> = <constante>

5. <constante> ::= [+|-] (<identificador> | <numero_inteiro>)

6. <parte das definições de tipo> ::=

type <definição de tipo> {; <definição de tipo> }

7. <definição de tipo> ::= <identificador> **=** <tipo>

8. <tipo> ::=

<identificador> |

<enumerado> |

<record> |

array [<indice>] **of** <tipo> |

string [<número_inteiro**>]**

9. <indice> ::= < número_inteiro > .. < número_inteiro >

10. <enumerado> ::= (<lista de identificadores>)

11. <record> ::= record <lista de campos> end

12. <lista de campos> ::= <lista de identificadores> : <tipo> { ; <lista de identificadores> : <tipo> }

13. <parte de declarações de variáveis> ::=

var <declaração de variáveis>

{; <declaração de variáveis>**};**

14. <declaração de variáveis>::=

<lista de identificadores> **:** <tipo>

15. <lista de identificadores> ::= <identificador> **{,** <identificador>}

16. <parte de declarações de subrotinas> ::=

{<declaração de procedimento> ; | <declaração de função> ;}

17. <declaração de procedimento> ::=

procedure <identificador> [<parâmetros formais>] **;** <bloco>

18. <declaração de função> ::=

function <identificador> [<parâmetros formais>] : <identificador> ; <bloco>

19. <parâmetros formais> ::=

(<seção de parâmetros formais> { **;** <seção de parâmetros formais>} **)**

20. <seção de parâmetros formais> ::=

[**var**] <lista de identificadores> **:** <identificador>

OBS: Todos os identificadores devem ser declarados antes de serem utilizados. Isto implica a impossibilidade de se utilizar recursão múltipla entre subrotinas quando elas não estão declaradas uma dentro da outra. São considerados **identificadores pré-declarados** os identificadores de tipo simples **integer** e **boolean**, os identificadores de procedimentos **read** e **write**, e os identificadores de constantes **true** e **false**.

Comandos

21. <comando composto> ::= **begin** <comando> { **;** <comando>} **end**

22. <comado> ::=

<atribuição>

| <chamada de procedimento> → **read/write são procedimentos pré-definidos**

| <comando composto>

| <comando condicional 1>

| <comando condicional 2>

| <comando repetitivo 1>

| <comando repetitivo 2>

| <comando repetitivo 3>

23. <atribuição>::= <variável> **:=** <expressão>

24. <chamada de procedimento> ::=

<identificador> [**(** <lista de expressões> **)**]

25. <comando condicional 1> ::=

if <expressão> **then** <comando>

[**else** <comando>]

26. <comando condicional 2> ::=

case <expressão> **of** <elemento do case> { ; <elemento do case> } **end**

27. <elemento do case>::= <constante> { , <constante> } : <comando>

28. <comando repetitivo 1> ::=

while <expressão> **do** <comando>

29. <comando repetitivo 2> ::= **for** <identificador> := <expressão>

(**to|downto**) <expressão> **do** <comando>

30. <comando repetitivo 3> ::= **repeat** <comando> { **;** <comando>} **until** <expressão>

OBS: Os comandos de entrada e saída estão incluídos nas chamadas de procedimentos. Supomos que existe um arquivo padrão de entrada contendo apenas números lidos pelo comando read (v1,v2, ...vn), em que v1, v2, ... vn são variáveis inteiras. Os pormenores sobre o formato do arquivo de entrada serão ignorados. Analogamente, o comando write (e1,e2, ...en) imprimirá os valores das expressões inteiras e1, e2, ...en num arquivo padrão de saída.

Expressões

31. <expressão> ::=

<expressão simples> [<relação> <expressão simples>]

32. <relação> ::= **=** | **<>** | **<** | **<=** | **>=** | **>**

33. <expressão simples> ::=

[**+** | **-**] <termo> {(**+** | **-** | **or**) <termo>}

34. <termo> ::=

<fator> {(***** | **div** | **and** | **/**) <fator> }

35. <fator> ::=

<variavel>

| <número_inteiro>

| <número_real>

| <string>

| <char>

| <chamada de função>

| **(** <expressão> **)**

| **not** <fator>

36. <variável> ::= <identificador> [(**[** <expressão> **]** **|** **.** <campo>)]

37. <lista de expressões> ::= <expressão> {, <expressão>}

38. <chamada de função> ::= <identificador> [(<lista de expressões>)]

39. <campo> ::= <identificador>

EBNF:

[α] = α | λ

{α} = repetição da cadeia α zero ou mais vezes

α | β = α ou β devem ser escolhidos