Sintaxe de FRANKIE em Notação EBNF (BNF estendida)

EXTENSÃO DO GRUPO:

FRANKIE possui a **definição de programa** e de **bloco**, a gramática de **expressões**, a **passagem e a lista de parâmetros** e definição de **procedimentos** de PASCAL.
A declaração de **variáveis** e os **tipos simples** (built-in) são de C, embora o tipo "Boolean" vem do Pascal para que haja diferença explícita entre inteiros e booleanos (TRUE/FALSE)
Os **comandos** IF e WHILE são do C, os outros do Pascal.
As **expressões** podem ser inteiras ou booleanas e a distinção deverá ser feita pelo compilador (nesta gramática não há checagem de tipo)
A forma dos **comentários** (a lá C ou a lá Pascal) será decidida pelo grupo.
A formação de **identificadores e palavras-chaves** (caixa) será decidida pelo grupo

G = (VN, VT, P, S)

- VN: são nomes mnemônicos colocados entre parênteses angulares.
- VT: formado por identificadores, número, símbolos especiais simples e compostos, e palavras-chaves que aparecem negrejadas.
- S: <programa>.
- P: dado abaixo.

Programa e Bloco

1. <programa> ::= **program** <identificador> **;**
   <bloco>**.**
2. <bloco> ::= [<parte de declarações de variáveis>]
   [<parte de declarações de procedimentos>]
   <comando composto>

Declarações

3. <parte de declarações de variáveis> ::= <declaration> {**;** <declaration>}**;**
4. <declaration> ::= <declaration-specifiers> <declarator list>
5. <declaration-specifiers> ::= <type-specifier>

5.1 <type-specifier>::= (**boolean** | **int**)

6. <declarator list> ::= <identificador> {**,** <identificador>}
7. <parte de declarações de procedimentos> ::=
   {declaração de procedimento> **;**}
8. <declaração de procedimento> ::=
   procedure <identificador>
   [<parâmetros formais>] **;** <bloco>
9. <parâmetros formais> ::=
   (<seção de parâmetros formais> { **;** <seção de parâmetros formais>} **)**
10. <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 procedimentos **read** e **write** e os identificadores de constantes **true** e **false**.

Comandos

11. <comando composto> ::= **begin** <comando> { **;** <comando>} **end**
12. <comado> ::=

    <atribuição>

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

    | <comando composto>

    | <comando condicional 1>

    | <comando repetitivo 1>
13. <atribuição>::= <variável> **:=** <expressão>
14. <chamada de procedimento> ::=

    <identificador> [**(** <lista de expressões> **)**]
15. <comando condicional 1> ::=

    if (<expressão>**)** <comando>

    [**else** <comando>]
16. <comando repetitivo 1> ::=

    while (<expressão> **)**<comando>

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

17. <expressão> ::= <expressão simples> [<relação> <expressão simples>]
18. <relação> ::= **=** | **<>** | **<** | **<=** | **>=** | **>**
19. <expressão simples> ::= [**+** | **-**] <termo> {(**+** | **-** | **or**) <termo>}
20. <termo> ::= <fator> {(***** | **div** | **and**) <fator> }
21. <fator> ::= <variavel>

    | <número>

    | **(** <expressão> **)**

    | **not** <fator>
22. <variável> ::= <identificador>
23. <lista de expressões> ::= <expressão> {, <expressão>}

EBNF:

[α] = α | λ {α} = repetição da cadeia α zero ou mais vezes α | β = α ou β devem ser escolhidos